Draagt een kennismodel bij aan het meer concreet maken van een informatie architectuur?

Toegevoegde waarde van een kennismodel bij onderhoud van software

Groosman, S. (Sebastian)

Award date:
2020

Draagt een kennismodel bij aan het meer concreet maken van een informatie architectuur?

Toegevoegde waarde van een kennismodel bij onderhoud van software

Cursus:	Afstuderen BPMIT
Student:	Sebastian Groosman
Identiteitsnummer:	
Datum rapport:	17 februari 2020 2.0
Versie nummer:	Definitief
Status:	

Draagt een kennismodel bij aan het meer concreet maken van informatie architectuur?

Toegevoegde waarde van een kennismodel bij onderhoud van software

Can a knowledge model help to make an information architecture more concrete?

Benefit of a knowledge model on software maintenance

Opleiding:	Open Universiteit, faculteit Management, Science & Technology Masteropleiding Business Process Management & IT
Program:	Open University of the Netherlands, faculty of Management, Science & Technology Master Business Process Management & IT
Cursus:	IM9806 Afstudeeropdracht BPMIT
Student:	Sebastian Groosman
Identiteitsnummer:	851120657
Datum:	17 februari 2020
Afstudeerbegeleider:	Stef Joosten
Meelezer:	Rogier van de Wetering

Abstract

Software documentation and architecture design is an important aspect of the software lifecycle and improvement on this has been going on for years. With the rise of Agile development, the goal is set to create a working system over comprehensive documentation. In this research a between-subjects Family of Experiments is conducted to investigate if a previously created conceptual model helps internal and external expert maintainers of known software systems to get a better understanding of the system and processes. With the intent of creating better impact analysis and even re-documenting the system.

Keywords
Software maintenance; conceptual model; UML; knowledge transfer

Summary

Software documentation and architecture design is an important aspect of the software lifecycle. With the rise of Agile development, documentation is set to a minimal level in order to create a working system. While software engineers will remember the details of their own developments, maintainers and consultants may need extra information to perform maintenance and enhancements.

A large sum of the software budget is spent on maintenance and enhancements and with a general lack of documentation in mind, this research investigated if a previously created conceptual model helps expert maintainers of known software systems to get a better understanding of the individual customer-specific system. This with the goal of creating a better impact analysis and clearer insights.

Literature research pointed out that using models have a positive impact on the domain knowledge and the transfer of knowledge. Recommendations are widely available to use UML diagrams created during the design phase, for software maintenance by novice and expert maintainers.

In a between-subject Families of Experiments with two replications, each experiment was divided in two parts. During the first part the participants gained information on a new system with the use of a UML model or text-only documentation, with the intent to get them acquainted with the system. The transfer of domain knowledge was tested on surface and deep understanding using multiple-choice, problem solving and Cloze tests. In the second part of the experiment, the focus was set on maintenance and enhancements of the then known system, and tested on knowledge transfer.

The statistics revealed no significant difference between the use of a UML model or text-only on surface or deep understanding, either using novice or expert software engineers. There were slight differences detected in time and effectiveness and minor differences in experience levels. The research results further suggested that 'creating a model, could be more relevant than having a model'.

Index

1. Introduction

1.1. Background

This research is done as a part of the curriculum for the master study Business Process Management & IT. Within this research, we teamed-up to investigate and answer a single research question. This with respect on our personal interest and perspective of the problem to adapt the research question and focus on a specific area.

The main topics are concentrated on domain knowledge, conceptual models and information architecture. And how conceptual models contribute to documentation and knowledge transfer, within the software lifecycle.

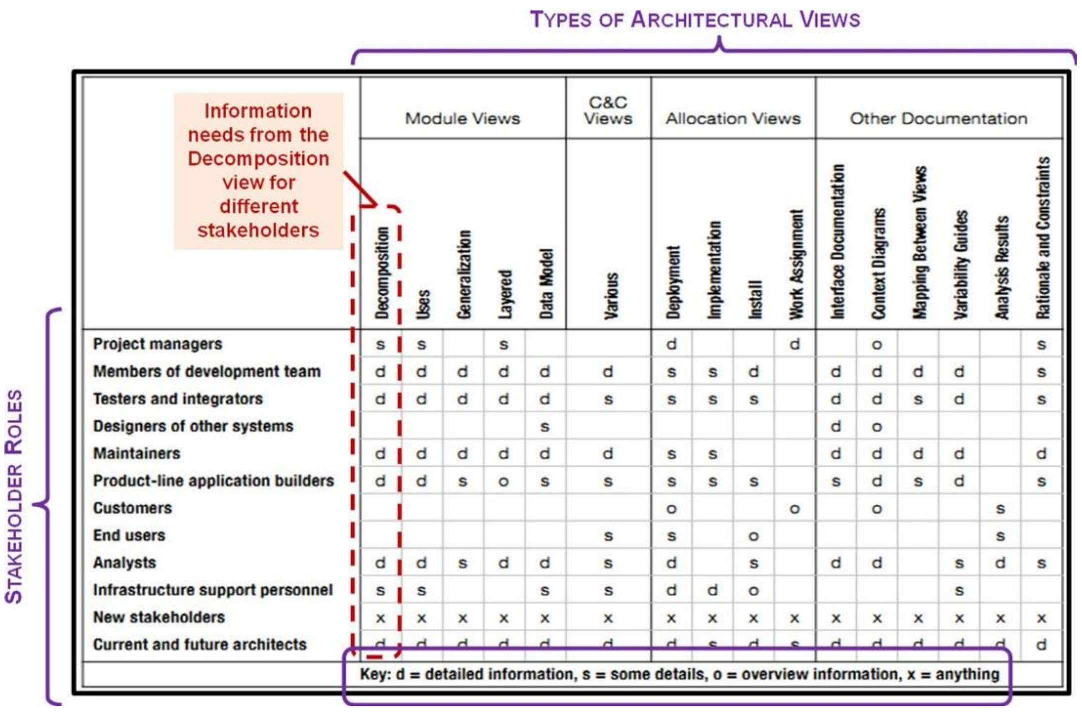
This paper was setup to reflect the research that has been done, working our way up from problem definition to literature research for better understanding and in-depth knowledge on the topics. Resulting in a refinement and a clear definition of the hypotheses, followed by further investigation on how the research experiment was setup, to be assessed and conducted. The statistical analysis and interpretation of these results are provided, with conclusion and suggestions for further research.

1.2. Context

A definition which needed to be clarified was a knowledge model. In our current research area of Business Process Management and IT we discussed and agreed that a knowledge model is a part of an information architecture (IA). In this we adhere that the conceptual model, with enough detail and clarity, is used to define and communicate the information and requirements of the IA domain with the stakeholders, analysts and designers in order to get a shared understanding and knowledge of the domain (Gemino & Wand, 2005). Examples of such models are OWL, data models and UML class diagrams which describe concepts or entities, relations and semantic constraints. Whereas models with a time component like state or sequence diagrams, petri-nets, behavioral and enterprise models are no conceptual models (Aljumaily, Cuadra, & Laefer, 2019).

Within the software lifecycle, the general phases are requirements engineering, modeling and designing, creation, testing to maintenance and updating. This paper focused on software maintenance and the goal was to clarify the usefulness and contribution of knowledge models on the domain knowledge of the stakeholders. According to (Díaz-Pace, Villavicencio, Schiaffino, Nicoletti, & Vázquez, 2016; Paul C et al., 2004), stakeholders of software systems have different information and documentation needs. The relation between the types of architectural views and the stakeholders' needs are depicted in Figure 1. This shows that the maintainer has, amongst others, a need for detailed information on data models, context diagrams and mapping between views. This confirms the need of a conceptual model in general. To immediately abate this, (S. C. de Souza, Anquetil, & de Oliveira, 2005) held a survey amongst 76 software maintainers and surprisingly found that, architectural models were not found to be very important at all. To frame this, there is still room for debate on this topic.

The domain of this research is within information systems (IS) and more in particular within software maintenance of a retail ERP system, where frequent process optimizations are requested, and new applications are introduced which should lead to new business opportunities. In all this tumult of fast change-requests, the level of reliability and good housekeeping request a lot of effort and stresses the maintainers to make fast design decisions. This with little room for error.



Types of Architectural Views
Stakeholder Roles
Information needs from the Decomposition view for different stakeholders

	Module Views					C&C Views	Allocation Views				Other Documentation					
	Decomposition	Uses	Generalization	Layered	Data Model	Various	Deployment	Implementation	Install	Work Assignment	Interface Documentation	Context Diagrams	Mapping Between Views	Variability Guides	Analysis Results	Rationale and Constraints
Project managers	s	s		s			d			d		o				s
Members of development team	d	d	d	d	d	d	s	s	d		d	d	d	d		s
Testers and integrators	d	d	d	d	d	s	s	s	s		d	d	s	d		s
Designers of other systems					s						d	o				
Maintainers	d	d	d	d	d	d	s	s			d	d	d	d		d
Product-line application builders	d	d	s	o	s	s	s	s	s		s	d	s	d		s
Customers							o			o		o			s	
End users						s	s		o						s	
Analysts	d	d	s	d	d	s	d		s		d	d		s	d	s
Infrastructure support personnel	s	s			s	s	d	d	o					s		
New stakeholders	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Current and future architects	d	d	d	d	d	d	d	s	d	s	d	d	d	d	d	d

Key: d = detailed information, s = some details, o = overview information, x = anything

Figure 1 Views and Beyond characterization of stakeholder preferences on architectural views (Díaz-Pace et al., 2016)

1.3. Relevance

When taken into account that more than 60% of the software budget is spent on maintenance and from which 60% is meant for enhancements (S. C. de Souza et al., 2005; Ana M. Fernández-Sáez et al., 2016; Tang, Avgeriou, Jansen, Capilla, & Ali Babar, 2010), there ought to be a minimal amount of documentation required when creating software, with the foresight of future maintenance (Garousi et al., 2015; Zhi et al., 2015).

Software documentation and architecture design are important to the software development live-cycle and is used to represent and communicate the system structure and behavior to all stakeholders (S. C. B. de Souza, Anquetil, & de Oliveira, 2006). As a part of this it was also pointed out that software engineering constantly wants to improve the practice of development and maintenance, in which documentation has long been set as the most important asset. Research identified that creating documentation is often seen as expensive and time consuming and should yield benefits at some point later in the development or maintenance phase (Garousi et al., 2015; Garousi, Garousi, Moussavi, Ruhe, & Smith, 2013). With the rise of Agile methods, software development shifts towards an approach in which documentation should be brief and just precise enough (Kajko-Mattsson, 2008; Stettina & Heijstek, 2011). On this matter (Abdullah, Honiden, Sharp, Nuseibeh, & Notkin, 2011) noticed a common understanding of the requirements as being the most important aspect, with the shift from story cards to a shared mental model and conceptualization amongst the Agile team members and the customers. In this same context, research questioned if Agile projects produce enough documentation for future maintenance (S. C. B. de Souza et al., 2006; Kajko-Mattsson, 2008).

On a personal note, as an IT manager in retail, yearly a large sum of my budget is spent on maintenance and external consultants to make the required changes in our software systems. I am therefore always keen to get answer to the question ‘How much documentation is enough?’ and as a part of this ‘Does a conceptual model help in maintaining or enhancing our systems?’, especially for working within known domains as SAP ERP. Having a clear answer will aid in setting documentation and modelling tasks and levels in future projects. But only if future costs - due to a lack of comprehension - outweigh the cost and time of documenting such models at an earlier stage.

Whilst informally discussing the matter of documentation with a number of retailers, they pointed out that they also face similar issues on documenting. For instance:

- Marketing director at Sligro; “we are examining and re-documenting all our processes, especially now that we are moving from our own build ERP system to SAP.”
- IT manager at Hema; “Documentation is always difficult, mostly to get consistency across different projects. We use an online tool where all information is bundled per project. Our Agile teams use a ticketing system, but we find it hard to find back information for maintenance purposes.”

I Briefly questioned other retailers and software vendors from the Netherlands (Hönkemuller, Zeeman, Jumbo, Didi, Hanos, Coop supermarket, VidaXL, Ctac and CowHills) and similar answers were given. Claiming that they or their customers faced issues on documentation and lack of comprehension, to an extent to be the root cause for projects to run longer and tests to be incomplete.

1.4. Problem statement

The initially proposed research question is: ‘Can a knowledge model ease the documentation burden of development teams in information systems?’.

Since this would cover a too large area to investigate, the focus is set on whether expert maintainers get a higher level of domain knowledge with the use of a conceptual model, in addition to written documentation. The answer to this question could help future development projects by setting the minimal documentation requirements, when keeping in mind that the part of the software is reused or maintained (Monperrus, Eichberg, Tekes, & Mezini, 2012).

The outliers in documentation showed that either a lot of time and cost is devoted to documentation or there is a lack of documentation. Although the truth is in generally in the middle; the latter we see more frequently when implementation has been done with Agile teams, whereas less documentation as possible is advocated to get a working system (Stettina & Heijstek, 2011). Whether or not documentation – and to which amount and level – is needed, is a key question in a number of research papers. Amongst others, (Paul C et al., 2004) set out to find the answer to the question on how to document an architecture for others to successfully use it, build a system and maintain it. With that in mind, they came with the approach of documenting the relevant views at the relevant time in the software lifecycle. Figure 1 represents a follow up on the work of (Paul C et al., 2004) and for example the maintainer will have a need for all views.

An important differentiator in the maintenance of an ERP system, compared to a blank slate application, is that the expert maintainer is commonly well aware of the possibilities and enhancements of the system in general. This is also true for software applications that allow customer specific configurations and customizations. In these situations the processes and specific business requirements, which can be captured in writing or conceptual models, will commonly be different in each business situation. As such, the possible solutions are different in each situation and the expertise in problem-solving capabilities of a maintainer or consultant is a dependent variable.

Basically, when starting a maintenance project, the common practice of project management is to start with an impact analysis, prior to the actual work. Within this process the system overview, architecture for specific parts of the system, its processes, key-users and rationale for earlier design decisions are collected. This with the goal to propose possible solutions and estimates of the required work. This is commonly done by business consultants and maintenance experts and it would be interesting to know, if an initial conceptual model helps them improve an impact analysis.

Problem statement:
The maintenance of a software system is burdened with a limited amount of documentation. The work needed to invest in the actual problem and to create an impact analysis, is time consuming. With misinformation this could lead to errors and rework. Does documentation in the form of a knowledge model positively contribute to create a faster and better impact analysis on a known system?

1.5. Research question

There where the Agile manifesto stated that "Working software [is valued] over comprehensive documentation" (Beck, 2001), it is recognize that documentation within Agile development is set to minimal. With more than 60% of software budgets spent on maintenance, of which 60% on enhancements (S. C. de Souza et al., 2005), a minimal amount of documentation is also required for software maintenance (Garousi et al., 2015). The different demands on documentation during the software lifecycle needed by the different stakeholders (Díaz-Pace et al., 2016) is a good starting point to be economical with time used for documenting. (Zhi et al., 2015) researched the cost, benefit and quality of documentation, since ratios of time spent of 11% were reported. (S. C. B. de Souza et al., 2006) experimented with the question on which documentation is actually used by maintainers. In the context of software maintenance, research was conducted by (Arisholm, Briand, Hove, & Labiche, 2006) to investigate the cost effectiveness of UML documentation.

The aim of this thesis is therefore set to investigate if a previously created conceptual model helps expert maintainers of known software systems to get a better understanding of the system. This with the goal of creating a better impact analysis and clearer insights, which according to (Zhi et al., 2015) should result in shortened task duration, improved code quality, higher productivity and other improvements related to software development. So does the use of a conceptual modal tip the balance of domain knowledge for the better?

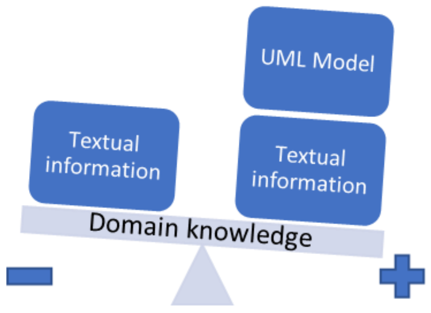


Figure 2 Domain knowledge balance

Research question:
Is there a difference in domain knowledge between the use of a conceptual modal and text-only information when maintaining or enhancing a known system?

