MSc Thesis

Sanam Ali (s114657), 24.2.2012

Weaving Requirements Fragments: An empirical investigation

Department of Applied Mathematics and Computer Science

Technical University of Denmark

DTU Compute
Department of Applied Mathematics and Computer Science
Building 303 B,DK-2800 Kongens Lyngby, Denmark
Telephone: +45 45253031, Fax: +45 45881399
compute@compute.dtu.dk

Abstract

In requirements engineering the process of converting textual requirements into formal models is a very intricate process which has attracted the interest of many researchers. This thesis proposes the method of transforming requirements stated in natural language into formal models (UML representations have been used in this thesis) by adding model fragments. These small model fragments can then be woven through iterative fragment weaving to obtain formal models. The goal of the thesis is to 1) formalize rules for the creating model fragments, 2) to explore whether model fragments can be created for a large number of requirements (case study) and 3) to conduct an experiment to validate the transformation rules.

The model fragments not only supports the forward process of transforming textual requirements into formal models but also have push back on the requirements. Model fragment creation modifies the textual requirements and also helps in adding new requirements. The case study and the experiment provided the author with various empirical insights concerning model fragments.

Acknowledgments

This is the Master's Thesis for Sanam Ali presented to the Technical University of Denmark as partial fulfillment of the requirements for obtaining the Master's Degree in Computer Science and Engineering at the Department of Applied Mathematics and Computer Science. This thesis has been written during the period of September 29th 2013 to February 24th 2014, under the supervision of Prof. Dr. Harald Störrle, and is worth 30 ECTS credits.

Foremost I would like to thank my supervisor Harald Störrle for not only being a helpful supervisor but a great mentor as well. Your patience, time, knowledge and enthusiasm towards my work have always motivated me to do my best. I hope this thesis will be helpful to you in your future work.

I would also like to thank Henrik Meilby, Associate Professor at University of Copenhagen for his time and help. His advice and insights regarding statistical concepts proved very helpful.

Last but not the least I would like to thank my family, without their support and prayers I would not have been able to complete this thesis. My sisters who have always believed in me and supported in whatever I have chosen to do.

Conpute, DTU, February 24th, 2014

Sanam Ali (s114657)

Table of Contents

1 Introduction

Chapter 1 Introduction

A picture is worth a thousand words. Arthur Brisbane (1911).

This phrase emerged in the early part of the 20th century and has been widely used ever since. It means that a complex idea which can be told in thousand words could easily be depicted in a single picture. A similar concept is being discussed in this thesis. This thesis focuses on adding model fragments in order to transform natural language requirements into formal models. These model fragments will not only prove beneficial for the developing team, but will hopefully reduce the communication gap between clients and requirement engineer.

At present there are no definite rules in the Requirements Engineering field, which can be used to create with model fragments for a set of requirements. The first part of the thesis attempts at formulating with some basic rules for doing so. Using the transformation rules model fragments are created against the pre stated requirements of a case study in the second part of the thesis. The rules are tried to be validated by carrying out a controlled experiment on the students of the course 02264 Requirements Engineering at the Technical University of Denmark. The results and observations from the experiment have been analyzed to obtain various insights.

For this thesis existing requirements from three different case studies have been worked upon. These three case studies have been taught during the course of 02264 Requirements Engineering at the Technical University of Denmark. These are Library Management System (LMS) (2011), Sjællands Banken Customer Kiosk (SBK) (2012) and Mobile Match Maker System (MMM) (2012). For the second part of the thesis only the LMS case study has been considered while the experiment conducted considers all the three case studies.

The remaining thesis has been divided into the following sections chapter 1 includes motivation for the thesis, the previous work done by others and describes the concept of fragment weaving and traceability (presents the big picture). In chapter 2 the various concepts associated with model fragments are discussed including how the transformation rules can be applied on requirements to create model fragments. Chapter 3 focuses on a case study, its results and the interpretation of the various results. Chapter 4 is dedicated to the experiment, methods and materials, observations and interpretation of the different observation. Chapter 5 is the discussion and Chapter 6 proposes some future work and conclusion.

1.1 Motivation

For the development of any type of system, Requirements engineering is undoubtedly a crucial part of the developments process. And in order to collect complete and real requirements, it is important to communicate with all the various stakeholders of the system.

The definition of stakeholders in the PMBOK® Guide (2008) is given as:

> *Stakeholder: An individual, group or organization who may affect, be affected by, or perceive itself to be affected by a decision, activity or outcome of the project.*

There are different types of stakeholders with different type and levels of academic knowledge. Not all the stakeholders are able to understand the syntax and semantics of UML and other visual models (Arlow & Neustadt, 2004). For example the developers, programmers, managers may be familiar with the requirement engineering terms and concepts while others like the customer, clients, suppliers etc. may not familiar to these concepts. For the latter group understanding the formal models and complex concepts of requirement engineering would be an endeavor in itself, hence the validation of the requirements from formal models not possible. Therefore the addition of simple, small uncomplicated model fragments to the requirements would help them in understanding and tracing their requirement in the complex large formal models. This problem has also been talked about by Al-Rawas and Easterbrook (1996, p. 5) where it is stated:

> “*Regardless of the chosen notations, most users express their requirements in natural language. Then it is the job of the analyst to translate requirements statements into some kind of representational objects in a domain model. Once the requirements are modelled, they are presented to end users for validation. At this stage the analysts are faced with another communication problem when end users are not familiar with the notations used to model their requirements.*"

This shows that the stakeholders are not able to validate the requirements as they do not have the knowledge to comprehend the complex and large models presented by the designers.

According to Davis (2005) the requirement management process can be improved by using both the models and the requirements written down in natural language. The advantage of having modeling techniques is that they are precise, expressive and helps the developing team to understand the requirements. Whereas the benefit of text is that it can act as a contract between the clients and the developers. Designers have the tendency to make model fragments without including them to the requirement specification document. As Lange et al. (2006) pointed out that one of the problems of using UML is its informal use

> *"Architects sometimes use UML in a very sketchy manner. For example, to obtain a better understanding of a system or to explain the architecture, they might use generic drawing software or even just make sketched on paper. These diagram deviate from official UML syntax, making their meaning ambiguous".*

The advantages and disadvantages of UML is out of scope of this thesis but Christian et al., statement shows that the initial step for a designer is usually a sketch of the system on a paper or a generic drawing tool. Adding to this Fowler (2003) goes so far as to say that UML could be used as a sketch, blue print and as a programming language again suggesting that the initial step for a designer to create a UML is usually a rough sketch. Thus an intermediate phase between translating the requirements expressed in natural language directly into UML models should be introduced to allow the clients to verify the requirements, at the same time allows the designers with initial phase of creating the formal models.

An important aspect in the development process is requirement traceability. According to Ramesh and Jarke (2001), requirement traceability allows the systems' requirements to be unmistakably linked to its source and the derived artifacts of these requirements during the entire development life cycle. Thus a person can view how his/her requirement described in natural language is modeled into some high language model and then programmed into a feature of the system.