2. Literature research

2.1. Process and execution

The literature research was executed in order to find answers within the scientific literature to become more acquainted with the field, and have a common understanding of the problem. The setup of the curriculum, where we started working on the same initial question with 5 team members, gave the opportunity to spread the initial research and to find more relevant literature. The term 'knowledge model' needed the most clarification and the generally known UML-modelling with class diagrams was selected over other conceptual modeling techniques. Mostly because of my personal knowledge in this area and relevant literature that was found in this area.

I used OneNote as a logbook for all my initial search queries in Google Scholar and Harvard Business Review, using the terms 'knowledge model' and 'information architecture'. Reading a number of easy to read papers and articles, provided me with a general overview and common findings on the domain. I found this very helpful as a starter and got more curious on the topics.

From the online university library, the terms 'knowledge model', 'software documentation' and 'software architecture' gave a dozen good papers to read. Along with snowballing the used references from these and the initially shared papers, gave 15 to 20 more relevant papers to read more in-depth. Very welcoming were the generated suggestions by the online libraries, based on my prior searches and preferences and 'people that read similar items'. By then I started using EndNote for safekeeping the references, relevant papers and personal notes on each of them.

The literature search tool 'Publish or Perish' gave a number of relevant abstracts of papers to read, but – frustratingly – most of these papers and books couldn't be accessed from the university library.

Since a number of papers gave insight in the problems concerning the lack of documentation during the maintenance phase, the high amount of costs in maintenance, combined with findings from my work, I repositioned my searches on software maintenance. Because several papers questioned the validity of their research due to the small size, the known domain and learning effect, my attention was grabbed. By using the validity shortcomings to my benefit I focused on the use of models within the maintenance of software by experts with high domain knowledge. Area where domain knowledge is high are ERP systems and applications that allow business specific configuration and customization.

Searching for combinations on 'ERP', 'Maintenance', 'documentation' and models like 'UML' gave me no significant results, other than research papers with the focus on critical success factors for ERP implementations, strategies for maintenance and troubleshooting, and how customization has effect on risks and future costs. This required me to focus and search on 'domain knowledge', 'maintenance of software', 'UML', 'impact analysis' and 'novice vs expert' and how to test knowledge transfer. This helped to get more insights and formed the basis on how to conduct the experiment.

In the end I read about 50 to 60 papers, from which I kept about 25 good references which were used in this thesis. After the experiments, some specific parts needed additional support. Extra literature research was conducted, which gave about 8 additional relevant papers in the area of maintenance and UML modeling, next to a deeper understanding of the initial 25 good references.

2.2. Results and follow up

The literature review focused on getting answers regarding knowledge modeling with UML class diagrams. Because UML is considered the facto standard for software analysis and design modeling (Grossman, Aronson, & McCarthy, 2005), much research was found on this topic. (Gemino & Wand, 2005) stated that a conceptual model, with enough detail and clarity, could be used to define and communicate the information and requirements of the IA domain with the stakeholders, for a shared understanding and knowledge of the domain. (Ding, Liang, Tang, & van Vliet, 2014) did a systematic literature review, with the objective to understand what knowledge-based approaches could be employed to improve the quality of IA documentation, and identified nine benefits from these studies. The model-based approach was used in requirements elicitation, requirements analysis and architecture recovery. They found that model-based approach made it easier to understand requirements and reduced the learning effort to capture and model domain knowledge. This could be used to evaluate the consistency, completeness, traceability, and reusability in requirements analysis.

One of the more in-depth literature searches was on class diagrams, the use and fit in regard to software maintenance and to what level of detail (Ana M. Fernández-Sáez et al., 2016). They investigated if the level of detail (LoD) impacts the performance of (preventive) maintenance tasks, in a model-centric approach in a Families of Experiments using 81 students in 3 countries. Because not all UML models have the same level of complexity, they defined low LoD and high LoD. They questioned to what LoD it is necessary to update UML documentation, in order to fit the source code and achieve benefit during maintenance. According their analysis of the literature, there is limited empirical evidence in the usefulness of UML diagrams in aid of source code maintenance. Evidence by (Arisholm et al., 2006) suggested that the use of UML diagrams benefits maintenance. Again, according to (Ana M. Fernández-Sáez et al., 2016) when the LoD used in a UML diagram is low, it typically contains only a few syntactical features, such as class-name and associations, without specifying any further facts about the class. When it is high, the diagram also includes class attributes and operations, association names, association directionality, and multiplicity. In their experiment no conclusive results were seen in favor of low or high LoD, but the descriptive statistics showed a favor in using low LoD, mainly because the subject found the high LoD hard to read. For our experiment this could implicate that a fairly simple UML class diagram can be used, with the addition multiplicity. For me, the question remains open if formal modeling techniques should be used to capture all business rules.

(A. M. Fernández-Sáez, Caivano, Genero, & Chaudron, 2015) conducted a survey on the use of different UML diagrams in software maintenance, with 178 professionals from 12 countries. They concluded that class diagrams were perceived as helpful during software maintenance, with less time needed for understanding and resulting in improved defect detection. The actual paper was not directly downloadable from the library, only the survey, so quoting (Scanniello et al., 2018) on this. Upon comparing the designs made during the design phase or during the analysis phase (Scanniello et al., 2018) observed that UML models created during design phase have a high LoD and a positive effect on the source-code comprehension, compared to models created during analysis which have a low LoD. This research claimed to be the largest in literature at that point in time, with 12 controlled experiments on different sites and levels of expertise and obtained 333 observations from these experiments. The models used in the analysis phase were created to get insight on the domain of the problem, and the design models were created to get to the bottom of the implementation aspects, thus source-code. They found that there is no advantage in time when considering the time needed to create the design model and the time needed for the actual maintenance. Even that it is actually useless to give UML-based analysis models to the software engineer when performing small maintenance operations on source code. This aligns with the findings of (Arisholm et al., 2006), that stated that the UML model only seemed to be useful with the need of understanding complex systems. This supported the idea that creating an analysis model enhances level of insight on the domain; in addition, domain knowledge might even be better, compared to the use of a previously created model by somebody else than the maintainer.

Related findings were read in (Ana M. Fernández-Sáez et al., 2015) where diagrams with a high LoD are reported to be more helpful during software development, while those with a low LoD seemed to be better when performing maintenance tasks. With their Families of Experiments, they show that forward designed UML diagrams are slightly better understood than reversed engineered diagrams, with the rationale that they provide a more attractive balance between detail and relevant information. Within the same paper they also found that source code is the most useful source of information on which the UML diagram added only little information on assessing the source code. The main finding of this study was that participants with a higher ability (experts) achieved better scores when using the diagrams with a forward design, while low ability participants (novices) got better scores when using reverse engineered diagrams. Additional point of interest is that they also gave the opinion that the creation of a model during the design phase of the maintenance is the significant factor, compared to viewing a model created by somebody or something else.

(Gravino, Scanniello, & Tortora, 2015) confirmed that for experienced maintainers it is useless to give additional information other than the source code from a managerial point of view, mainly because of the time needed for setup and maintenance of the models, even though they were perceived as an improvement. (Ana M. Fernández-Sáez et al., 2016) concluded that smaller software projects within known domains don't need additional models and this could set the limit for the IA to a high-level overview of the system.

When focusing more on the domain knowledge and what would help maintainers in enhancing their impact analysis, the literature gives some insight. (Arisholm et al., 2006) pointed out that UML analysis models appear to uselessly overload participants when performing comprehension tasks and that design models appeared to help to achieve a better understanding of the systems. This aligned with findings in the literature, confirming analysis models are created to capture a domain, represent a set of requirements, understand a problem and its boundaries, while design models can be used to structure source code and capture design artifacts that do not directly emerge from requirements (Scanniello et al., 2018). This brings the question back to knowledge management theory. According to (Ding et al., 2014) knowledge can be classified as 'tacit' or 'explicit' knowledge. Explicit knowledge is the knowledge organized in certain form like a model or document, where tacit knowledge resides in people's head and is not easily visible and expressible.

Concerning the question whether the level of experience of the maintainers matter, (Ji-Ye Mao, 2000) sought out the difference between novice and expert professionals. By studying the effects of users' domain expertise against knowledgebase systems, they noticed that novices requested significantly more explanations than experts. This due to novices often fail to identify their own knowledge deficit and frequently miss contradictions and inconsistencies. The failure to identify deficits was founded in the lack in ability for asking questions, since *"questions arise from knowledge rather than ignorance"* (Ji-Ye Mao, 2000). Questions asked by novices are also typically shallow and address only the content and interpretation of explicit material, rather than high-level questions that involve inferences, applications, and synthesis. Experts in a given domain were thought to have a greater amount of declarative knowledge than novices, and able to access information more easily and more rapidly. So another factor to be aware of, according to (Ji-Ye Mao, 2000), is the domain-specific knowledge when it comes to reading-comprehension. This is also supported by (Garousi et al., 2015), who identified that the information stored in developers' minds is formed as domain-specific knowledge and has critical impact on understanding the documents. Another quote (not entirely in context), is that "*learning is inhibited by lack of time and working is inhibited by lack of knowledge"* (Ji-Ye Mao, 2000). To put this into perspective, the paper recognized that people are more concerned about getting their tasks done than gaining new knowledge and tend to avoid learning if they can muddle through.

On learning, (Gemino & Wand, 2005) stated that no learning occurs where comprehension and problem-solving is low. Fragmented learning occurs where comprehension is high, but problem-solving is low. Such results indicate material was received, but not integrated with prior knowledge. This would suggest memorization, rather than meaningful learning occurred. Meaningful learning occurs when both comprehension and problem-solving are high. High problem-solving indicate information is integrated into long-term knowledge and a high level of understanding of the presented material.

If the use of a UML model lowers the cost of learning and speed up the process of gaining domain knowledge, this could still be a win. Learning would also include the use of easy to read conceptual models. A general recommendation from (Ana M. Fernández-Sáez et al., 2015) is to keep UML diagrams up to date, also in order to improve maintainers' performances, thus learning by doing. The way an expert versus a novice looks at a UML model should be considered within our own experiment. This is similar in the sense that the 'domain knowledge' and 'experience' of the members should be taken into account, when estimating the efforts in development and maintenance (Sudhaman & Thangavel, 2015).

A matter of concern is the validity in some of the found research experiments. These are often questioned in regard to generalization, limited size and upfront domain knowledge, for example:

- (Garousi et al., 2015) gave clear insights on the usage of various types of documentation for software creation and maintenance and concluded that source code is most frequently used during maintenance in favor of other forms of documentation. But generalization of their outcome to other fields is not taken for granted, since they conducted their experiment one specific company specialized in embedded software.
- (Ana M. Fernández-Sáez et al., 2016) sought the relevance on the level of detail in UML diagrams, but partially limited the validity of their experiment because of the small size of the project and the upfront domain knowledge of the student participants.
- (Ji-Ye Mao, 2000) contemplated on the difference between novice and domain experts for the implementation of knowledge base systems. Where the novices tend to analyze problems in terms of surface features, experts are more able to identify underlying problem structures and appear to use simpler, domain-specific recognition procedures when faced with a problem. But they were concerned about the limited verbal responses from the experts and found a contraction with another study.
- (Arisholm et al., 2006) investigated the cost effectiveness of UML documentation on software maintenance and learned that for complex tasks and past a certain learning curve, the availability of UML documentation may result in significant improvements. Considering the time needed for updating, there did not seem to be any saving of time, especially for simpler tasks. But with their controlled experiment, using students, mentioned that they might even have underestimated (or overestimated?) the actual benefits of using UML documentation.

2.3. Hypotheses formulation

These matters are food for thought when researching the use of a conceptual model in areas where domain knowledge is high. Although complexity might be high for a novice, they would be quite simple for an expert.

Since the general theme is that conceptual models will aid in knowledge transfer, because viewing the model combined with textual information will require less effort on getting the whole picture than deriving this cognitive model using only the textual form of documentation. This brings forth the following hypothesis.

H_1 The domain knowledge will be higher between the groups of software engineers using a UML model compared to the group that only uses text documentation.

Since it is difficult to test domain knowledge, we tested this by measuring tacit knowledge, which is derived from the explicit knowledge. According to (Gemino & Wand, 2005) the knowledge transfer could be done using following tests; surface understanding can be measured by multiple-choice questions and deep understanding can be measured by problem solving questions. By following up with a Cloze test the deeper subtleties and meaning could be assessed for real deep understanding of the domain (Gemino & Wand, 2005; Kleijn, Pander Maat, & Sanders, 2019). So, a number of sub hypotheses are derived.

$H_{1A.0}$ Multiple-choice scores will be higher for the groups of software engineers using a UML model compared to the group that only uses text documentation.
$H_{1B.0}$ Problem-solving scores will be higher for the groups of software engineers using a UML model compared to the group that only uses text documentation.
$H_{1C.0}$ Cloze test scores will be higher for the groups of software engineers using a UML model compared to the group that only uses text documentation.

Whether the answers are true or false, the difference in domain knowledge on the maintenance aspect of known systems is yet to be answered. Therefore, an additional hypothesis was formulated.

H_2 There is no significant difference in domain knowledge between the use of a UML model or text only documentation when maintaining a system by system engineers with prior domain knowledge.

And followed with similar sub hypotheses.
$H_{2A.0}$ Multiple-choice scores will not be significantly higher between the use of a UML model or text only documentation when maintaining a system by software engineers with prior domain knowledge.
$H_{2B.0}$ Problem-solving scores will not be significantly higher between the use of a UML model or text only documentation when maintaining a system by software engineers with prior domain knowledge.
$H_{2C.0}$ Cloze test scores will not be significantly higher in domain knowledge between the use of a UML model or text only documentation when maintaining a system by software engineers with prior domain knowledge.

Following up on the study of (Ji-Ye Mao, 2000), there is the question if expert participants achieve better scores that novices. (Ji-Ye Mao, 2000) concluded that experts (in a given domain) are able to access information more easily and more rapidly when compared to novices. This is concluded by the final hypothesis where we challenge if there is a difference on expert scores on maintenance compared to a novice, with or without the use of a UML model. In this hypothesis the problem-solving scores and efficiency are used to address the domain knowledge. The level of expertise was dependent of the job level of the subject.

$H_{3.0}$ There is a significant difference in problem solving scores and efficiency, when maintaining a system on the use of a UML model or text only documentation, while controlling the level of expertise

These hypotheses are all null hypotheses and the goal of the statistical analysis will be to reject these and possibly accept the alternative.

3. Method

3.1. Strategy

The research objective was to determine if knowledge models contribute in increased understanding of a system. This, with respect to maintenance and the aim of improved problem-solving. The selected design is a between-subjects experiment, comparable to the research experiments done by (Arisholm et al., 2006) and (Gemino & Wand, 2005) where there are two groups and a difference in outcome is expected, when using a conceptual modal or not. With a between-subjects experiment, each subject receives only one treatment. A within-subjects design (or repeated-measures), where each subject receive all the treatments, is not selected because of the learning effect (Graziano & Raulin, 2004).

Creating and performing a single perfect and robust experiment is near to impossible. So, in order to rule out the threats to validity in regard to a single experiment and to obtain more data from different types of subjects the choice was made to conduct a number of replications, thus conducting a Families of Experiments.

Families of Experiments replicate individual experiments, and allow researchers to answer questions that cannot be singled out in a single experiment. This also allows to generalize the findings across similar studies (Basili, Shull, & Lanubile, 1999). A replication of an experiment is defined either as closely as possible following the original experiment or with deliberate changes to one or more parameters, named a close replication. A replication is called conceptual when the question remains the same, but the experimental procedure is different (Shull, Carver, Vegas, & Juristo, 2008). Next to that there can be internal or external replications. With internal replications the research is conducted by the same researchers, where external replications might be biased by the experimenters (Gravino et al., 2015). The Families of Experiments in this report consist of an original experiment and two internal close replications with the covariance in experience of the participants.

Using non-probability sampling to determine the correct sample size and given that the group is homogeneous, a minimal sample sizes of 4-12 participants per experiment is needed (Saunders, Lewis, & Thornhill, 2009). The participants were selected to have experience in either software development or maintenance tasks and were randomly mixed within each experiment.

We had an accessible population of 5 groups of 8 participants each, but because of limited time and planning only 3 groups were used. The participants and their companies volunteered in participating in our experiment, during regular office hours.

The following groups participated in the study:

- 8 participants, professionals from R&D software department at Mendix. (English)
- 8 participants, starter level software engineers at Educom. (Dutch)
- 8 participants, starter and professional web developers at Beter Bed. (Dutch)

Backup groups:

- 8 participants, professional Functional Application Managers SAP ERP at Beter Bed (Dutch)
- 8 participants, professional SAP ERP consultants at Ctac (Dutch)

The possible drawback of not using the 2 backup groups is that these groups consist of experienced software maintainers and could make the findings of this research even more compelling. Using more groups would probably also improve statistical significance.