As stated earlier this traceability can be a bit tricky if the stakeholders are not familiar with the high language models and thus introducing small and simple model fragments to the requirements will help the stakeholders in tracing their requirement. Traceability also helps in giving information about justification, important decisions and assumptions behind requirements (Ramesh & Jarke, 2001). Thus tracing a requirement back to the right source and the stakeholder being able to validate the requirement is important.

1.2 Related work

In this section the previous work of some researchers and scholars will be discussed and compared to get an idea of what has been done in the field of requirements engineering to automate/semi automate the process of transforming textual requirements into formal models. This transformation can be carried out by two different methods (Fatwanto A. , 2012).

1. Artificial intelligence (rule based approach)
2. Grammatical analysis (natural language processing approach)

In the rule based approach, artificial intelligence is used in the transformation of informal specification to formal specification. Some common examples of using rule based approach are Specification Acquisition from Expert (1978), the Requirement Apprentice (1991) and Specifier (1991)

The second type uses linguistic analysis i.e. grammatical knowledge to transform textual requirements into formal Specification. Examples are Conceptual Model Builder (2000), Language Extended Lexicon (2004) and Natural Language Processing (NLP) (2008).

Moreno and van de Riet (1997) represents an approach for comparing linguistic world (i.e. requirements in textual format) to conceptual world (i.e. class models). The requirements in natural language are converted into First Order Logic and conceptual models are converted into Set theory representation. Then using mathematical theories and concepts the two are compared. Moreno and van de Riet work does not discuss forward or backward traceability in any way. Fatwanto (2012) present an approach of translating textual requirements in CARE (Fatwanto A. , 2011) framework into class and state machines. His work does not support the concepts of forward or backward traceability and even though the transformation from requirements to models has been implemented, his work has not been validated in any way.

A tool called TESSI (Kroha, 2000) supports iterative RE processes. In TESSI the analyst specifies the roles of words in text. In complete roles will lead to incomplete UML models. Thus this tool has the limitation that the analysts have to consider every role of every word during the modeling process. Also the generation of requirements specification from a modified model, will result in a new set of requirements specification instead of updating the original one.

There are many different criteria’s on the basis of which the existing methods, approaches and tools for textual requirement transformation can be compared. But for the purpose of this thesis, some tools and methods have been considered and categorized as shown in Table 1.

The Table 1 shows which methods supports forward and backward traceability, in what format or framework the initial textual requirements are in and to what extent is the model generation process automated. It also includes a column on how the various methods have been validated. As it can be seen there are many approaches that attempt to fully automate this process, but the very idea of automation in the context of RE is somewhat unintuitive to us. Indeed, some of the approaches mentioned, has been validated by experiments or case studies. But, most of them have not been validated at all. In contrast to this work, we pursue an approach that relies on human understanding and insight, guided by rules and examples.

Table 1 Comparison of Previous Work on the Basis of Certain Parameters

Author & Year	Scope	textual requirements	Automation (from requirements to model)	Forward traceability	Automation (from models to requirements)	Backward traceability	Validated
Moreno & van de Riet (1997)	Class models	Restricted to some extend	Manual (transformation rules applied)	No	No	No	Mathematical reasoning
Kroha (2000)	-	-	Automated	Yes	Yes	Yes	-
Overmyer et al. (2001)	Class models	Not restricted	Semi-automatic	Yes	Yes	Yes	No
Kealey and Amyot (2006)	Use case	UCEd framework	Automated (jUCMNav tool)	Yes (Telelogic DOORS)	No	No	No
Gelhausen & Tichy (2007)	Not stated but example of class model given	Transformed into SENSE representation	Manual (automation proposed as future work)	Not mentioned	No	No	One case study
Montes et al. (2008)	Class models	Considers only use case scenarios	Manual	No	No	No	No
Hasegawa et al. (2009)	OO model + feature model	Not restricted	Automated	No	No	No	Experiment to validate tool
Agung Fatwanto (2012)	Class+ state machine	Follows CARE framework	Transformation rules (implemented)	No	No	No	No
Landhäußer (2014)	All models supported by SALe mx (model generator)	Complies with RESI	Fully automated	Yes	Yes	Yes	Several case study

1.3 Approach

The idea behind adding model fragments to a requirement is to automate/semi-automate the process of transforming textual requirements into formal models while incorporating traceability. In this section the definition of requirement traceability along with related concepts is given. Also the process of forward and backward traceability from textual requirements to formal Models using model fragments is discussed. According to Gotel and Finkelstein (1994) requirement traceability can be defined as

> *"The requirement traceability is the ability to describe and follow the life of a requirement, in both forward and backward direction, i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases."*

As stated from the definition of requirement traceability there are two different types of traceability, backward and forward traceability. According to Wieringa (1995) they can be defined as

> ***Forward traceability*** *of a requirement is the ability to trace components of a requirements specification to components of a design or of an implementation.*
>
> ***Backward traceability*** *of a requirement is the ability to trace a requirement to its source i.e. to a person, institution, law, argument, etc. that demands the requirement to be present.*

The process of transforming textual requirements into formal models starts with a set of requirements described in natural language. These requirements are not restricted to any particular format like the UCEd (Cockburn, 2000) , RESI (körner & Brumm, 2010) or CARE Framework (Fatwanto A. , 2011) rather considers the requirements as text specified by the client and Analyst together. Against these requirements as many model fragments as possible are generated using the transformation rules (Appendix A). These model fragments can be of various different types like class, state machine, use case, sequence (interaction) and activity models. It should be noted that not all the requirements will generate equal number or types of model fragments. Some requirements may generate 3, others 1 while some requirements might not have any model fragments associated with it.

The next step is the transformation of these small sized model fragments into larger models and eventually formal models (for example UML models). This can be done by doing a series of fragment weaving. The small fragments of the same type are woven into slightly larger fragments until the formal models are achieved.

This phenomenon is illustrated in Figure 1 which shows a set of requirements req_1, req_2… req_n. These requirements generate different number and type of model fragments. The models fragments of the same type are woven together through successive fragment weaving that results in one overall model file comprising of the different types of formal models. Fragment weaving is out of scope of this thesis, but Jakob's Master thesis (2012) describes fragment weaving in the RED tool to some extent.