The findings from the literature (Arisholm et al., 2006; Gemino & Wand, 2005) suggested that using models have impact on the domain knowledge and the transfer of this knowledge to the subjects. It was also recommended by (Ana M. Fernández-Sáez et al., 2015) to use the UML diagrams created during the design phase for software maintenance, because of the improved understanding of the system, compared to reverse engineered diagrams.

In order to test the transfer of domain knowledge in both an initial development as a maintenance situation, the experiment was divided in two parts. The first part provided the participants information on a software system, using written information and with or without the use of a UML model. This first part would also set the basic domain knowledge, and fit the need of the research question to test if the use of a model in a maintenance situation where system knowledge is available.

The questions in the tests consisted questions which are either in the text, the model, both or not presented at all. These tests challenged the subjects to fill in the gaps and allowed us to measure the shallow understanding (Gemino & Wand, 2005). This was combined with problem-solving questions and Cloze tests, where deep understanding was measured and in which experience would also be a significant factor (Ding et al., 2014; Greene, 2001; Kleijn et al., 2019).

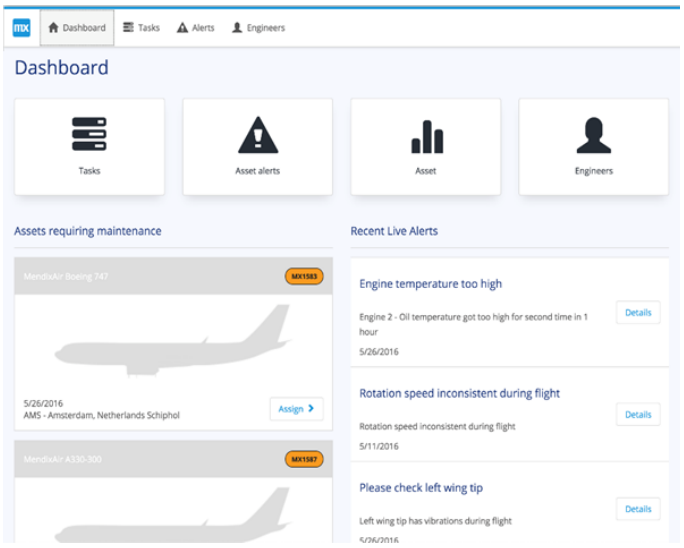
3.2. Research setup

In this Families of Experiments, each experiment had two parts, the first with the intention of providing the participants information on the system and testing the knowledge transfer using written text and with or without the use of a UML model. This with the focus whether a conceptual model aids in knowledge transfer, to be used in the initial design phase of the software lifecycle within Agile teams. The presence of a UML model will be the independent variable, also called the ‘main factor’. This is a nominal variable with two values (treatments): present or absent.

We selected a between-subjects balanced design in which each group was evenly split (Ana M. Fernández-Sáez et al., 2015) and decided to use this design rather than a within-subjects design because of time constraints and to avoid learning effects. The expected difference between the subjects was on individuals with an upfront higher domain knowledge and on those with a more experienced level in software. These threats to validity of a between-subjects design were taken into account, by randomly selecting the two groups and using a small preliminary questionnaire. The pre-questionnaire consisted of questions related to the subjects’ skills, educational level and experience and a set of questions in regard to the subjects’ knowledge of UML. Questions in regard to ease of use of the UML knowledge and readability were also added as post-questions after each test and the outcomes were used as dependent variables.

To ensure reproducibility and validity across the different groups, we carefully prepared the project and provided the information in a fixed format in order to minimize the influence of the researchers. In order to avoid threats to validity and too large differences between participants pre-knowledge, a topic was selected that was not part of the domain of the companies.

A project was constructed for this experiment, consisting an airplane maintenance application which shows alerts and maintenance tasks to a planner and engineers on airplane at a local airport. For the maintenance tasks, assets and a group of engineers are needed. Some additional business requirements were added to the text and model, such as amount of time needed, quality checks, etc. The full text of the project can be found in Appendix A. Case Text - ACME. See Figure 3 for a mockup of this application. The constructed conceptual model is shown in Appendix C. UML model.



Recent Live Alerts

5/26/2016
AMS - Amsterdam, Netherlands Schiphol

Engine temperature too high
Engine 2 - Oil temperature got too high for second time in 1 hour
5/26/2016

Rotation speed inconsistent during flight
Rotation speed inconsistent during flight
5/11/2016

Please check left wing tip
Left wing tip has vibrations during flight
5/26/2016

Figure 3 Mockup airplane maintenance system

The subjects in each group received the same information and questions were asked in the same order. Since the groups had a different native language, the information and questions were written and modelled in English and Dutch. The difference in language was not regarded as a thread to validity, regarding the large amount of papers read on Families of Experiments in multiple countries and languages.

The time given for reading the provided information was set to 15 minutes. This time was set in a dry run to be just enough to read and comprehend the given data. In order to stimulate the subjects to take notes, pens and markers were provided. All the written data was handed in before taking the quiz. Eliminating the models and notes was important for these tasks because it ensured that the only information available to the participant, is the cognitive model developed by reading the text and/or viewing the conceptual model. This raised the question on the impact of the UML model to the cognitive model, see Figure 4.

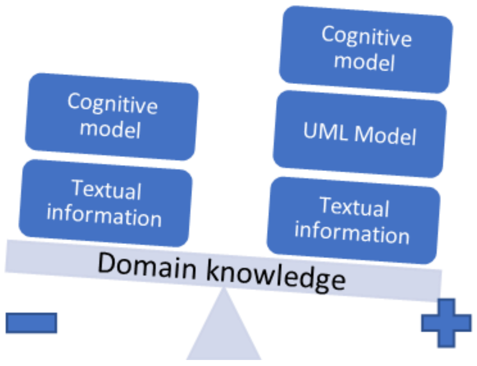


Figure 4 Impact of UML on Cognitive model?

The tests were conducted with the use of an online form specially created for this experiment. This automation reduces experimenter-participant contact and allowed to record time to answer the complete session and each individual question. In order to avoid stress no hard deadline was given, solely an indication that the first test should also take about 15 minutes.

The multiple-choice questions in the first part of the experiment were either in the text, the model, both or not presented at all. These factors were selected for evaluating the surface understanding, but also at a slightly deeper level for those not presented. Some problem-solving questions were also not directly solvable by the provided material, but closely related to measure the implicit transfer of domain knowledge. The order of the questions was setup as a follow up of earlier questions and could reveal critical information on those questions. It was therefore not possible to go back and change any answer.

At the end of the first test the participants were given the correct answers, with the intent of having all participants within the group on an equal level of system knowledge. At that point the subjects could ask brief questions, without sharing the actual model between the groups. A short coffee-break was given to avoid fatigue that could influence the results of the second part of the experiment. A threat to validity was that the subjects could discuss the model during this coffee-break, so the researchers were setup to listen in on the conversations and halt these if the subject of the model arise.

For the second part new textual information with maintenance and change requests was provided to the participants, along with their earlier written personal notes and information. The setup of the group of participants remained exactly the same. The time given to get acquainted with the new information and renew the basic information, was again set at 15 minutes.

The setup of the research questions was similar to the first test, using multiple-choice, problem solving questions and a Cloze test. The given time to answer the questions was set to 20 minutes, because more problem-solving questions were requested. The application knowledge gained in the first part was tested by asking specific control questions within the multiple-choice part. The multiple-choice questions were therefore categorized as either ‘basic’ or ‘new’ information.

The problem-solving questions were setup in such a way that it measures the level of understanding of the requested change with focus on ‘retention’ and ‘transfer’. Retention is defined as the comprehension of material being presented. Transfer, or problem solving, is the ability to use knowledge gained from the material to solve related problems which are not directly answerable from this material (Gemino & Wand, 2005). Prior logistic domain knowledge and expertise of the maintainer will be a dependent variable for our research, since other studies like (Ana M. Fernández-Sáez et al., 2016) and (Gravino et al., 2015) have questioned the validity of their experiments on this.

Any validity with respect to learning effect on the similar type and form of the questionnaire in the second part is neglectable.

Overview of the experiment in time is shown below in Figure 5.

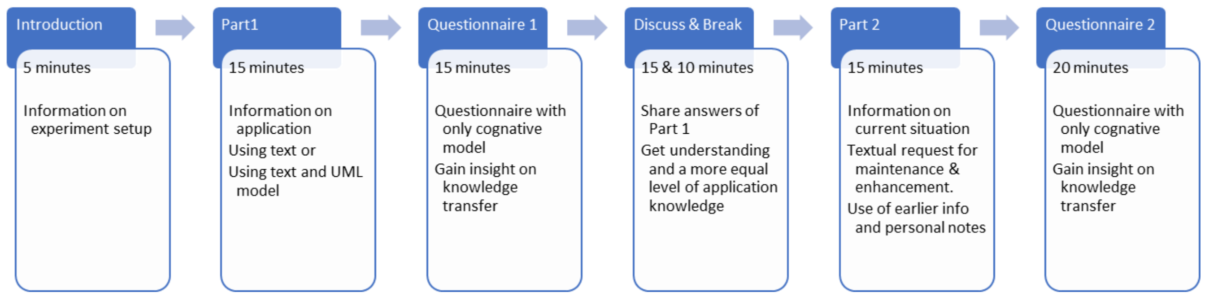

Introduction
5 minutes
Information on experiment setup
Part1
15 minutes
Information on application
Using text or
Using text and UML model
Questionnaire 1
15 minutes
Questionnaire with only cognative model
Gain insight on knowledge transfer
Discuss & Break
15 & 10 minutes
Share answers of Part 1
Get understanding and a more equal level of application knowledge
Part 2
15 minutes
Information on current situation
Textual request for maintenance & enhancement.
Use of earlier info and personal notes
Questionnaire 2
20 minutes
Questionnaire with only cognative model
Gain insight on knowledge transfer

Figure 5 Overview of the experiment in time

3.3. Variable selection, grading and analysis

Quantitative Analysis was done with the use of the independent variable, the use of UML model, and two treatments: UML, no-UML. A number of dependent variables are described in this section in order to measure the transfer of knowledge and variables to overcome threats to validity. The following variables are directly or indirectly obtained using the following type of tests:

- Multiple-choice questions for shallow understanding.
  Example question
  Question: The only way to resolve an alert is to complete the procedure attached to it.

Answer: Yes, see relation "resolves" from Procedure -> Alert and explanation in text.

- Problem solving for deep understanding and knowledge transfer
  Example question
  Question: The Boeing Max 737 has caused multiple crashes by similar malfunctions of a sensor. Regulation entity FAA decided to ground this type of plane. How would you report such a malfunction in our maintenance system?
  Possible answer: By reporting this malfunction manually, for all aircraft of this type.
- Cloze test for deep understanding and measuring more subtle differences of domain knowledge.
  Example question
  Question / Answer: If a *part* $^{C2.1}$ reaches a *minimum* $^{C2.2}$ stock level in the *warehouse* $^{C2.3}$, an *order* $^{C2.4}$ will be placed automatically at the *supplier* $^{C2.5}$ by the *unknown / system* $^{C2.6}$. When ordering parts just-in-time, the *delivery time* $^{C2.7}$ is most important

The complete list of questions is available in Appendix B.

For grading an acceptable and reliable score, all the answers were compared to the answer model and were graded independently by the two researchers. To indicate the degree of agreement in grading, the score of the acceptable solutions a Pearson correlation measured the strength of relationship. Also, a dependent t-test was used in order to establish if the means between the grades differ significantly. A Cohen's Kappa was performed on each question where the difference from the t-test was too high. With a Cohen's Kappa of 0,75 or above for each question, the scores given by each of the two raters can be used.

Based on the origin of the information (text / model / both / none) the multiple-choice questions of part 1 were computed into 4 different variables in percentage of correctness. The multiple-choice questions of part 2 were computed into 2 variables, based on whether the information was given in the first part or as new information: basic / new. The new dependent variables are in percentage of correct answers and will be used to measure surface understanding.

For the problem-solving questions and the Cloze test, all the individual scores along with the sum of these were used as dependent variables for analyzing deep understanding. The time components as a whole and for each problem-solving question were also nominated to derive the effectiveness, by dividing the correct answers by the time. A higher value of this ratio reflects better efficiency.

The other dependent variables were participant related factors, that might influence the outcome of the experiment. As a part of the questionnaire a short pre- and post-experiment survey was done on following factors:
Pre-test questions

- Educational level
- Position
- Experience in software (working years)
- Maintenance / software creation
- Experience level in system modeling (7-point Likert scale)
- Experience level in UML modeling (7-point Likert scale)

Post-test questions (all as 7-point Likert ordinal scale variable)

- Perceived ease of readability of the text
- Perceived ease of interpretation of the UML model
- Perceived ease of answering the questions

Logistics domain knowledge was added to the group of BeterBed as pre-knowledge, because of the retail activities that they are in a larger sense aware of than the rest of the subjects. From the experience in years and position, the dependent Job level variable was created with an ordinal scale, with following scores: 1: Student, 2:Entry Level (< 6 years' experience), 3:Experienced (Non-Manager), 4:Manager.

The level of significance for the hypotheses was set to 0,05. A likelihood ratio Chi-Square test was used to test the difference in the proportion of subjects with correct solutions for each variable, with the objective to test the difference between surface and deep understanding.

For testing the hypotheses, the selection of statistics was done via an online test selector (Statistics, 2018). The hypothesis $\mathbf{H_{1A.0}}$ and $\mathbf{H_{2A.0}}$ are tested with Hotelling's T^2 One-Way MANOVA because of the 2 groups on the independent variable and the joint multiple-choice questions. These scores are in percentages and give a ratio scale. In order to initially test whether the difference between the groups is significant, an independent-samples t-test was performed on these scores.

The hypotheses $\mathbf{H_{1B.0}}$, $\mathbf{H_{1C.0}}$, $\mathbf{H_{2B.0}}$ and $\mathbf{H_{2C.0}}$ have scale variables on the problem solving and Cloze tests, summing up the correct answers. These are tested with one-way ANOVA.

For $\mathbf{H_{3.0}}$ the choice for statistics could be two-way ANOVA, if the job level is taken as an independent variable and to check if an interaction effect exists. A one-way ANCOVA was not possible since the job level, taken as a covariate as expertise, is not a scale variable. A post hoc test or planned contrasts was also run to determine difference between the groups.

4. Research execution

The Families of Experiments was conducted initially at Mendix in Rotterdam, where 2 groups of 4 participants was split in a morning and afternoon session: starting with the group without the UML model. This setup was chosen so that we, the researchers, got more acquainted with the process and would learn from this experience and adjust if needed for the following experiments. The preparations on the informational part, the questions and computerized system were initially done in Dutch, which were afterwards translated into English. Some of the Cloze test questions couldn't be translated in the same sequence but were numbered in such a way that both languages had the same order for grading afterwards. With the short preparation time, one technical error was made which resulted in not measuring the time of the individual problem-solving questions.

Chocolate and candy were provided during the whole session, because of its positive effect on the participants brain health and improvement on cognitive performance during stressful conditions. The impact of coffee and chocolate on neuronal adenosine receptors and the brain is left out of this report, but it's recommended literature for any cold study-evading evening (Camandola et al., 2019).

During the introduction the participants were introduced with the setup of the experiment, a mockup of the system and some example questions. The subjects then got textual information and were given markers and pens and encouraged to take notes. 3 out of 4 took extensive notes and created models themselves. The difference could be checked on this 'creation of a model'. During the given 15 minutes, the available time was given every 5 minutes and at the last 2 minutes. At the end the notes and text were handed in and each labeled with their unique code, which was also used for their questionnaire. The system had a build in timer which was constantly visible for the subjects, to keep them aware of the time. Well within 15 minutes everybody was ready with the questionnaire and for 10 minutes all the correct answers were openly discussed. During this discussion the subjects were told that there was a discrepancy between the model and the text, such that they didn't have the answers to all the questions. Which was answered with relief, because they were a bit frustrated on these specific questions. There was also an ambiguity noted on one multiple choice question, on which we agreed that the answer should indeed be inversed.

During the 10-minute coffee break it was noted that nobody actually talked about the experiment, so no additional information was exchanged.