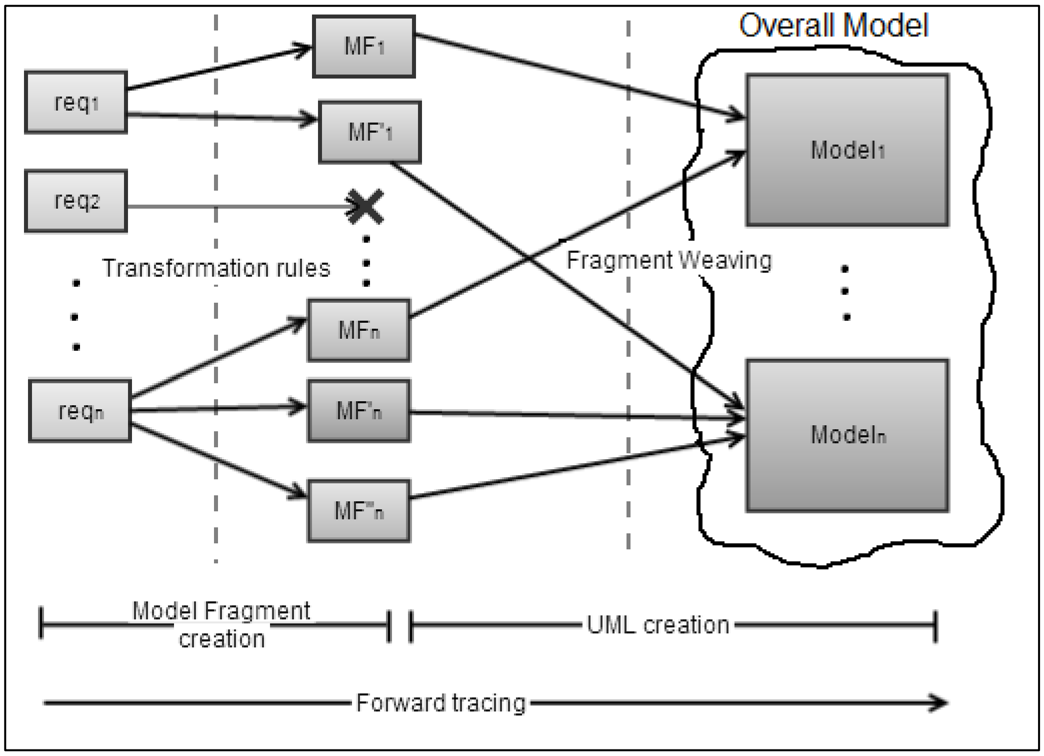


Figure 1 Transforming Textual Requirements to Formal Models by Creating Model Fragments (Forward Tracing)

The push back of creating model fragments is also very interesting to see. The creation of model fragments may require modification of the original requirement text. These modifications can enhance, decrease or have no impact on the quality of the requirements. This will be determined and discussed later on in Chapter 3. Often it has been seen that while specifying requirements, both the client and analyst miss writing the simple and obvious requirements. It may happen because both the parties consider them to be trivial. But ideally this should not happen. Thus in addition to modifying the textual requirements, the creation of model fragments can be used to visualize the missing trivial requirements. This is a very important push back of the model fragments and will help analysts in creating a complete SS document of a system. In Figure 2 the push back phenomenon of model fragments have been visualized. The red color model fragment represents the source of modification for its corresponding requirement. Another model fragment (green color) represents a model fragment that has been added, due to which its corresponding requirement has been added to the SS document of the developing system.

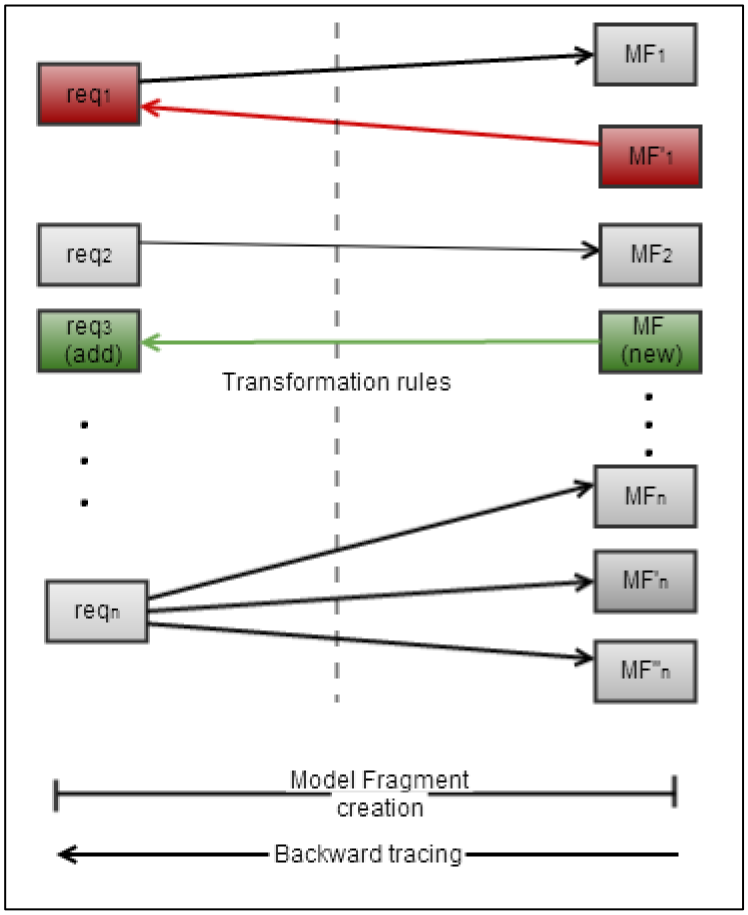


Figure 2 The Push Back of Model Fragments on the Textual Requirements (Backward Tracing)

Chapter 2 Model Fragments for Requirement Engineering

2.1 Model fragments

According to Siikarla et al., (2006) the concept of model fragment is undefined, hence there does not exist a proper, correct definition of model fragments, yet they have tried to define model fragment to some extent. According to which a model fragment can be defined as "*a copy of a piece of a model. A model fragment is a concrete entity distinct from the model and in that way different from a view.*" Siikarla et al., (2006) work comprised of splitting a given complex and large model to smaller pieces. Hence the definition is derived in that context.

For this thesis the model Fragments can be defined as a simple, small, less detailed and incomplete graphical representation of a requirement. The concrete entity part described by Siikarla et al., (2006) still holds for this definition. The need and importance of model fragments have been discussed in section 1.1.

2.2 Properties of Model fragments

The concept behind including these model fragments to convert textual requirements into formal models supporting forward and backward traceability. For this purpose the model fragments should correspond to a number of properties, which are

- Model fragments should be small. Requirements generating large model fragments should be split up if possible.
- Model fragments should be simple and easy to understand.
- The constraints and limitations of UML modeling should not be applicable to generating model fragments.
- Model fragments should appear like they hand sketched. This will allow designers to make changes/addition to the models as they appear incomplete.
- A model fragment should only represent the requirement it is drawn against.

These properties have been kept in mind by the author when selecting a tool for recording the model fragments (section 3.3).

2.3 Selection of Model Fragments Types

For the purpose of this thesis, the UML modeling notations are used. Just as in UML there are many different models (use case, component, state machine, classes and many others), various types of model fragments can also be created. In order to make the scope of the thesis realistic, findings of Dobing and Parsons (2006) will be used to select different types of model fragments for this thesis.