At the start of the second part of the first experiment, the subject's original notes and also the text of part two was provided. The participants were again seriously on their notetaking and also re-examining the text and notes of the first part. The same setup was conducted with the online questionnaire of part two and everyone finished well within the given 20 minutes. After finishing up we briefly talked with the participants on their experience on the topics and they gave the feedback that they liked the experiment but were less at ease with the logistics part of the questions of the second part. This is also additional domain knowledge, which is better known at Beter Bed.

In the afternoon the second group had undergone the same routine, with the difference that they were provided with a UML model. During the information intake, it was noticed by the researchers that there was less notetaking than during the first group. Another point of interest was that during the whole time the UML model was just briefly looked at, just until the last 30 seconds when all participants got a sudden interest of the model. A difference we noticed that this group took on average 4 minutes longer to answer the questionnaire, compared to the first group.

While giving the correct answers, in order to get the domain knowledge at an equal level, the same relief was felt on not having all the answers in those cases where information wasn't available at all. But it was also noticed that information from the UML model was missed by them, concerning multiplicity and the option of manual information entry. The second part of the experiment went comparably to the first group, with the difference that they took on average 2 minutes longer for answering all the questions.

We clearly noticed that during the coffee breaks, nobody talked about the experiment. Also, during the experiment everybody was really working independently. We therefore decided that during the next experiment we wouldn't split up the two groups, so that we could stay together during the experiment. Thus, making sure that the same way of information sharing was done during the two sessions and also no experimenter effects would bias the study.

The second experiment was done at Beter Bed. And where we were super prepared at the first experiment, we forgot to print the translated texts and UML models. Since it is a paperless office, we were forced to mail the subjects the information in tranches and gave them blanc paper to take notes. Because they were working on their computer, less notes were being taken in general, compared to the first experiment. At the end one person of the UML-group noticed that he received the model but was unaware of that fact. We therefore changed his position to the non-UML-group in the results.

The third experiment was done at Educom in Arnhem. As it turned out, the location should have been at Educom in Eindhoven. Luckily there were 8 students in Arnhem willingly to participate in the experiment. The rest of the experiment was comparable, and no information was exchanged between the groups concerning the models. By looking at their behavior, it was noticed that not all of the students were really interested in conducting the experiment.

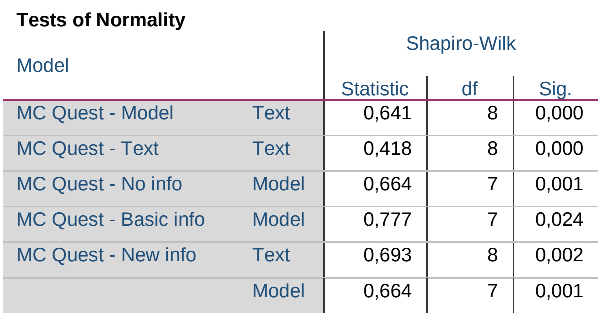
5. Results

5.1. Descriptive and rating

We first graded the results of the experiments and carried out descriptive statistics to find potential outliers and get more feeling for the data by checking the frequencies, normal distribution and skewness. The results of one of the subjects of the third group was completely removed, since the answers showed that he might have had an overdose of coffee and sugar. The means were tested with sample t-test, see Figure 13. These results had room for discussion. By using Cohen’s Kappa to check our two ratings the observations were reevaluated for questions where the Kappa was lower than 0,75. The final Pearson correlation between the graders showed that the overall levels were above 0.97, meaning there is a high degree of agreement. With the Kappa higher than 0,75 the results of either rater could be used, so obviously I used my own.

With the resulting data the tests for normality was done, revealing that with significance level of .01 all scores were normally distributed as assessed by calculating the z-scores for skewness and Kurtosis, with z-scores well below ± 2.58.

Since basic assumptions must be met for the parametric statistical tests, testing of approximately normally distribution was done on all the dependent variables for each group of the independent variable. As assessed by Shapiro-Wilk's test ($p < .05$) only see Figure 6, not all the scores are not normally distributed for multiple-choice questions on Model and on Text in both part 1 and 2.



Tests of Normality

Model		Shapiro-Wilk Statistic	df	Sig.
MC Quest - Model	Text	0,641	8	0,000
MC Quest - Text	Text	0,418	8	0,000
MC Quest - No info	Model	0,664	7	0,001
MC Quest - Basic info	Model	0,777	7	0,024
MC Quest - New info	Text	0,693	8	0,002
	Model	0,664	7	0,001

Figure 6 Shapiro-Wilk's Tests of Normality with p<.05

For all other dependent variables, the scores were normally distributed on both model and text. As assessed by visual inspection of the histograms, all scores were approximately normally distributed. All dependent variables scores were also normally distributed for both cases of the model, as assessed by visual inspection of Normal Q-Q Plots.

Before assessing the quantitative analytics, the categorical information from the pre- and post-questionnaire was checked using one-way ANOVA on model to get more insight on the participants. These scores were normally distributed throughout the subjects (Figure 16) and as assessed by visual inspection of the boxplots, there were no outliers apart from the post text question from the second part. Between the three groups there were no statistic significant differences. Assessing the level of expertise of the subjects on UML modelling, these are on the low side with a stated ‘Fair’ average (score 3 on 7-point Likert scale).

5.2. Shallow understanding - $H_{1A.0}$ | $H_{2A.0}$

In order to determine the effect of the usage of a model on surface understanding Hotelling's T^2 was run. Four measures of performance were assessed for **$H_{1A.0}$**: information given in Text/Model/Model and Text or nowhere. Data are expressed as mean ± standard deviation. Preliminary assumption checking revealed that data was not normally distributed, as assessed by Shapiro-Wilk test ($p > .05$)

and described in the previous section; there were some univariate but no multivariate outliers, as assessed respectively by the boxplot (Figure 19) and Mahalanobis distance ($p > .001$). There was only one linear relationship, as assessed by the scatterplot in the second part; no multicollinearity ($|r| < .9$); and there was homogeneity of variance-covariance matrices, as assessed by Box's M test ($p=.625$ for **$H_{1A.0}$**, $p = .193$ for **$H_{2A.0}$**).

It was shown that subjects that used:

1. a model score higher on questions with info in the model (M=33.3 ± 6.3 and T=25.0 ± 6.0)
2. text score higher on questions with info in the text (M=63.6 ± 8.6 and T=83.3 ± 8.2)
3. text score higher on questions with info in both (M=51.5 ± 8.5 and T=58.3 ± 8.1)
4. a model score higher on questions with no info (M=77.3 ± 8.9 and T=66.7 ± 8.6)
5. text score higher on questions with basic info (M=69.7 ± 6.4 and T=72.2 ± 6.2)
6. a model score higher on questions with new info (M=85.5 ± 5.0 and T=80.0 ± 4.7)

The differences between the use of a model and text was not statistically significant for **$H_{1A.0}$**, $F(4, 18) = 1.127$, $p < .375$; Wilks' $\Lambda = .800$; partial $\eta 2 = .200$ and not for **$H_{2A.0}$** , $F(2, 20) =.355$, $p < .705$; Wilks' $\Lambda = .966$; partial $\eta 2 = .034$.

The combined group means were not statistically significant different ($p > .05$). Therefore, we cannot reject the null hypotheses **$H_{1A.0}$** and **$H_{2A.0}$** nor we cannot accept alternative hypotheses. With these results we could only state that the use of a model did not have any significant effect on shallow understanding within our experiment.

A point of interest is the means of multiple-choice questions on subjects that only had info in the text. These means were high, compared to the subjects that used UML in combination with the text. The means of the other dependent variables are more closely together.

Basically, when you would take the time to initially create a model in consideration, one could say it is better to write a good text with all business rules than to model it in UML. Alternatively, it could be that the persons which used the text and modelled themselves also had better shallow understanding.

5.3. Deep understanding - $H_{1B.0}$ | $H_{1C.0}$ | $H_{2B.0}$ | $H_{2C.0}$

One-way ANOVA was run to determine if there are significant differences between the means of the groups on the use of the model or text only, in order to make observations on deep understanding. These are part of hypothesis **$H_{1B.0}$**, **$H_{1C.0}$**, **$H_{2B.0}$** and **$H_{2C.0}$**.With initially testing the assumptions, we noticed several outliers in part 1 on problem-solving and Cloze test 2 scores and in part 2 on problem-solving and Cloze test scores. Since we agreed not to shop on our data to get significant levels, the non-parametric Kruskal-Wallis H test was selected for further testing. Due to a version error in the second run of the experiment the data for Cloze test 2 was missing, the results are therefor left out of the analysis.

With the Kruskal-Wallis H test the differences in mean scores between the two groups of the Model was determined.

1. Median scores on problem-solving part 1 decreased from text (8.5) to model (8.0)
2. Median scores on Cloze test 1 part 1 increased from text (60%) to model (70%)
3. Median scores on problem-solving part 2 increased from text (6.5) to model (7.0)
4. Median scores on Cloze test part 2 decreased from text (68.2) to model (63.6)

Distributions of scores were similar, as assessed by visual inspection of the boxplots, see Figure 7.

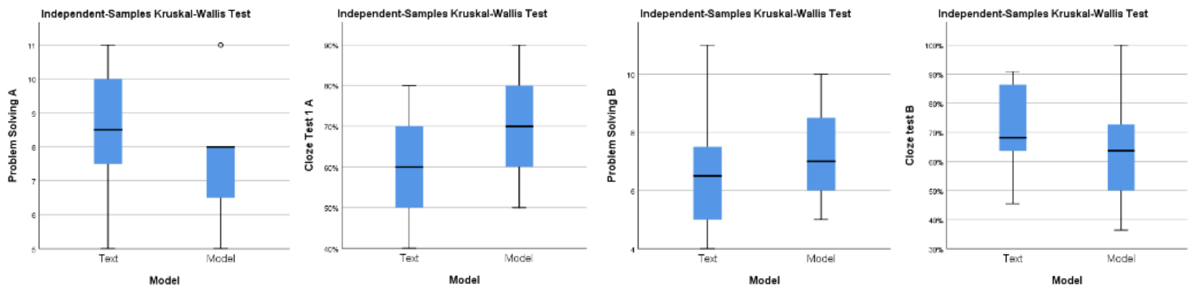


Figure 7 Kruskal-Wallis boxplots

But median scores were not statistically significantly different between groups for none of the variables, *1.* $\chi^2(1) = 1.348$, $p = .246$, *2.* $\chi^2(1) = 2.900$, $p = .089$, *3.* $\chi^2(1) = .609$, $p = .435$, *4.* $\chi^2(1) = 2.018$, $p = .155$.

This means that we cannot statistically reject the null hypotheses **$H_{1B.0}$**,s **$H_{1C.0}$** ,d **$H_{2B.0}$** and **$H_{2C.0}$** nor accept alternative hypotheses. Also for deep understanding it seems that in our case there is no real significance in the use of the model.

5.4. Problem-solving and efficiency - $H_{3.0}$

A two-way ANOVA was conducted to examine the effects of problem-solving and efficiency and job level on the use of a UML model for maintenance. Residual analysis was performed to test for the assumptions of the two-way ANOVA. Outliers were assessed by inspection of boxplots, normality was assessed using Shapiro-Wilk's normality test for each cell of the design and homogeneity of variances was assessed by Levene's test. Apart from using text on a student job level, there were no outliers and also here we kept the data as-is. Residuals were normally distributed ($p > .05$), see Figure 20. There was homogeneity of variances, as assessed by Levene's test for equality of variances for both problem-solving and efficiency of part 2, $p = .347$ and $p= .220$ respectively.

There was no statistically significant effect between Model and Job level on problem-solving nor for efficiency with scores, $F(2, 16) = .599$, $p = .561$, partial $\eta^2 = .070$ and $F(2, 15) = .383$, $p = .694$, partial $\eta^2 = .087$ respectively. Therefore, an analysis of the main effect for education level was performed, with no statistically significant results.

This also means that we cannot statistically reject the null hypothesis **$H_{3.0}$** nor accept an alternative hypothesis.

Looking at the individual hypotheses and parts of the experiment on the use of a UML model, we can conclude that the use of a model had no statistical effect on either shallow or deep knowledge within the context of our experiment.

The statistics were not significant, but merely looking at the plots you could be led to believe that having a model does have a positive effect. This comes more into effect when taking the job level into account. As seen in Figure 8, the novices seem to score better in getting acquainted with the system when using a model, where the more experienced subjects score better with only the use of the text. And most probably the models they created for their own, and thus creating a stronger conceptual model. It is however also visible in these plots that all the participants that used the model tend to take longer in answering the questions and thus have a lower efficiency.

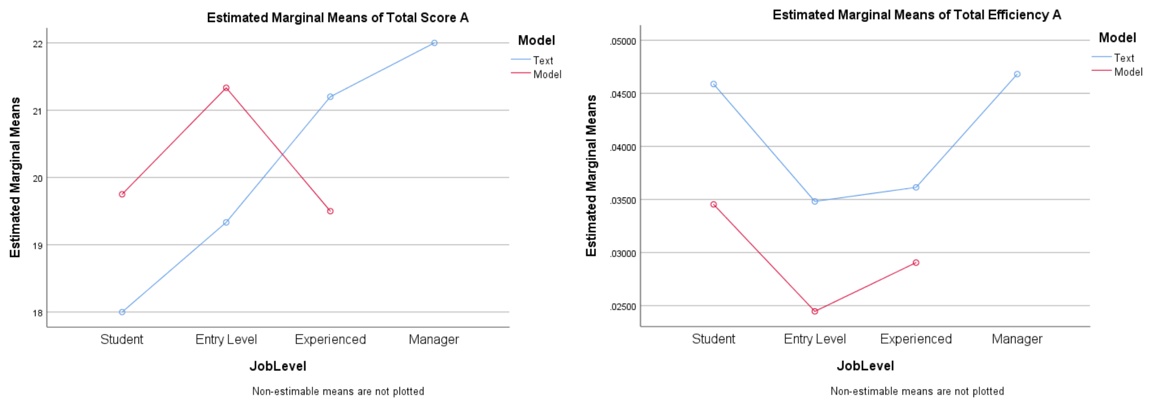


Figure 8 Estimated Means plots on Model and Job level on total score and efficiency part 1

When looking at the second part, where all subjects got a basic level of the application and more maintenance related questions were asked, the effect of the model seemed to be lower for the more novice subjects, see Figure 9. The model also seemed to have some effect on the more experienced participants. Since the students answered more quickly in this second part and no conclusive effect can be drawn, this leaded to my personal believe that experiments with students have less value than using professionals.

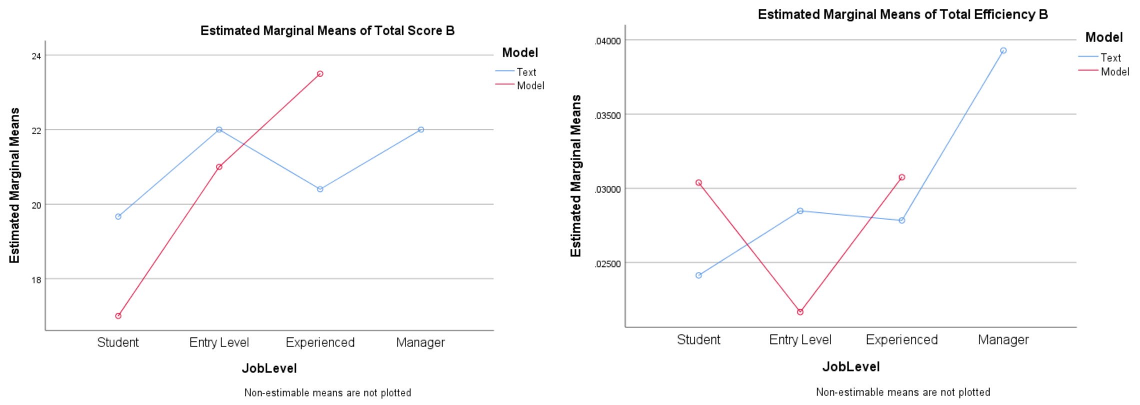
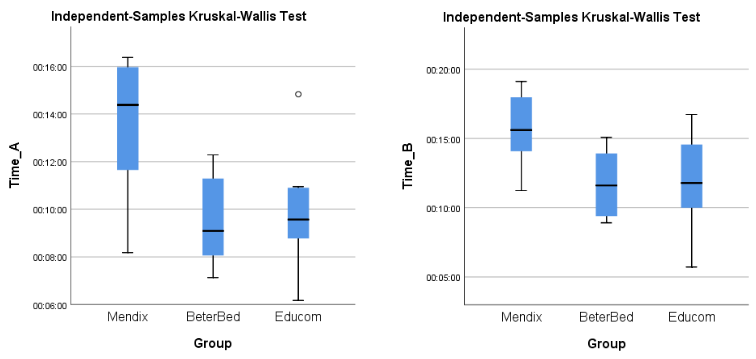
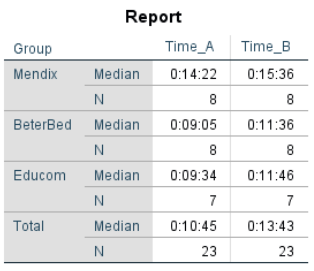


Figure 9 Estimated Means plots on Model and Job level on total score and efficiency part 2

5.5. Between-group comparisons

Violations of validity between groups was assessed with a Kruskal-Wallis H test, because the data was not normally distributed and outliers were detected. Assessing the time taken to conduct the questionnaires, differences were found between Medix and the following groups on both parts of the experiment, with respectively $\chi^2(2) = 5.866$, $p = .053$ and $\chi^2(2) = 6.745$, $p = .034$; see Figure 10. The reason for this difference could be, because the run at Mendix was the first test and the model-groups were split in a morning and afternoon session. The observers nor any other external influences were different from the two later sessions.