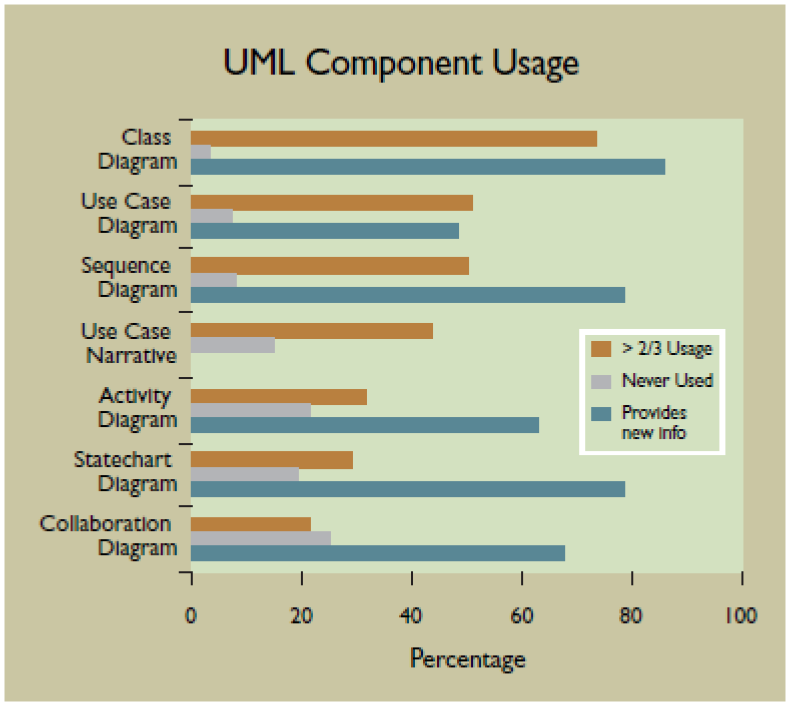


Figure 3 Use and Perceived Information Added of UML Components, Source (Dobing & Parsons, 2006)

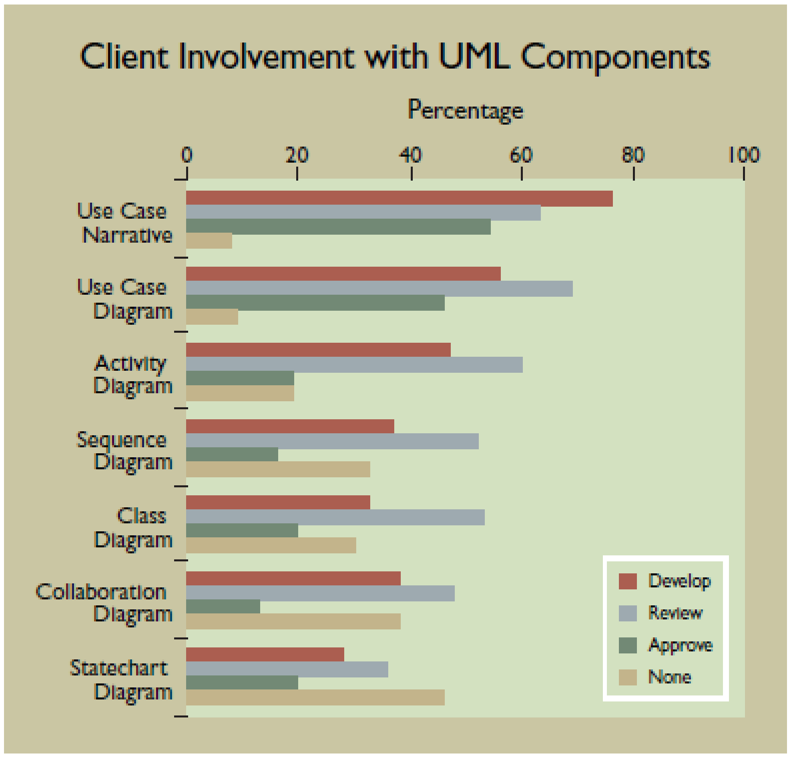


Figure 4 Client involvement, Source (Dobing & Parsons, 2006)

As it can be seen from Figure 3 and Figure 4 the usage and client involvement for the model types use case, class, sequence (interaction), state machine and activity is the highest, thus these model types will be considered for this thesis. Even though the client involvement for use case narrative is high, it is not being considered because the RED tool does not support it (selection of tool for recording model fragments has been discussed in section 3.3). Similarly the collaboration diagram usage is very low as shown in Figure 3; hence it is not given a priority at the moment. Future work can be done by considering various other model types.

2.4 Illustration of the Various Types of Model Fragments

This section includes examples for the different types of model fragments with its associated requirement. It is to show how the different requirements and their associated model fragments look like. For this section each requirement will only have one type of model fragment created against it, but it should be remembered that a single requirement can generate more than one type of model fragment.

These examples have been taken from the LMS case study, introduction to the LMS case study is given in Chapter 3 discussed in. Examples of the various different types of model fragments with their respective requirement is given below

2.4.1 Activity Model Fragment

The example given below represents a simple activity model consisting of swim lanes, activities and control flow. Other model elements like case distinction, decision node and objects can also be generated as part of the model fragment using the rules described in Appendix A.

Requirement ID	Text
REI5	If a librarian initiates reader actions on behalf of a reader, the reader receives notification.

Model Fragment:

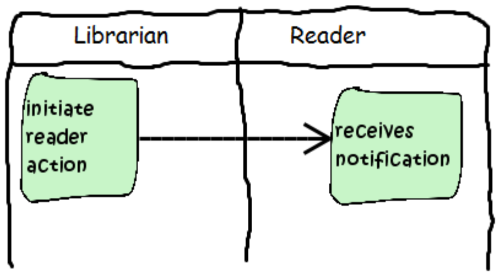


Figure 5 Example of Activity Model Fragment

2.4.2 Class Model Fragment

The class model fragment supports the creation of a number of model elements like classes, properties, methods, associations, generalization and to some extent multiplicity. The below example represent a simple class model fragment consisting of class and properties.

Requirement ID	Text
REI1	All actions are recorded with the action type, a time stamp, the person issuing the action and changes to the balance and account status.

Model Fragment:

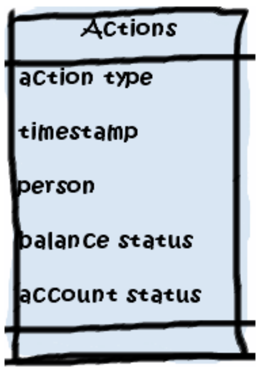


Figure 6 Example of Class Model Fragment

2.4.3 Interaction Model Fragment

The interaction/sequence model fragment consists of the model elements life lines, message transfer and time constraints. Below is shown an example of a requirement generating an interaction model fragment.

Requirement ID	Text
RET2	If the user returns an overdue copy to the library, the LMS displays the overdue days are to the user within 2s.

Model Fragment:

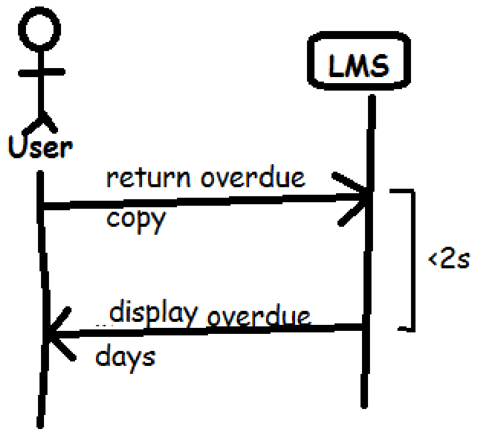


Figure 7 Example of Interaction Model Fragment

2.4.4 State machine Model Fragment

The model fragment of the type state machine consists of the following elements; states, state transition and guards. The example given below illustrates a state machine model fragment consisting of states and transitions created against the requirement.

Requirement ID	Text
ESM5	The expert search queries can be saved, then loaded and executed later to perform various actions of them.

Model Fragment:

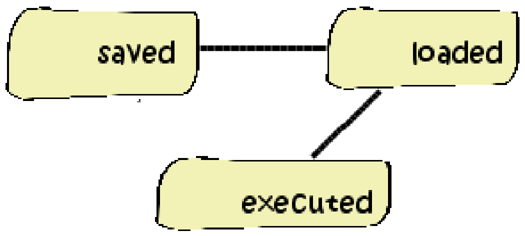


Figure 8 Example of State machine Model Fragment

2.4.5 Use case Model Fragment:

The use case model fragment consists of the model elements actor, use case, system boundary, association, extends and includes relations. An example of a use case model fragment created for a requirement is given below

Requirement ID	Text
MLC2	Librarians may add, update, and delete corpus items manually.

Model Fragment:

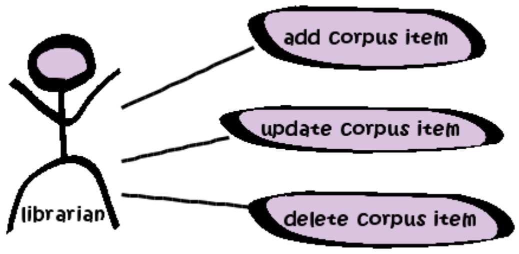


Figure 9 Example of Use case Model Fragment

2.5 Applying Rules to derive Model Fragments from Textual Requirements

This section describes how to apply the rules to create model fragments from textual requirements. The complete rules along with examples can be found in Appendix A. All the examples used to illustrate the rules have been taken from the LMS case study.

The rules are to be used in a top down manner meaning given the requirement, the type of the model fragment to create shall be selected first (rules 1-5). This allows the creation of multiple model fragments against a single rule. Afterwards depending on the model fragment type the sub rules shall be applied to create the model elements.

Let us consider an example to illustrate this process.

Requirement ID	**Text**
WEB5	An online registration guide might be used by the reader to set up a new account

Given the requirement, we apply the rules starting from rule 1

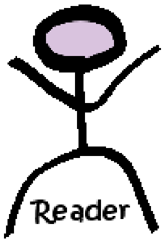
1. **If a requirement specifies a role with activities, create a use case diagram.**

The given requirement does specify a role and some activities hence a use case model fragment is to be created. Now apply the sub rules (1a-1f) of 1 to create the model elements.

1a Subjects and nouns indicated actors or components.

Example

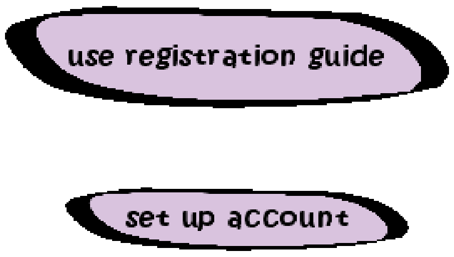
An online registration guide might be used by the reader to set up a new account



1b Predicates (strong verbs) indicate use cases.

Example

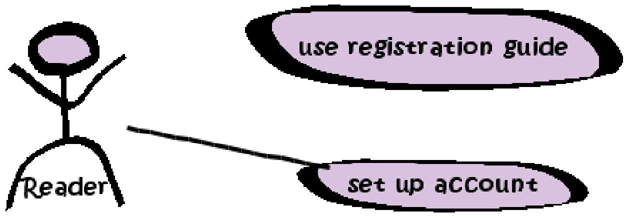
An online registration guide might be used by the reader to set up an account.



1c If a predicate relates to a subject, the corresponding actor and use case shall be connected by an association.

Example

An online registration guide might be used by the reader to set up an account.



1d Prepositions indicating location or ownership ("in", "within") indicate the container of use cases

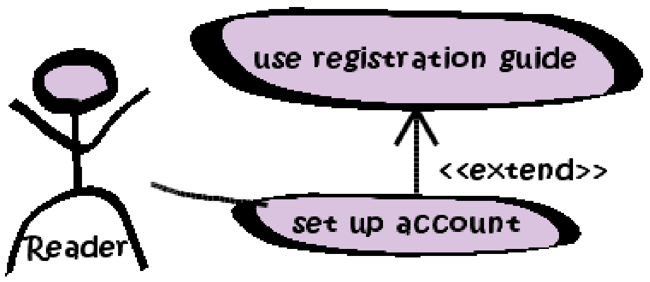
1e Relational prepositions ("from", "to", "via", "by") indicate relationships between use cases.

Rules 1d and 1e are not applicable in this example, hence they can be ignored.

1f Unrestricted relations ("might", "may", "extends") indicate "extends" relations.

Example

An online registration guide might be used by the reader to set up an account.



In a similar fashion first the model type is selected and then the corresponding sub rules are applied to generate model fragments for state machine, interaction, activity and class model fragments. Examples of which can be found in Appendix A.

Chapter 3 Case study

3.1 Research Question

The previous chapters indulged into the concepts of what model fragments are, how they can be used in transforming textual requirements into formal models, and how to create model fragments using the transformation rules. In this section, a case study will be considered to answer the following research questions.

RQ_1: Can model fragments be created for a large set of requirements?

RQ_2: What are the various empirical insights to be collected from creating model fragments to a case study?

RQ_3: What is the pushback of model fragment creation on the requirements?

3.2 Case Selection

The Library management System (LMS) (2011) designed by Prof. Dr. Harald Störrle and used in the course 02264 “Requirements Engineering” has been selected as a case study sample for this thesis. LMS is a state of the art information system for a library, which offers a number of facilities to both the users and staff of the library. The LMS provides the users with a self-service kiosk and online access in addition to many other facilities.

The System Specification (SS) document for LMS was created in Sep 2010 and last updated in Nov 2011. The SS document for LMS as any other SS document describes various aspects of the developing system, including the classification of the stakeholders to determine their importance to the system. This could potentially influence the process of adding model fragments to requirements, as one could be tempted to add fragments to the requirements concerning the more important stakeholder and neglecting the less important ones. Hence to avoid this, the author has only considered the requirements specifications section of the SS document of LMS.

The reason for selecting LMS is that as this thesis focuses on an empirical research for academic students and lecturers, thus a case study familiar to both the author and the supervisor is considered. The LMS example has been used in different context by different people over a long period of time, hence it is time tested. The LMS is a fairly large system with substantial amount of requirements stated in its specification document, thus providing the author with well documented requirements.

3.3 Data Collection Procedure

In this section the process for collecting and recording data has been discussed. For the case study the model fragments to add to the pre-stated requirements specified in the SS document of LMS are the data points. Initially out of the 178 requirements, 24 requirements already had model fragment associated with them.