Report

Group		Time_A	Time_B
Mendix	Median	0:14:22	0:15:36
	N	8	8
BeterBed	Median	0:09:05	0:11:36
	N	8	8
Educom	Median	0:09:34	0:11:46
	N	7	7
Total	Median	0:10:45	0:13:43
	N	23	23

Figure 10 Time differences between groups

When looking at education and job levels between the groups there was also a significant difference, which was expected since we used students in the last group. But, looking at the boxplots in Figure 11 on education and job levels, the first group scored significantly higher. This could indicate a correlation between these levels and the time taken on a questionnaire, but no significant correlation was found.

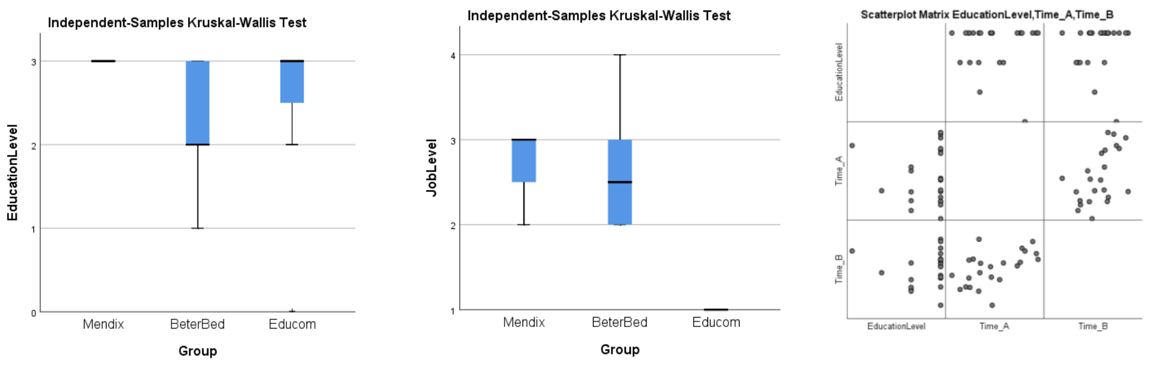


Figure 11 Education and job levels

As a follow up, the three groups are compared in combination with the model and as assessed with a Kruskal-Wallis H test on all used dependent variables. The only statistically significant difference between the groups was on the model on time and total efficiency in part 1 and total score of part 2 as seen in Figure 12. The reason for the time-part might be because of the effect the combined groups had on each other in the replications. The level of scoring was probably caused by a slight lack of interest by some of the students.

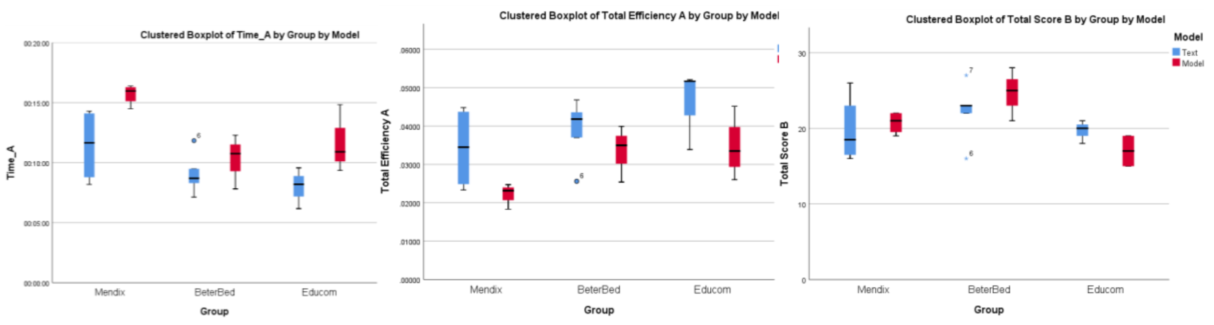


Figure 12 Boxplots between groups comparisons

6. Discussion on validity

Within the setting of the experiment, threats to validity were avoided as much as possible which lead to conducting a Family of Experiments between-subjects. The hypotheses were constructed in the theoretical context of the studied literature, meaning that no threat to construct validity is expected. A number of issues that may have threatened validity are discussed below.

- **Statistical validity**: The significance was probably affected by the select number of observations, since none of the results gave accuracy of the p-value and no statistically significant conclusions could be drawn from the data. This might have caused a Type II error, where we thought that no effect is measured. Additional replications are therefore required to confirm or reject the results.
- **External validity**: The results of these experiments were obtained in comparable settings with representatives in the field of software engineering. The number of maintainers was limited, which were available in the - not used - backup groups. In spite of this, we believed that this experiment could be considered appropriate to generalize after conducting more experiments. The participation by the subjects on the experiment was entirely voluntarily and apart from one very specific case, all subjects did their best on the tests. The speed at which some of the students finished the second part and the thus shown results are questionable, which might indicate that they had less interest in the exercise.
- **Internal validity**: The experiment was setup to control for violations on internal validity by using a Family of Experiments with a between-subjects approach. The selected groups were all involved in software engineering and divided randomly for the use of the model. The participants were tested using the same questionnaire system and the questions were asked in the same order. On the first run, the time on individual questioned wasn't measured due to a technical error, this limited the check on effectiveness. The material was translated In Dutch and English to avoid threats in comprehension. Some English textual errors were found during the first run, which rose some questions, but this had no influence on the test results. The second Cloze test in the first part was not correct in the second run, due to a wrong version of the translation. These questions were left out of the data to avoid threats to validity.

  To avoid experimenter effects, the influence of the researchers was limited by using written information, with only a small explanation of the questions of part 1. This was done by the same researcher in all 3 runs. In the second run, we forgot to print the textual information, and was therefore read from screen. We believed that this wouldn't pose a threat to validity. Grading was done by using multiple observers and running the statistics across the different groups all gave similar results. Violations on experimenter effects can thus be neglected.

  For one specific participant the data was removed due to nonsense reactions, but was expected not to cause any violation towards attrition.

  The experience on UML modelling, comprehension of the text, the ease use of the model and the questionnaires was questioned in the pre- and post-questions. The UML model wasn't 100% correct UML and it is difficult to insert in all the business rules. The pretests also showed that the knowledge of modelling wasn't overly available throughout all subjects. The text in the second part was initially thought to be easier to read for maintainers with logistical knowledge, but the post-questions on UML and questionnaires showed that the subjects were neutral in the ease of use on both parts, regardless of the logistical knowledge. One remarkable note can be made on multiple-choice question 6 of part 1, which nobody answered correctly. The information on this could only be found in the model but wasn't picked up. This could mean that the model wasn't clear enough, experience in UML modelling or the actual use of the model was limited.

7. Conclusion and recommendations

My personal work-problem on maintenance could be stated as: “There is a lack in documentation on our known ERP application concerning its systems and processes. With the request to reduce costs and the foresight of knowledge evaporation, what kind of documentation would be needed to:

a. Secure information concerning the current system and the processes?
b. Speed up impact analysis and maintenance?
c. Make onboarding of new employees easier and faster?
d. Let people look beyond the scope of their own daily work?
e. Help future projects to avoid knowledge gaps?

The focus of this thesis was set on the second and last of these problems and could simply be put as: “Does a conceptual model help expert maintainers to maintain a known system?”. This with the intent that hired external consultants can perform at their best, with a minimal amount of time needed.

The earlier findings from the literature (Arisholm et al., 2006; Gemino & Wand, 2005) suggested that using models have positive impact on the domain knowledge and the transfer of knowledge. It was recommended (Ana M. Fernández-Sáez et al., 2015; Scanniello et al., 2018) to use the UML diagrams created during the design phase, for software maintenance because of the improved understanding of the system. But when pressure and time are a constraint, documentation is less used and if the domain is known, the time taken to read UML diagrams is minimized (Ana M. Fernández-Sáez et al., 2016).

This was taken in mind with the setup of the between-subjects Family of Experiments, which was carried out with three groups using a text-only or text and UML model experiment. As noticed in the pre-questions, the expertise on modelling was limited in all groups, which posed a threat to internal validity. During the experiments we saw that a number of subjects who only received the text, started modelling themselves and tried to draw the relationships and fill in the blanks. All the subjects with the model at hand, took the model for granted and didn’t draw any models.

From the statistics of our experiment, the conclusion was drawn that there was no significant difference between the use of a UML model or text-only, when looking at the transfer of domain knowledge. Either when getting to know a system or maintaining the by then known system on both surface as on deep understanding. There was however a slight difference in time and effectiveness, with a tendency that the groups with UML models needed more time to answer the questions.

Concerning the difference between novice and expert maintainers there was also no statistical evidence that experts outperformed novices with the use of a UML model.

Since we did not clearly write down who did or did not do the actual modelling in the text-only groups, nor how much time was spend on the UML model in the other groups. It is recommended to take notes on this in follow-up research. It is also recommended to extent the experiment with the two backup groups to gain more knowledge on how expert maintainers will respond to the use of UML models. Further replication of this experiment is advised to be done with UML practitioners.

As questioned before, the above findings might indicate that creating a model, is more relevant than having a model. In the sense that it helps in creating a better cognitive model since the creation of conceptual models seems to be an important aspect of gathering domain knowledge. One could also opt for pre-documenting missing information and architecture recovery (Ding et al., 2014). If not used for maintenance, it could be for onboarding new employees and let novices learn modelling in order to become experts. It is thus recommended to investigate further on this topic.

Along with this we also questioned if UML models are clear enough in their use, since they don’t cover all aspects of the business rules. Other more formal models could better fit this need.

8. Reflection

The research was done with three groups of participants. During these runs we recognize that there were minor startup problems. Feedback from our professor was that researchers often remove the initial 3 tests, and state these as dry runs to create an even better experimental setup. The use of UML compared with the level of expertise gave doubt for generalization. Also one of the subjects asked me if the experiment was more a cognitive challenge and test in reading comprehension, than the use of a model versus text. The fact that this subject was in the model-group, made me question if used the model. In this I at least agreed with (Ji-Ye Mao, 2000) in that novice subjects with a lack of knowledge on UML modeling will muddle through without correct use of the model.

Also the question could be if UML is really the facto standard and suitable in this context. This is still to be answered and further research on this would be wise. Personally I would like to contact M. Chaudron on this at a later point in time.

The questions themselves could have been looked at more in-depth, to exactly pinpoint the differences between a model and text in relation to maintenance. A good test could also have been to let the subjects model the requested enhancements prior to answering additional questions.

Concerning the setup of the application for the experiment, knowledge was gained in the first part, but not to the extent of knowing all common options of the domain. The experiment did not test on real-life situations where domain experts know the extent of the application, rather than the appropriate business rules setup in the configuration and customization. A different setup with change requests, using application experts could be used in future research.

The thesis and Family of Experiments was setup in a very decent and pleasant way, especially given the time constraints and personal workload. It helped to do this as a group and later with the only remaining and fellow researcher. This gave combined insights, hard deadlines and the will to do a better job, when working closely together. Also the literature research gave more insight on knowledgebase systems, which I haven't used in-depth in this thesis. Paying heed upon documenting within my own organization, where most of the maintenance is done on the SAP ERP system, the need for documentation in knowledgebase systems is a necessity and a formal way of working is work in progress on which I would like to report later in time.

I am very happy and glad to have done the setup together with Jos, the advices from our professor and support of my loving family, friends and colleagues.

References

Abdullah, N., Honiden, S., Sharp, H., Nuseibeh, B., & Notkin, D. (2011, 2011). *Communication patterns of agile requirements engineering*.

Aljumaily, H., Cuadra, D., & Laefer, D. F. (2019). An empirical study to evaluate students' conceptual modeling skills using UML. *Computer Science Education, 29*(4), 407-427. doi:10.1080/08993408.2019.1642699

Arisholm, E., Briand, L. C., Hove, S. E., & Labiche, Y. (2006). The impact of UML documentation on software maintenance: an experimental evaluation. *IEEE Transactions on Software Engineering, 32*(6), 365-381. doi:10.1109/TSE.2006.59

Basili, V. R., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering, 25*(4), 456-473. doi:10.1109/32.799939

Beck, K. B., M.; Bennekum, A.v.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R.C.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D. (2001, 2001). Manifesto for Agile Software Development. Retrieved from http://www.agilemanifesto.org

Camandola, S., Camandola, S., Plick, N., Plick, N., Mattson, M. P., & Mattson, M. P. (2019). Impact of Coffee and Cacao Purine Metabolites on Neuroplasticity and Neurodegenerative Disease. *Neurochemical Research, 44*(1), 214-227. doi:10.1007/s11064-018-2492-0

de Souza, S. C., Anquetil, N., & de Oliveira, K. (2005). *A study of the documentation essential to software maintenance*.

de Souza, S. C. B., Anquetil, N., & de Oliveira, K. M. (2006). Which documentation for software maintenance? *Journal of the Brazilian Computer Society, 12*(3), 31-44. doi:10.1007/BF03194494

Díaz-Pace, J. A., Villavicencio, C., Schiaffino, S., Nicoletti, M., & Vázquez, H. (2016). Producing Just Enough Documentation: An Optimization Approach Applied to the Software Architecture Domain. *Journal on Data Semantics, 5*(1), 37-53. doi:10.1007/s13740-015-0053-0

Ding, W., Liang, P., Tang, A., & van Vliet, H. (2014). Knowledge-based approaches in software documentation: A systematic literature review. *Information and Software Technology, 56*(6), 545-567. doi:https://doi.org/10.1016/j.infsof.2014.01.008

Fernández-Sáez, A. M., Caivano, D., Genero, M., & Chaudron, M. R. V. (2015, 30 Sept.-2 Oct. 2015). *On the use of UML documentation in software maintenance: Results from a survey in industry.* Paper presented at the 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS).

Fernández-Sáez, A. M., Genero, M., Caivano, D., Chaudron, M. R. V., Institutionen för data- och, i., fakulteten, I. T., . . . Faculty, I. T. (2016). Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments. *Empirical Software Engineering, 21*(1), 212-259. doi:10.1007/s10664-014-9354-4

Fernández-Sáez, A. M., Genero, M., Chaudron, M. R. V., Caivano, D., Ramos, I., Institutionen för data- och, i., . . . Faculty, I. T. (2015). Are Forward Designed or Reverse-Engineered UML diagrams more helpful for code maintenance?: A family of experiments. *Information and Software Technology, 57*, 644-663. doi:10.1016/j.infsof.2014.05.014

Garousi, G., Garousi-Yusifoğlu, V., Ruhe, G., Zhi, J., Moussavi, M., & Smith, B. (2015). Usage and usefulness of technical software documentation: An industrial case study. *Information and Software Technology, 57*, 664-682. doi:10.1016/j.infsof.2014.08.003

Garousi, G., Garousi, V., Moussavi, M., Ruhe, G., & Smith, B. (2013, 2013). *Evaluating usage and quality of technical software documentation: an empirical study*.

Gemino, A., & Wand, Y. (2005). Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. *Data & Knowledge Engineering, 55*(3), 301-326. doi:10.1016/j.datak.2004.12.009

Gravino, C., Scanniello, G., & Tortora, G. (2015). Source-code comprehension tasks supported by UML design models: Results from a controlled experiment and a differentiated replication. *Journal of Visual Languages and Computing, 28*, 23-38. doi:10.1016/j.jvlc.2014.12.004

Graziano, A. M., & Raulin, M. L. (2004). *Research Methods: A Process of Inquiry* (5th edition ed.): Pearson Education Group.

Greene, B. (2001). Testing reading comprehension of theoretical discourse with cloze. *Journal of Research in Reading, 24*(1), 82-98. doi:10.1111/1467-9817.00134

Grossman, M., Aronson, J. E., & McCarthy, R. V. (2005). Does UML make the grade? Insights from the software development community. *Information and Software Technology, 47*(6), 383-397. doi:10.1016/j.infsof.2004.09.005

Ji-Ye Mao, I. B. (2000). The Use of Explanations in Knowledge-Based Systems: Cognitive Perspectives and a Process-Tracing Analysis. *Journal of Management Information Systems, 17*(2), 153-179. doi:10.1080/07421222.2000.11045646

Kajko-Mattsson, M. (2008). *Problems in agile trenches*. Paper presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, Kaiserslautern, Germany.

Kleijn, S., Pander Maat, H. L. W., & Sanders, T. J. M. (2019). Cloze testing for comprehension assessment : The HyTeC-cloze. *Language Testing, 36*(4), 553-572. doi:10.1177/0265532219840382

Monperrus, M., Eichberg, M., Tekes, E., & Mezini, M. (2012). What should developers be aware of? An empirical study on the directives of API documentation. *Empirical Software Engineering, 17*(6), 703-737. doi:10.1007/s10664-011-9186-4

Paul C, C., Felix, B., Len, B., David, G., James, I., Reed, L., . . . Judith, S. (2004). A Practical Method for Documenting Software Architectures. doi:10.1184/R1/6591197.v1

Saunders, M., Lewis, P., & Thornhill, A. (2009). Research methods for business students. In P. E. Limited (Ed.), (5th edition ed., pp. 279).

Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J. A., Tortora, G., Risi, M., & Dodero, G. (2018). Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments. *Empirical Software Engineering, 23*(5), 2695-2733. doi:10.1007/s10664-017-9591-4

Shull, F. J., Carver, J. C., Vegas, S., & Juristo, N. (2008). The role of replications in Empirical Software Engineering. *Empirical Software Engineering, 13*(2), 211-218. doi:10.1007/s10664-008-9060-1

Statistics, L. (2018). Statistical tutorials and software guides. Retrieved from https://statistics.laerd.com/

Stettina, C., & Heijstek, W. (2011, 2011). *Necessary and neglected?: an empirical study of internal documentation in agile software development teams*.

Sudhaman, P., & Thangavel, C. (2015). Efficiency analysis of ERP projects—software quality perspective. *International Journal of Project Management, 33*(4), 961-970. doi:https://doi.org/10.1016/j.ijproman.2014.10.011

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Ali Babar, M. (2010). A comparative study of architecture knowledge management tools. *The Journal of Systems & Software, 83*(3), 352-370. doi:10.1016/j.jss.2009.08.032

Zhi, J., Garousi-Yusifoğlu, V., Sun, B., Garousi, G., Shahnewaz, S., & Ruhe, G. (2015). Cost, benefits and quality of software development documentation: A systematic mapping. *Journal of Systems and Software, 99*, 175-198. doi:https://doi.org/10.1016/j.jss.2014.09.042

Appendix A. Case Text - ACME

Part 1 - Text

Airplane Controlled Maintenance Enterprise (Acme) is a company which is primarily focused on the maintenance of aircrafts. It's located at Schiphol, where they have multiple hangars from which they performance maintenance. From every hangar only one maintenance action can be performed at the same time. Every airplane that returns to its base station Schiphol will deliver input to the maintenance system.

To easily keep track of the current state of maintenance activities a mockup created of an app that helps managing this. It contains views with information on tasks performed by engineers, alerts, engineers and the aircrafts. The main dashboard shows an overview of items of the highest importance. It contains a list of recent malfunctions, with a date and some short descriptions. These malfunctions are based on the alerts that are stored by the planes during flight; these are automatically uploaded when the plane lands at Schiphol. Based on the severity of the alert the maintenance will be planned with a certain urgency, the planner will take care about handling this and ensures that a procedure is started. This procedure consists of multiple tasks, assigned to an engineer.

On the dashboard there is also an overview of all assets that require maintenance. This overview contains the plane's manufacturer, type, planned maintenance and its current location. The regularly scheduled maintenance depends on a couple of variables, such as flight hours, age and type of the plane. This overview can be used to assign maintenance tasks to an engineer.

One of the top priorities in aviation industry is security. This results in a lot of rules and regulation concerning maintenance and thus results in standardized procedures and tasks for all kinds of maintenance. As part of a task, parts can be replaced, for each of which a specific manual should be available. To enable the planner to plan ahead, every task and procedure have an estimation of the time required to perform it.

Engineers work according to standardized procedures, for both the regular maintenance work as well as for resolving malfunctions indicated by alerts. After all procedures and tasks are finalized an engineer should mark an alert as resolved. An engineer should be qualified to perform certain tasks, which is regulated by certification. Only certified engineers can be planned to perform tasks that have requirements on this. An engineer should report on working hours and note down his remarks. This is kept in the maintenance logs of the asset; all of the performed procedures form its maintenance history.

An aircraft remains grounded as long as there are procedures being performed, or alerts haven't been resolved yet. A plane can be parked in the hangar, but only one plane at the time is under maintenance. If a more complicated procedure needs to be performed a plane can be moved to a different hangar, as some of the required equipment is not available in every maintenance location. All alerts have to be resolved by an engineer before the procedure is resolved. An alert can be marked as resolved by performing a procedure that resolves the issue right away or based on testing in the procedure decide to plan the maintenance later as part of the regular schedule. Based on this information the planner decides to put the airplane in or to take it out of operation. The hangar is available again for another maintenance procedure as soon as this has been done. If a plane reaches the maximum number of flight hours, it will be taken out of operation by the planner.

For this you can assume that there's an unlimited amount of parts available in the local warehouse, which can be used without considering costs or whether it's in stock. The planner will make sure that the required materials and engineers will be in the right hangar on time.

Part 2 - Text

Business is good for the Acme corporation and they want to upgrade their current systems. Mostly because of the higher demand on maintenance, they need to scale up the logistics part, but without increasing too much in costs. To take the high engineering personnel costs into account before starting a repair, the total costs should be estimated more reliably.

Logistic errors are occurring more often, e.g. not all required materials are ordered in time by the planner, parts are lost or used in another maintenance location. This adds up to the maintenance time, resulting in aircrafts being grounded longer than expected. Other reasons for delays and high costs are sloppy engineers who don't return parts after completing their maintenance tasks. This results in superfluous stock of parts. It's also common that for some popular types of aircraft the maintenance is delayed because some parts have a high turnover rate a long delivery time.

The local warehouse doesn't have enough space to store all spare parts. The warehouse is therefore provided with exact stock locations which contain a certain minimal amount of stock for a certain part. Because not every part requires a high amount of stock, the system should order the right amount of parts for certain just-in-time from their supplier.

To provide a more detailed cost overview for regular, scheduled maintenance, a specification of the estimated cost, parts, maintenance location and engineers should be supplied to the planner. This enables the customer to determine if it's economically viable to continue with the maintenance or that the plane should be put out of service by the planner

This requires the following changes to the system

- The order process should be automated: it should take minimal stock amounts and rotation speed into account. Especially for products with a long delivery time.
- Logistics between warehouse and hangars: parts should be automatically reserved at the warehouse based on the planning of tasks. This enables the planner to get an overview if the planned maintenance date is feasible.
- Costs: a new overview should be added that displays the costs of the used parts, engineers and maintenance location for every maintenance service being performed.

If there's no supply the planner will contact the warehouse to order materials in time, this happens by phone and is not logged in the system. He can also request to change the minimal amount of stock for supplies or to move the maintenance to a later date.

Appendix B. Questions Case - ACME

<u>Part 1 – Questions + answers</u>

Comprehension question

1. Maintenance tasks on an aircraft are performed by certified engineers, can an engineer be assigned to multiple maintenance locations per timeslot? **[Model]**
   No, see relation assigned_to from Engineer to Maintenance_Location in the knowledge model.
2. If an aircraft touches down on Schiphol and sends over its notice to the maintenance system, it will <u>always</u> result in starting a procedure. **[None]**
   No, not in the model or text.
3. The parts that are required for a maintenance task are determined while performing the task. **[Text]**
   No, tasks and parts are performed based on guidelines from a manual.
4. There are always multiple engineers involved for performing a maintenance task. **[Model]**
   Yes, see relation quality_check from Task -> Engineer in the knowledge model.
5. An alert with a status higher then warning leads to a single procedure, which consists of multiple tasks. **[Text]**
   No, an alert can consist of multiple procedures. See multiplicity Alert -> Procedure in the knowledge model.
6. All alerts are inserted just automatically to the maintenance system **[Model]**
   No, there's also a manual flow which facilitates pilots to pass information about a malfunction. See Alert – Source in knowledge model
7. The estimated time to perform a difficult procedure is always known before start. **[Model & Text]**
   Yes, check the Procedure's – Estimated_Time property in knowledge model and explanation in text.
8. Work can be performed in multiple maintenance locations during a maintenance procedure of an aircraft. **[Model & Text]**
   Yes, there's only one active maintenance location which is attached but this one can change during the procedure. This is expressed in the model as well as the text.
9. The only way to resolve an alert is to complete the procedure attached to it. **[Model & Text]**
   Yes, see relation "resolves" from Procedure -> Alert and explanation in text.
10. The planner contacts the pilot after touchdown to get the asset to the right location **[None]**
    No, not in the model or text.

Problem solving

1. How does the maintenance process ensure the quality of the performed task? Please name the four implemented measures:
   - *The usage of standardized procedures and tasks*
   - *Quality check by another engineer (see quality_checked_by from Task -> Engineer)*
   - *Certifications of engineers (see certification of Engineer)*
   - *Parts are applied in a standardized way*
   - *Plane can't leave with an error*

2. What should be taken into account while planning the work of an engineer?
   - *The engineer should be available*
   - *The engineer should be certified to perform a certain task*
   - *The work location and / or equipment should be available, otherwise the assigned engineer can't work there.*
   - *Urgency of the repair / alert*
   - *Estimated time of the repair*

3. Which properties of an aircraft effect the maintenance schedule?
   - Age
   - Amount of flying hours
   - Type of aircraft
   - Information from previous tests, procedures or other malfunction reports
     - Previous alerts, errors or failures
     - When was the last scheduled maintenance performed?
   - Current alerts, errors

4. Which circumstances can keep an aircraft grounded?
   - *Maintenance of an aircraft which is part of the regular maintenance schedule but isn't performed in time*
   - *The plane has reached the maximum amount of flying hours*
   - *Alerts / tasks are not resolved, the procedures of these are still running*

Cloze test

1. Based on the availability of a maintenance location / hangar / location$^{C1.1}$ an asset / aircraft$^{C1.2}$ can be repaired by an engineer$^{C1.3}$. The performed procedure / repair / (maintenance) task$^{C1.4}$ is based on an automatic / manually$^{C1.5}$ or automatically / manually$^{C1.6}$ inserted alert. A (maintenance) procedure$^{C1.7}$ contains multiple tasks, within a (maintenance) task$^{C1.8}$ multiple parts / materials $^{C1.9}$ can be used. Every part / material$^{C1.10}$ has a manual.
2. To release the asset / aircraft$^{C2.1}$ a quality check / check / inspection / test /signoff$^{C2.2}$ should be performed by another engineer$^{C2.3}$ which ensures the plane can be set in operation / valid to fly / rotation / active$^{C2.4}$ again.

Part 2 – Questions + answers

Comprehension question

1. Is there a time limit for maintenance on an aircraft? **[New]**
   No, there might be a dependency because the cost of the repair, but none is given.
2. Because the same type of aircraft often faces similar malfunctions, the turnover rate of these particular parts is lower. **[Basic]**
   No, the turnover of these parts is higher
3. To improve stock availability and to minimize costs on spare parts, a minimal stock level is added as a property of a part. **[New]**
   Yes, minimum stock is applicable to parts
4. The estimated maintenance time is of influence for the decision to take an aircraft out of order. **[New]**
   Yes, the estimated costs of the engineers and location are part of this decision.

5. For automatically booking a part at the local warehouse, the costs of these parts are included in the booking. **[New]**
   No, amount and preferred date are included.
6. Costs of the needed hours for engineers, the hangar and the used parts can be calculated before the procedure starts. **[New]**
   Yes, procedures and tasks have a time attached. Tasks contain the used parts.
7. Maintenance can still be planned in a hanger, even though the materials are being used in another hangar at this moment. **[Basic]**
   Yes, it is because there's a maintenance slot that has procedures and parts included.
8. To check the quality of a task a second engineer is planned as part of the procedure. **[Basic]**
   Yes, basic knowledge of part 1.

<u>**Problem Solving**</u>

1. What would you add as a property to parts to improve its availability?
   - *Minimal stock level*
   - *Current stock level / Available stock level*
   - *Turnover rate (popularity/usage of part)*
   - *Currently in use*
   - *Delivery time supplier*
   - *Related alert, belongs to critical or regular alert (and thus can be planned and ordered later)*
   - *Future use of part in relation with scheduled maintenance / demand*
   - *Location of part as a separate entity*
   - *MTBF*

2. What would you change in the system to provide more insight in the costs of maintenance tasks?
   - *Costs/usage of a part (already visible in model)*
   - *Actual worked hours (in model & text part1)*
   - *Hourly cost of an engineer*
   - *Hourly cost of a hangar*
   - *Complete time of maintenance procedure (in hangar / on ground)*
   - *Cost/Time of maintenance procedures in history for future improvement*

3. What kind of additional information should be added to help the planner decide if maintenance is still economically viable?
   - *Time/Costs of the (complete) maintenance*
   - *Cost (overview) of engineer/hangar/parts etc.*
   - *Future costs / gains of an aircraft*
   - *(Current) value of aircraft*
   - *Age of plane (flight hours left)*

4. The Boeing Max 737 has caused multiple crashes by similar malfunctions of a sensor. Regulation entity FAA decided to ground this type of plane. How would you report such a malfunction in our maintenance system?
   - *By reporting this malfunction manually, for all aircraft of this type.*
   - *Adding a parameter to an alert to take the aircraft out-of-order.*

5. Procedures and tasks have a certain time-estimate to complete. How does Acme know the exact <u>personnel</u> costs at the end of the maintenance procedures of a specific airplane?

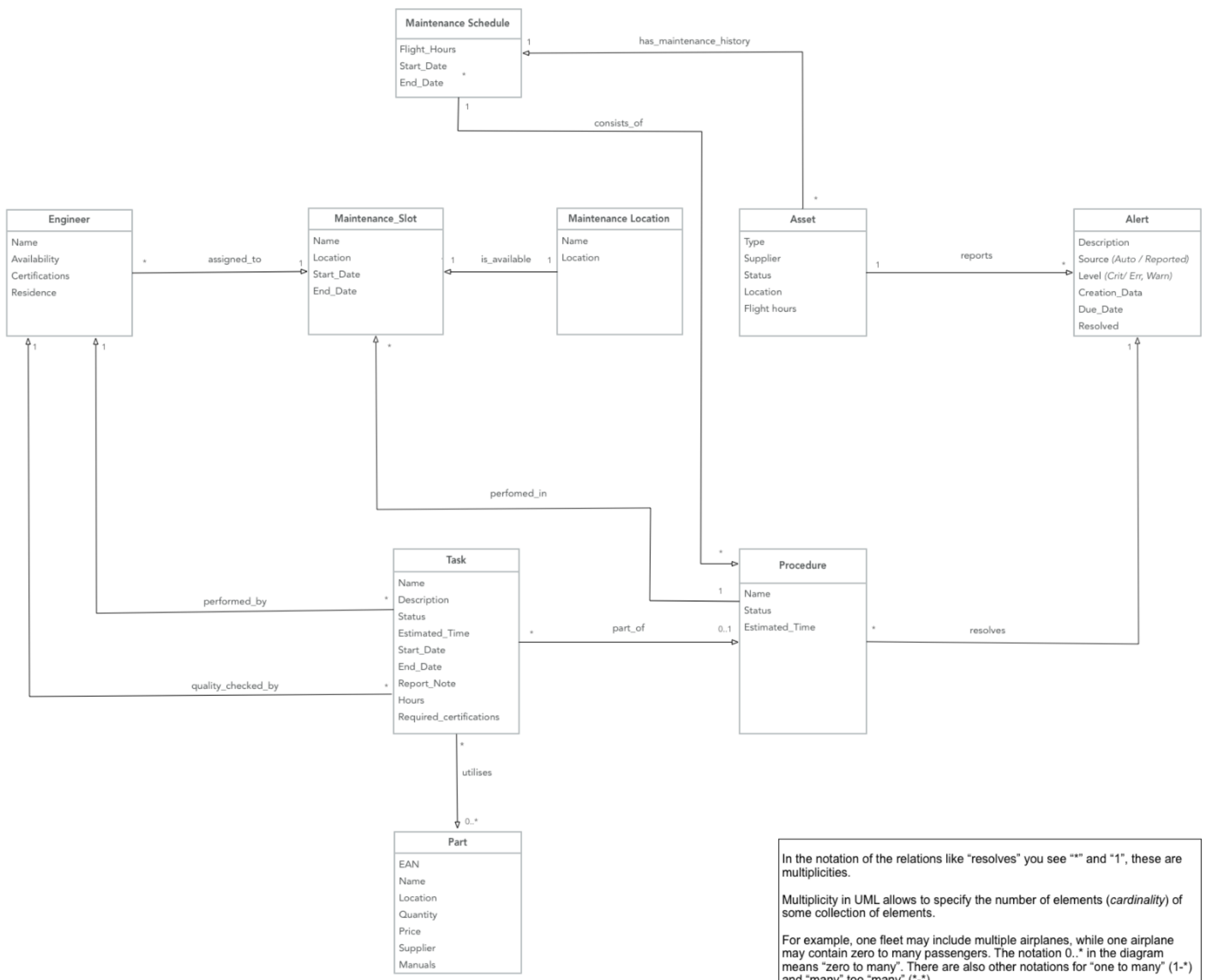
- *The engineers record the actual time for a task.*

Cloze test

1. The delivery time / availability / (minimal) stock level / turnover rate [C1.1], delivery time / availability / (minimal) stock level / turnover rate [C1.2] and delivery time / availability / (minimal) stock level / turnover rate [C1.3] of parts in the warehouse are needed to determine the (planned) date [C1.4] for the next maintenance.