In the beginning a single model fragment was developed against a requirement intuitively by the author. The existing model fragments were also examined and modified where the author saw fit. A model fragment was drawn on a piece of paper and then added to the already existing RED file (2013) of the LMS. This was a very time consuming process as it required drawing of the model fragments twice (once on paper and then in the RED tool. Also the model fragments created varied greatly in size and range. Thus some kind of framework or rules to develop model fragments in a systematic way was needed. After the rules were formalized the author started adding model fragments to the requirements from the top again, modifying the already created model fragments and requirements to better fit the rules. In addition to this it was established to create as many model fragments against a single requirement as possible.

For selecting a tool to record the model fragments the following properties had to be taken into consideration

- The tool must provide hand-drawn sketchy effect.
- The tool must be easy to use with user friendly interface.
- The time required in learning the tool and drawing model fragments with the tool should be low.
- The tool should support drawing all the major model types.

Table 2 provides an evaluation of the different tools based on the author’s judgment after using the various tools.

Table 2 Evaluation of Different Tools for Recording Model Fragments

Tool	Effort	Cost	Sketching Effect	Time	Diagrams Supported
MagicDraw	High		No	High	All
RED	Medium	Free	Yes	Medium	Not all but major ones
Yuml	Low	Free	Yes	Low	3
Paper & Pen	Low	Free	Yes	Low	All
Other drawing tools	Medium	Free	No	Medium	All

From Table 2 it is apparent that the Paper and pen is the best option to record the model fragments against the various requirements. The model fragments are drawn by hands thus giving the hand-sketch effect that is desired. This also allows for the model fragments to be easily modified and all the various kinds of models can be drawn on paper. However when it comes to adding model fragments to the report the best option would be RED or paint, because they are neater and better looking than the hand drawn ones.

After model fragments have been added and recorded, the different quantitative attributes like size, requirement type, model type were recorded in Microsoft Excel so as to perform various statistical analysis and get results.

3.4 Results

This section includes the various empirical results obtained after adding model fragments to the requirements in the LMS case study in order to get some answers for RQ_1 and RQ_2. From this point onwards the remaining of the thesis document may include abbreviations like id, cd, ud, ad and sd that stands for interaction (sequence) diagram, class diagram, use case diagram, activity diagram and state machine diagram respectively. The data analysis and various illustrations have been done using Microsoft Excel.

3.4.1 Distribution of various requirement types

The requirements stated in the System Specification document of the LMs can be classified in to different requirement types. Given that the author has a good understanding of the different requirement types, the author found that the 178 total requirements can be distributed as shown below in Figure 10. The figure shows that a major portion (73%) of the requirements is categorized as functional requirements while the remaining 27% make up for the quality attributes. The quality attributes have been further classified as security, performance, usability, maintainability and non-functional requirements.

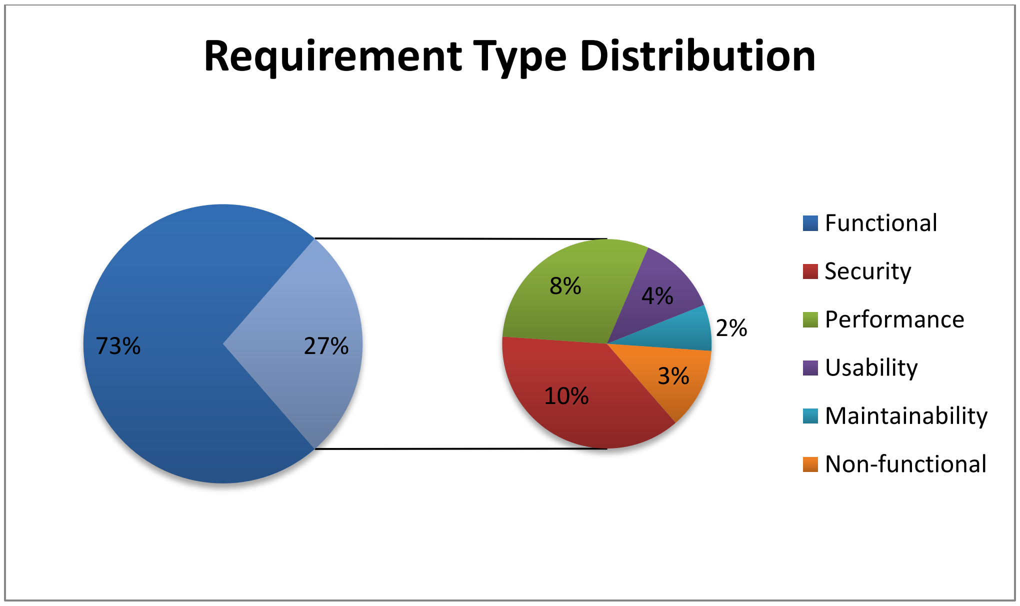


Figure 10 Requirement Type Distribution (Case Study)

3.4.2 Distribution of various type of model fragments

As many model fragments as possible have been created against each requirement according to the rules specified in Appendix A. The model fragments could be of the following types class, interaction, state machine, use case and activity model fragments. In total 203 model fragments were created against the 178 requirements. The distribution of the different type of model fragments is shown below in Figure 11.

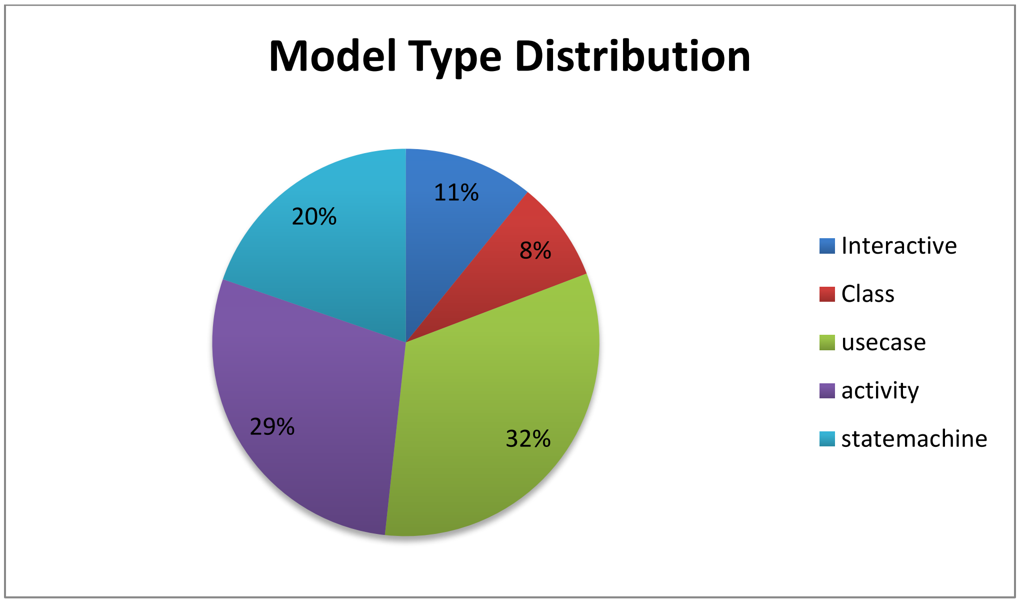


Figure 11 Model Type Distribution (Case Study)

On the basis of this figure, use case model fragments have the highest percentage (32%) followed by activity model fragments at 29%. The remaining three model fragment types make up for 39% of the entire pie chart.