2. If a part [C2.1] reaches a minimum [C2.2] stock level in the warehouse [C2.3], an order [C2.4] will be placed automatically at the supplier [C2.5] by the unknown / system [C2.6]. When ordering parts just-in-time, the delivery time [C2.7] is most important

Appendix C. UML model



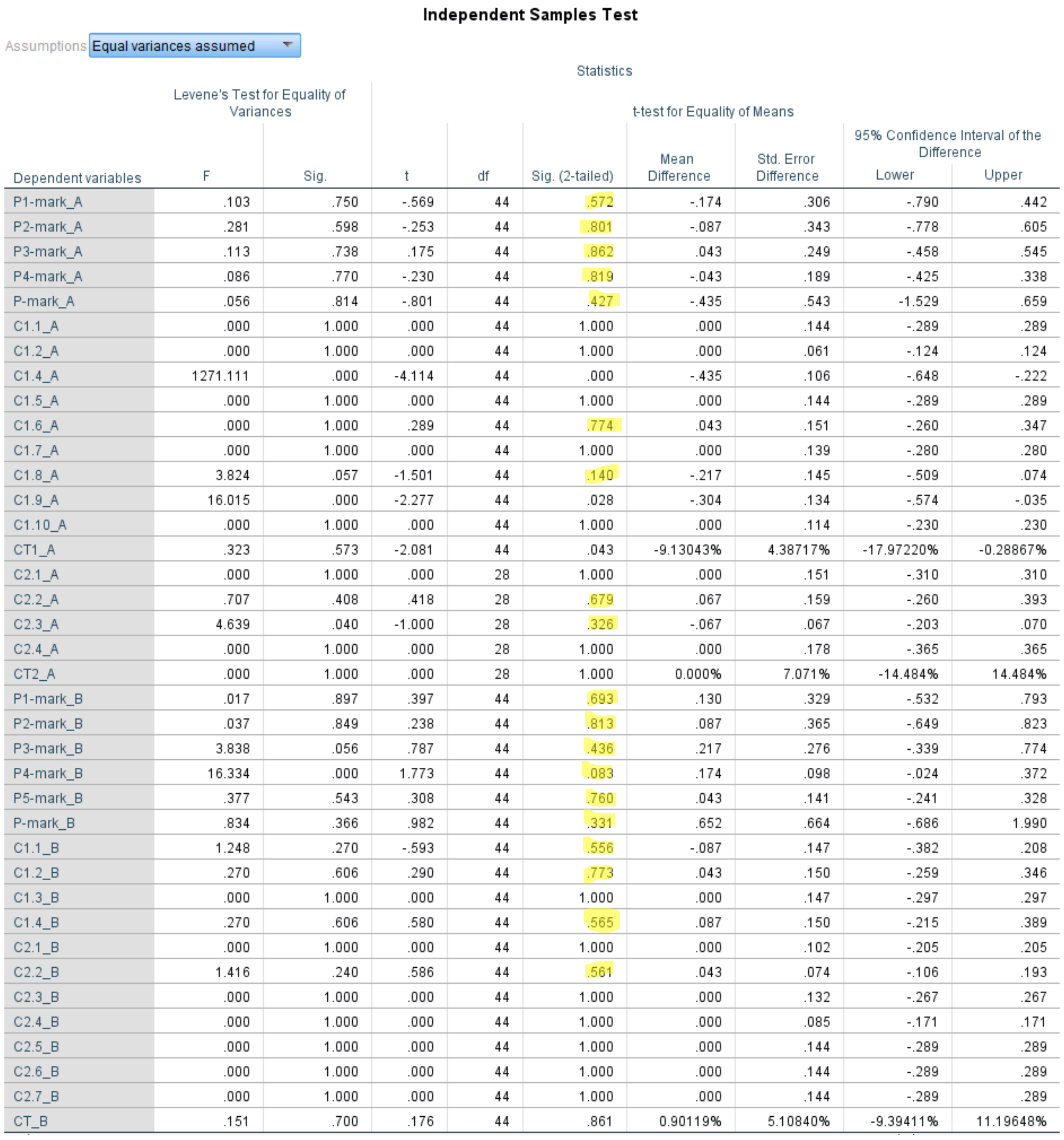
In the notation of the relations like “resolves” you see “*” and “1”, these are multiplicities.

Multiplicity in UML allows to specify the number of elements (*cardinality*) of some collection of elements.

For example, one fleet may include multiple airplanes, while one airplane may contain zero to many passengers. The notation 0..* in the diagram means “zero to many”. There are also other notations for “one to many” (1-*) and “many” too “many” (*-*).

Appendix D. Samples T-Test

Initial Samples T-Test to test the grading differences between the graders.



Independent Samples Test

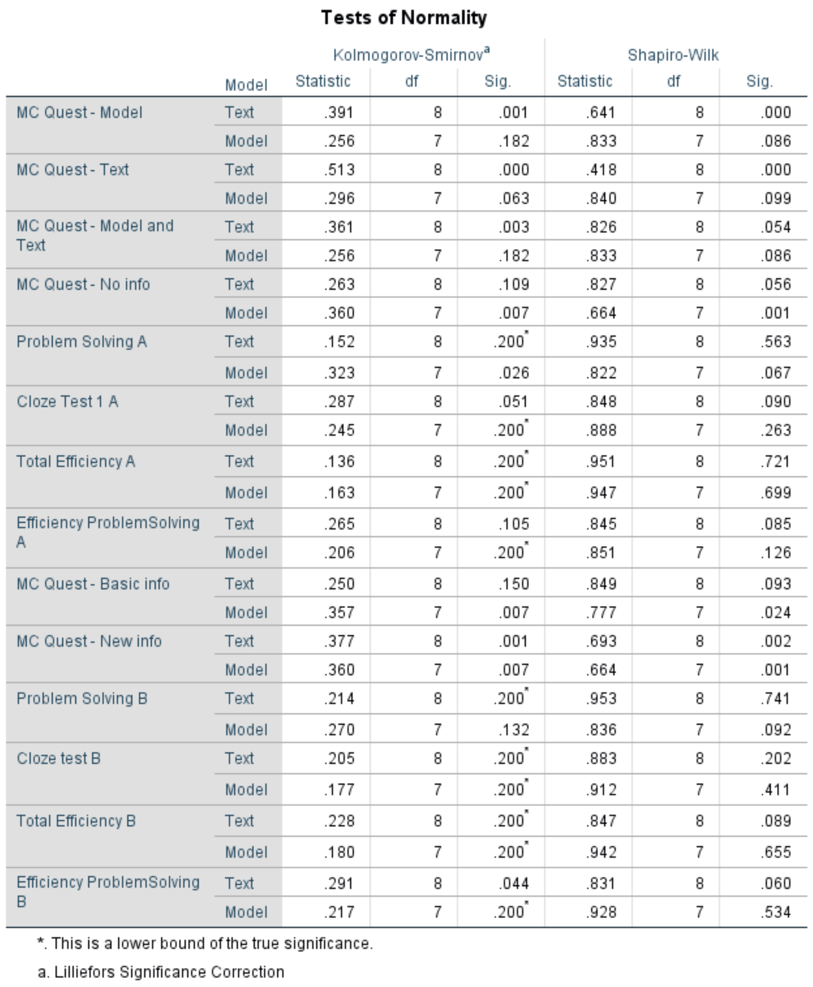
Assumptions: Equal variances assumed

Statistics

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
Dependent variables	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
P1-mark_A	.103	.750	-.569	44	.572	-.174	.306	-.790	.442
P2-mark_A	.281	.598	-.253	44	.801	-.087	.343	-.778	.605
P3-mark_A	.113	.738	.175	44	.862	.043	.249	-.458	.545
P4-mark_A	.086	.770	-.230	44	.819	-.043	.189	-.425	.338
P-mark_A	.056	.814	-.801	44	.427	-.435	.543	-1.529	.659
C1.1_A	.000	1.000	.000	44	1.000	.000	.144	-.289	.289
C1.2_A	.000	1.000	.000	44	1.000	.000	.061	-.124	.124
C1.4_A	1271.111	.000	-4.114	44	.000	-.435	.106	-.648	-.222
C1.5_A	.000	1.000	.000	44	1.000	.000	.144	-.289	.289
C1.6_A	.000	1.000	.289	44	.774	.043	.151	-.260	.347
C1.7_A	.000	1.000	.000	44	1.000	.000	.139	-.280	.280
C1.8_A	3.824	.057	-1.501	44	.140	-.217	.145	-.509	.074
C1.9_A	16.015	.000	-2.277	44	.028	-.304	.134	-.574	-.035
C1.10_A	.000	1.000	.000	44	1.000	.000	.114	-.230	.230
CT1_A	.323	.573	-2.081	44	.043	-9.13043%	4.38717%	-17.97220%	-0.28867%
C2.1_A	.000	1.000	.000	28	1.000	.000	.151	-.310	.310
C2.2_A	.707	.408	.418	28	.679	.067	.159	-.260	.393
C2.3_A	4.639	.040	-1.000	28	.326	-.067	.067	-.203	.070
C2.4_A	.000	1.000	.000	28	1.000	.000	.178	-.365	.365
CT2_A	.000	1.000	.000	28	1.000	0.000%	7.071%	-14.484%	14.484%
P1-mark_B	.017	.897	.397	44	.693	.130	.329	-.532	.793
P2-mark_B	.037	.849	.238	44	.813	.087	.365	-.649	.823
P3-mark_B	3.838	.056	.787	44	.436	.217	.276	-.339	.774
P4-mark_B	16.334	.000	1.773	44	.083	.174	.098	-.024	.372
P5-mark_B	.377	.543	.308	44	.760	.043	.141	-.241	.328
P-mark_B	.834	.366	.982	44	.331	.652	.664	-.686	1.990
C1.1_B	1.248	.270	-.593	44	.556	-.087	.147	-.382	.208
C1.2_B	.270	.606	.290	44	.773	.043	.150	-.259	.346
C1.3_B	.000	1.000	.000	44	1.000	.000	.147	-.297	.297
C1.4_B	.270	.606	.580	44	.565	.087	.150	-.215	.389
C2.1_B	.000	1.000	.000	44	1.000	.000	.102	-.205	.205
C2.2_B	1.416	.240	.586	44	.561	.043	.074	-.106	.193
C2.3_B	.000	1.000	.000	44	1.000	.000	.132	-.267	.267
C2.4_B	.000	1.000	.000	44	1.000	.000	.085	-.171	.171
C2.5_B	.000	1.000	.000	44	1.000	.000	.144	-.289	.289
C2.6_B	.000	1.000	.000	44	1.000	.000	.144	-.289	.289
C2.7_B	.000	1.000	.000	44	1.000	.000	.144	-.289	.289
CT_B	.151	.700	.176	44	.861	0.90119%	5.10840%	-9.39411%	11.19648%

Figure 13 Initial Samples T-test

Appendix E. Descriptive statistics



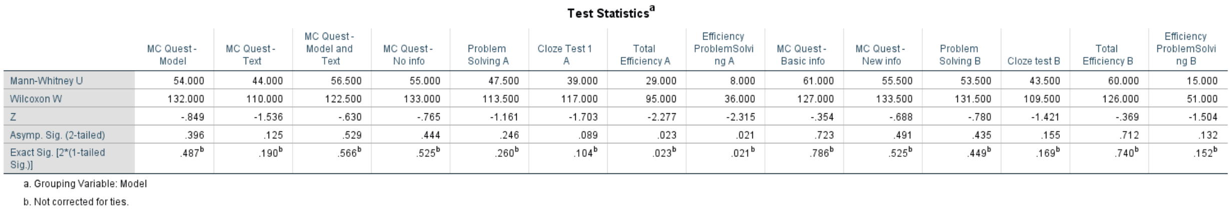
Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Model	Statistic	df	Sig.	Statistic	df	Sig.
MC Quest - Model	Text	.391	8	.001	.641	8	.000
	Model	.256	7	.182	.833	7	.086
MC Quest - Text	Text	.513	8	.000	.418	8	.000
	Model	.296	7	.063	.840	7	.099
MC Quest - Model and Text	Text	.361	8	.003	.826	8	.054
	Model	.256	7	.182	.833	7	.086
MC Quest - No info	Text	.263	8	.109	.827	8	.056
	Model	.360	7	.007	.664	7	.001
Problem Solving A	Text	.152	8	.200*	.935	8	.563
	Model	.323	7	.026	.822	7	.067
Cloze Test 1 A	Text	.287	8	.051	.848	8	.090
	Model	.245	7	.200*	.888	7	.263
Total Efficiency A	Text	.136	8	.200*	.951	8	.721
	Model	.163	7	.200*	.947	7	.699
Efficiency ProblemSolving A	Text	.265	8	.105	.845	8	.085
	Model	.206	7	.200*	.851	7	.126
MC Quest - Basic info	Text	.250	8	.150	.849	8	.093
	Model	.357	7	.007	.777	7	.024
MC Quest - New info	Text	.377	8	.001	.693	8	.002
	Model	.360	7	.007	.664	7	.001
Problem Solving B	Text	.214	8	.200*	.953	8	.741
	Model	.270	7	.132	.836	7	.092
Cloze test B	Text	.205	8	.200*	.883	8	.202
	Model	.177	7	.200*	.912	7	.411
Total Efficiency B	Text	.228	8	.200*	.847	8	.089
	Model	.180	7	.200*	.942	7	.655
Efficiency ProblemSolving B	Text	.291	8	.044	.831	8	.060
	Model	.217	7	.200*	.928	7	.534

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Figure 14 Tests of Normality



Test Statistics[a]

	MC Quest - Model	MC Quest - Text	MC Quest - Model and Text	MC Quest - No info	Problem Solving A	Cloze Test 1 A	Total Efficiency A	Efficiency ProblemSolving A	MC Quest - Basic info	MC Quest - New info	Problem Solving B	Cloze test B	Total Efficiency B	Efficiency ProblemSolving B
Mann-Whitney U	54.000	44.000	56.500	55.000	47.500	39.000	29.000	8.000	61.000	55.500	53.500	43.500	60.000	15.000
Wilcoxon W	132.000	110.000	122.500	133.000	113.500	117.000	95.000	36.000	127.000	133.500	131.500	109.500	126.000	51.000
Z	-.849	-1.536	-.630	-.765	-1.161	-1.703	-2.277	-2.315	-.354	-.688	-.780	-1.421	-.369	-1.504
Asymp. Sig. (2-tailed)	.396	.125	.529	.444	.246	.089	.023	.021	.723	.491	.435	.155	.712	.132
Exact Sig. [2*(1-tailed Sig.)]	.487[b]	.190[b]	.566[b]	.525[b]	.260[b]	.104[b]	.023[b]	.021[b]	.786[b]	.525[b]	.449[b]	.169[b]	.740[b]	.152[b]

a. Grouping Variable: Model

b. Not corrected for ties.