3.4.3 Distribution of number of model fragments per requirement

The objective of the author is to create as many model fragments as possible against a requirement. The reason being that the next step after fragment addition (towards the automation of requirements into formal models) is the weaving of these smaller fragments into larger models and eventually into formal models (future work). It is expected that the more model fragments we have the better the result of the merge will be.

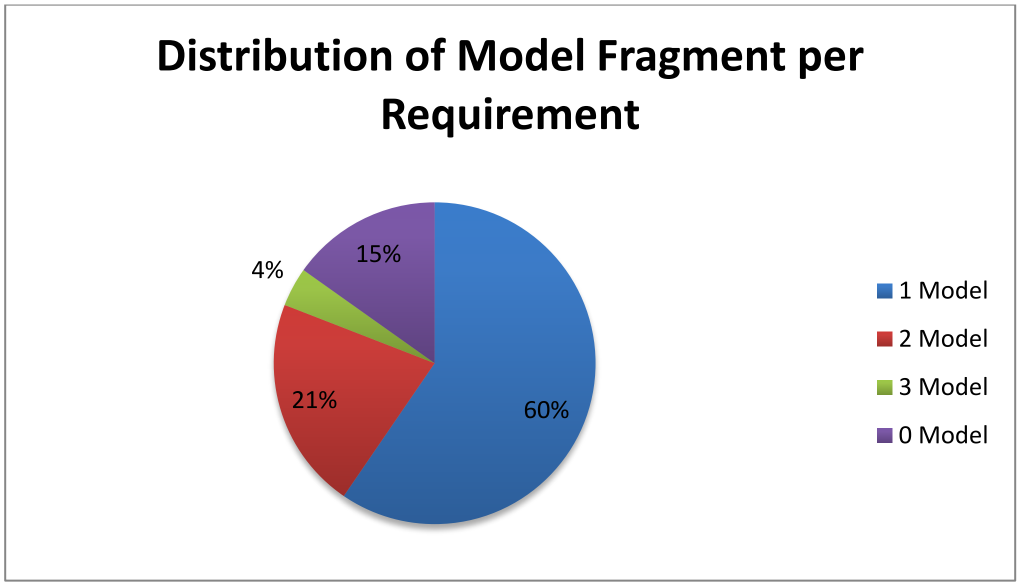


Figure 12 Distribution of Model Fragments per requirement (Case Study)

Out of the 178 requirements only 4% requirements have 3 different model fragments created against them. 15% of the requirements did not result in any model fragment. However a major portion (60%) of the requirements resulted in having a single model fragment. For requirements with 2 model fragments, the Table 3 below represents the requirement count in the various model fragment combinations.

Table 3 Requirement count for the various Model Fragment combinations

	id	**cd**	**ud**	**ad**	**sd**
id	0	0	2	5	1
cd	0	0	0	0	0
ud	2	0	0	10	4
ad	5	0	10	0	16
sd	1	0	4	16	0

The Table 3 shows that out of the 38 requirements (with 2 model fragments), the highest number of requirements fall in the combination ad & sd and ad & ud, with 16 and 10 requirements in each combination respectively. The other combinations i.e. (id, cd), (ud, sd), (id, ud) and (id, sd) land 5, 4, 2 and 1 requirement respectively.

3.4.4 Frequency distribution of model fragment size

The fragment size corresponds to the number of elements in a model fragment. The different types of model elements accounted for are actors, use cases, associations, system boundary, extend/include relations, class, properties, life line, messages, swim lanes, decision nodes, time constraints, objects, activities, methods, guards, states, generalization and relations. Figure 13 represents the frequency distribution of the model fragments on the basis of the fragment size. Most of the model fragments conform to the fragment sizes between 3 and 7. The mean for the fragments sizes of all the model fragments is 5.71 with a standard deviation of 3.71.

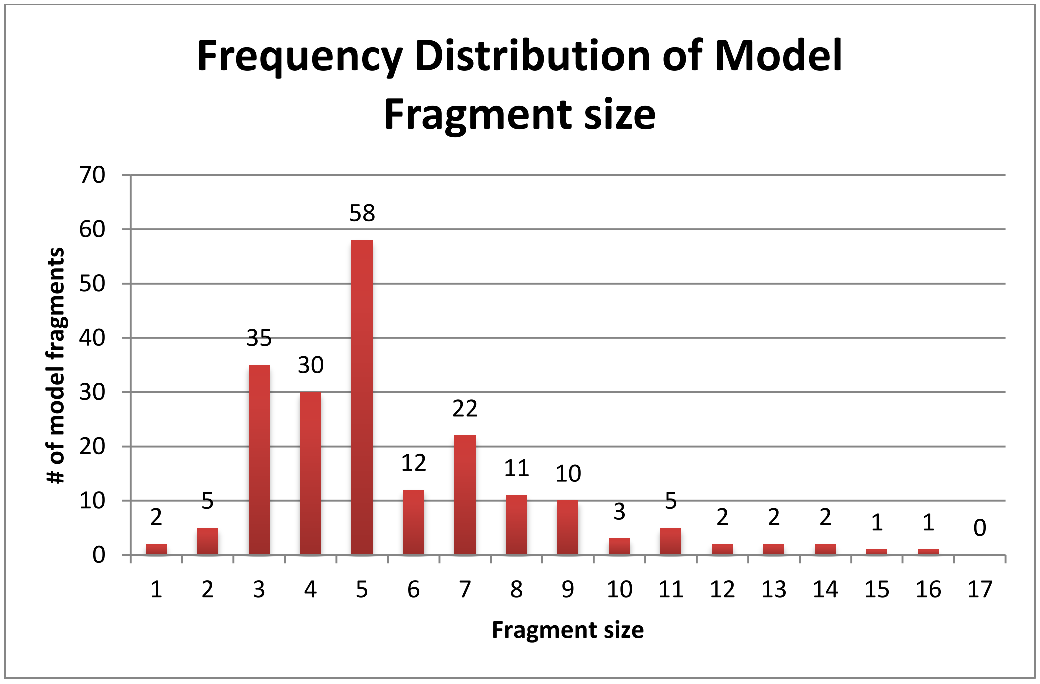


Figure 13 Frequency Distribution of Model Fragment size (Case Study)

3.4.5 Distinction in model types with respect to fragment size.

The previous section describes the frequency distribution of model fragments with respect to fragment size over the entire set of model fragments. In this section the model size of the different model fragment types will be discussed. Table 4 represents the different statistical information about the different model types. From the Table 4 and Figure 14 it can be observed that the different model fragments have not a very high variance with the exception of class model.