Figure 15 Two Independent Samples Test

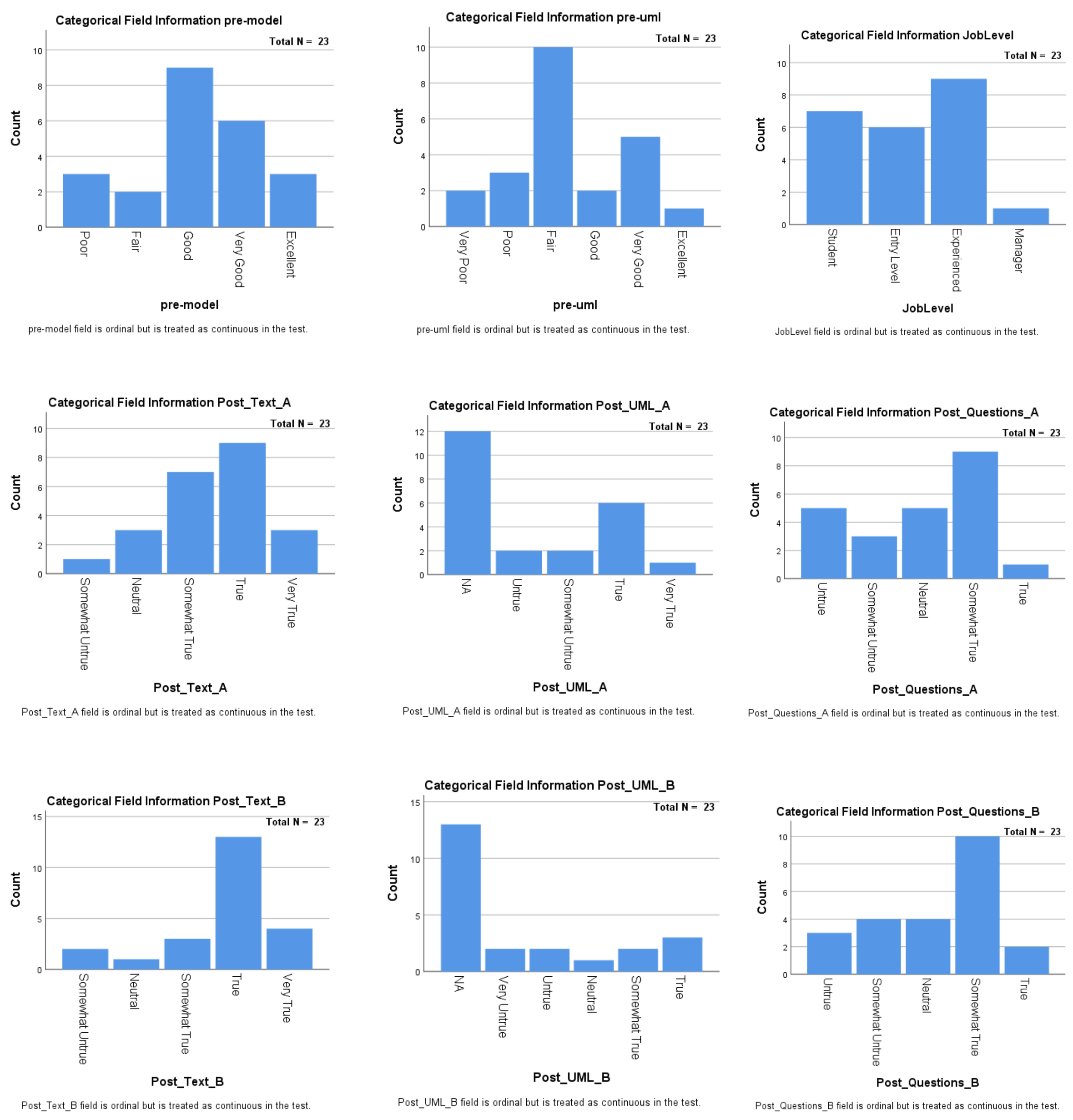
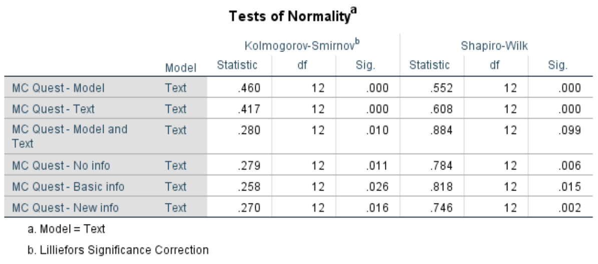


Figure 16 pre and post questions

Appendix F. Statistics H1A / H2A

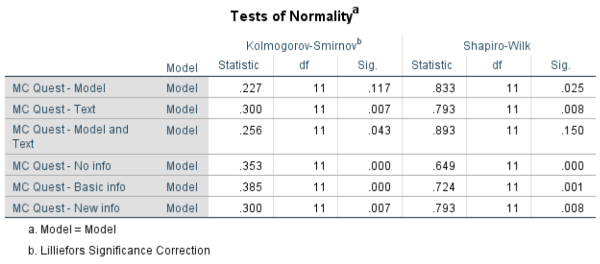


Tests of Normality[a]

		Kolmogorov-Smirnov[b]			Shapiro-Wilk		
	Model	Statistic	df	Sig.	Statistic	df	Sig.
MC Quest - Model	Text	.460	12	.000	.552	12	.000
MC Quest - Text	Text	.417	12	.000	.608	12	.000
MC Quest - Model and Text	Text	.280	12	.010	.884	12	.099
MC Quest - No info	Text	.279	12	.011	.784	12	.006
MC Quest - Basic info	Text	.258	12	.026	.818	12	.015
MC Quest - New info	Text	.270	12	.016	.746	12	.002

a. Model = Text

b. Lilliefors Significance Correction

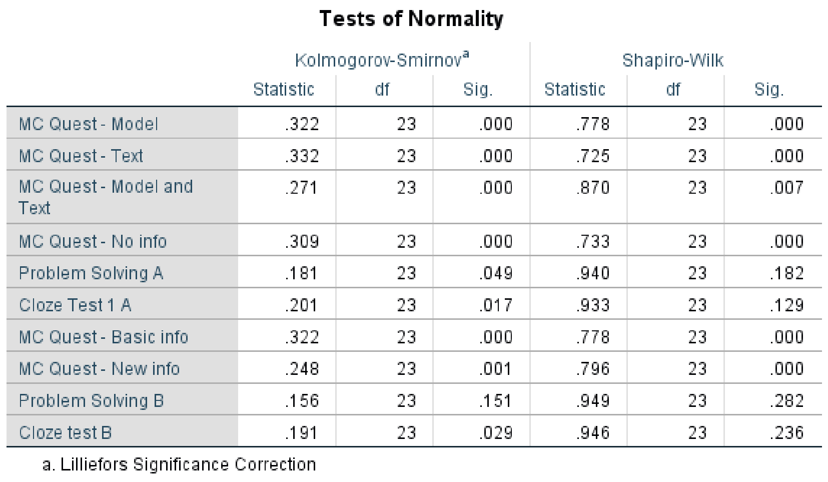


Tests of Normality[a]

		Kolmogorov-Smirnov[b]			Shapiro-Wilk		
	Model	Statistic	df	Sig.	Statistic	df	Sig.
MC Quest - Model	Model	.227	11	.117	.833	11	.025
MC Quest - Text	Model	.300	11	.007	.793	11	.008
MC Quest - Model and Text	Model	.256	11	.043	.893	11	.150
MC Quest - No info	Model	.353	11	.000	.649	11	.000
MC Quest - Basic info	Model	.385	11	.000	.724	11	.001
MC Quest - New info	Model	.300	11	.007	.793	11	.008

a. Model = Model

b. Lilliefors Significance Correction

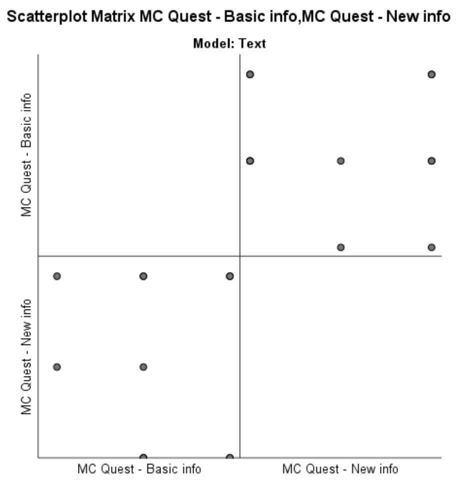


Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MC Quest - Model	.322	23	.000	.778	23	.000
MC Quest - Text	.332	23	.000	.725	23	.000
MC Quest - Model and Text	.271	23	.000	.870	23	.007
MC Quest - No info	.309	23	.000	.733	23	.000
Problem Solving A	.181	23	.049	.940	23	.182
Cloze Test 1 A	.201	23	.017	.933	23	.129
MC Quest - Basic info	.322	23	.000	.778	23	.000
MC Quest - New info	.248	23	.001	.796	23	.000
Problem Solving B	.156	23	.151	.949	23	.282
Cloze test B	.191	23	.029	.946	23	.236

a. Lilliefors Significance Correction

Figure 17 Tests of normality



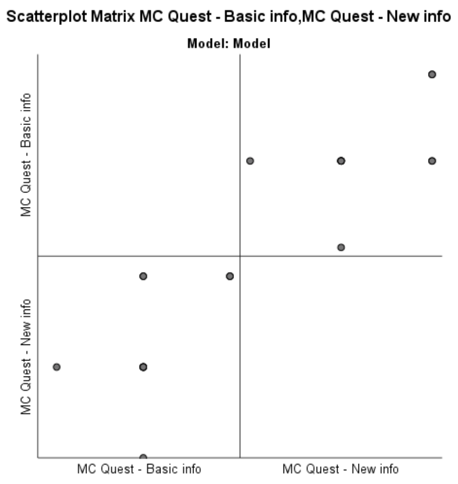


Figure 18 Scatterplot for test on linearity on Text and Model

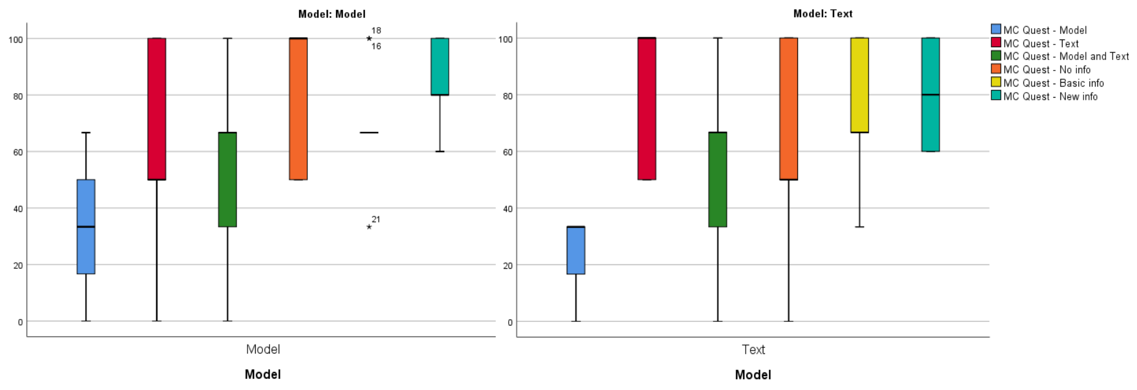


Figure 19 Boxplot models for H1A and H2A on text/model

Appendix G. Statistics H3

Tests of Normality[b,c,d,e]

			Kolmogorov-Smirnov[a]			Shapiro-Wilk		
JobLevel		Model	Statistic	df	Sig.	Statistic	df	Sig.
Student	Residual for Pmark_B	Text	.385	3	.	.750	3	.000
		Model	.260	4	.	.827	4	.161
	Residual for TM_PT_B	Text	.313	3	.	.895	3	.368
		Model	.238	4	.	.968	4	.831
Entry Level	Residual for Pmark_B	Text	.314	3	.	.893	3	.363
	Residual for TM_PT_B	Text	.250	3	.	.966	3	.648
Experienced	Residual for Pmark_B	Model	.	2	.			
	Residual for TM_PT_B	Model	.260	2	.			

Tests of Normality[b,c,d,e,f,g]

			Kolmogorov-Smirnov[a]			Shapiro-Wilk		
Model		JobLevel	Statistic	df	Sig.	Statistic	df	Sig.
Text	Residual for Pmark_B	Student	.385	3	.	.750	3	.000
		Entry Level	.314	3	.	.893	3	.363
	Residual for TM_PT_B	Student	.313	3	.	.895	3	.368
		Entry Level	.250	3	.	.966	3	.648
Model	Residual for Pmark_B	Student	.260	4	.	.827	4	.161
		Experienced	.	2	.			
	Residual for TM_PT_B	Student	.238	4	.	.968	4	.831
		Experienced	.260	2	.			

Figure 20 Shapiro-Wilk's Tests of normality

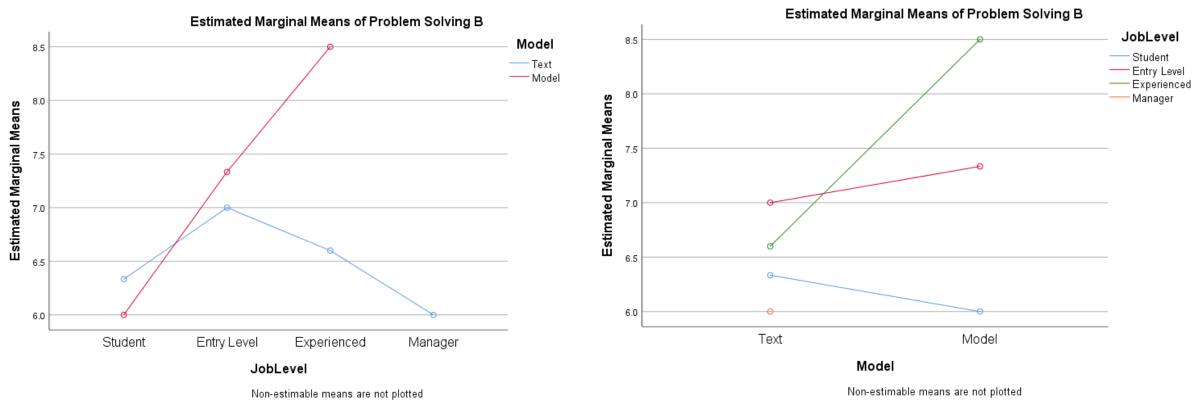


Figure 21 Estimated Means plots on Problem Solving and Job level

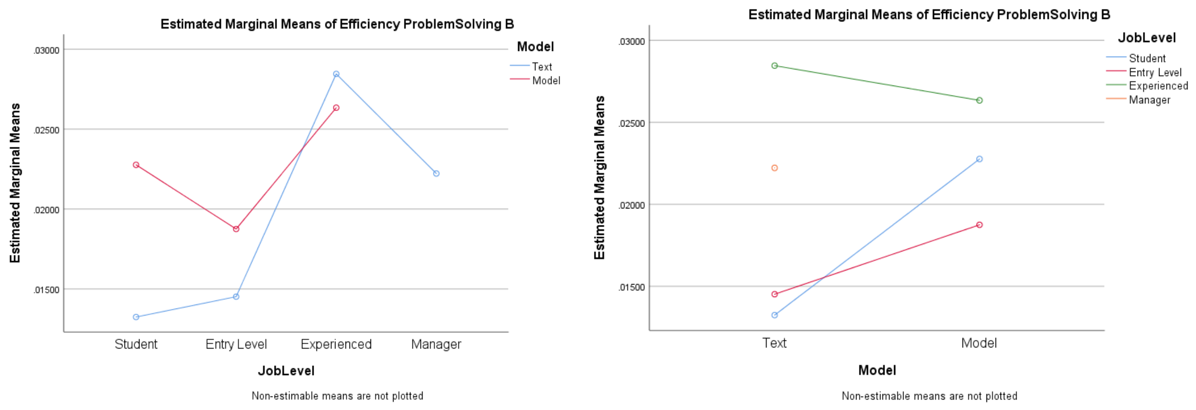


Figure 22 Estimated Means plots on Model and Job level