Table 4 Statistical Information for the Different Model Fragment Types

Groups	Count	St.dev	Average	Variance
id	21	1.961535	5.380952381	3.847619048
cd	17	6.290165	7.235294118	39.56617647
ud	66	2.374937	4.742424242	5.64032634
ad	58	2.741759	7.517241379	7.517241379
sd	40	1.208676	4.225	1.460897436

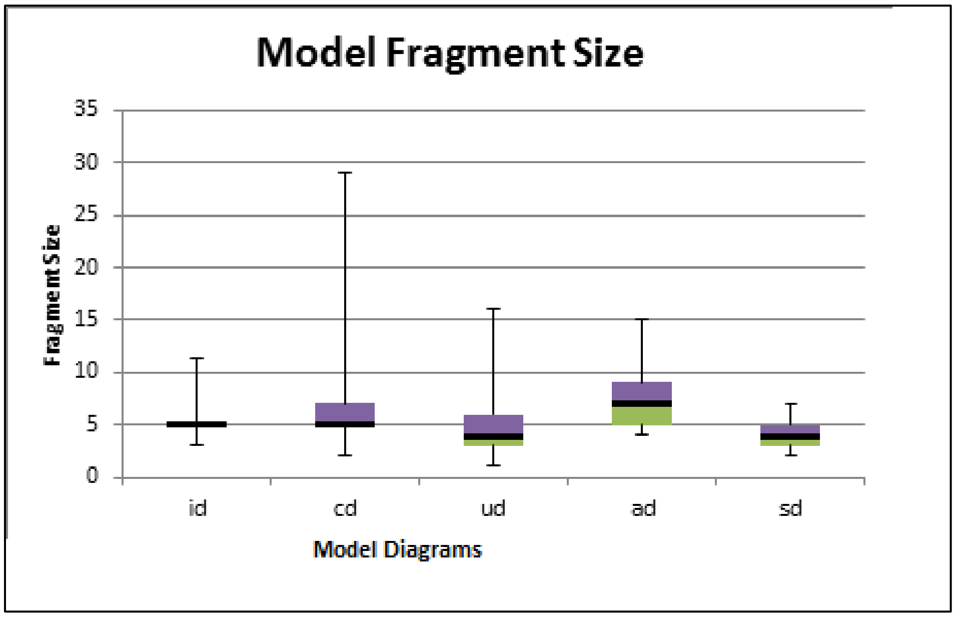


Figure 14 Model Fragment Sizes of the Different Model Fragment Types

Let us elaborate one of the box plots shown in Figure 14. The minimum size of a model fragment of the type ud is observed to be 1 (shown by the bottom whisker). The area of the box corresponds to the 25th, median (black horizontal line in the middle of the box) and the 75th percentile. The maximum size of a model fragment of the type ud is observed to be 16 (shown by the top whisker). All the other model fragment types can be explained in a similar way. The summary for Figure 14 in shown in Table 5.

Table 5 Summary Information for Figure 14

	id	cd	ud	ad	sd
Minimum	3	2	1	4	2
25th percentile	5	5	3	5	3
Median	5	5	4	7	4
75th percentile	5	7	6	9	5
Maximum	11	29	16	15	7

3.4.6 Guidelines for reinforcing natural languages

During the fragment formation, the requirements are to some extent modified to better suit a particular set of transformation rules. It is interesting to see this push back from the model fragments to the requirements and to observe whether the modification to a requirement make the requirements to better fit with the guidelines.

When stating the requirements, a requirements engineer can follow a set of guidelines to help them reduce the risks of misinterpreting the requirements. The requirements following the guidelines do not ensure the deliverance of a better or effective product; rather it helps in reducing the probability of delivering a bad one.

The guidelines for reinforcing the requirements expressed in natural language to be observed in this thesis are

Table 6 Guidelines for Reinforcing Requirements in Natural Language

Guideline1	Use active tense only.
Guideline2	Avoid empty verbs; empty verbs function as a predicate of a clause together with a noun, usually expressing a state or change of state.
Guideline3	Avoid incomplete verb forms; many verbs have implicit references to various other objects. These objects can be detected by asking who, when, what, how.
Guideline4	Avoid negation wherever possible and avoid double negation completely.
Guideline5	Comparisons and all kinds of quantifications should be made complete and precise. There should also be a consistent usage of keywords.
Guideline6	Explicitly state all branches of a condition. This commonly leads to the splitting up of a requirement.
Guideline7	Use definite articles instead of indefinite articles. Do not let political correctness get in your way and prefer singular, use plural only if it is essential.
Guideline8	Universal quantifiers should be used when intended.
Guideline9	Move conditions to the front.
Guideline10	Avoid complex terms and expressions to reduce complexity.

These guidelines have been used to check the quality of the requirements before and after the model fragment creation. Each individual requirement has been accessed before and after the model transformation, to record which of these 10 guidelines are violated by them. Before modifying the requirements for the fragment creation, only 30% of the requirements completely followed all the guidelines. After modifying 58% of the requirements complied with all the guidelines of reinforcing the natural language. This can be seen in Figure 15.

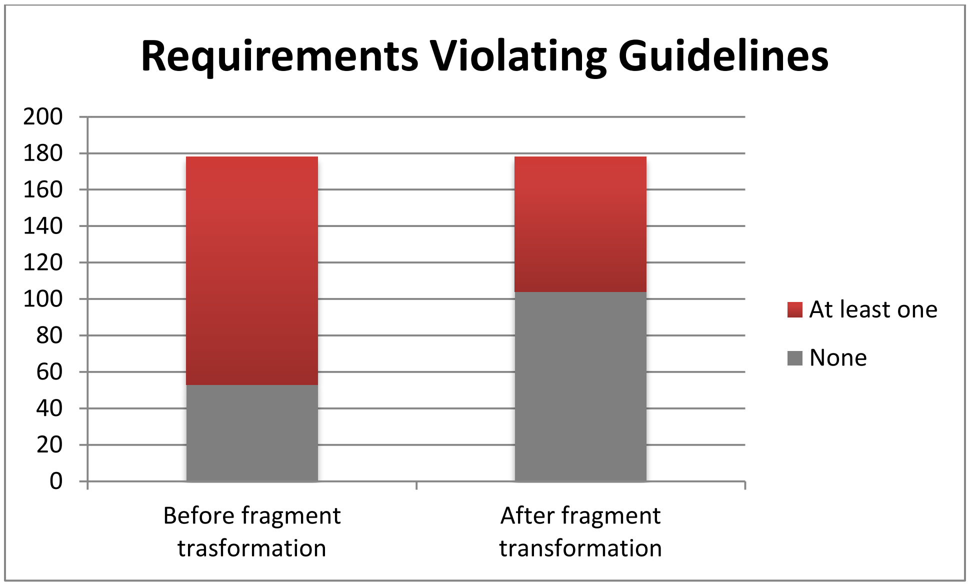


Figure 15 Number of Requirements Violating Guidelines

3.4.7 Individual guidelines assessment

In the previous section the push back of model fragment creation on the textual requirements collectively was observed. Here in this section the individual guidelines will be accessed to observe which guidelines are addressed by modifying the requirements.

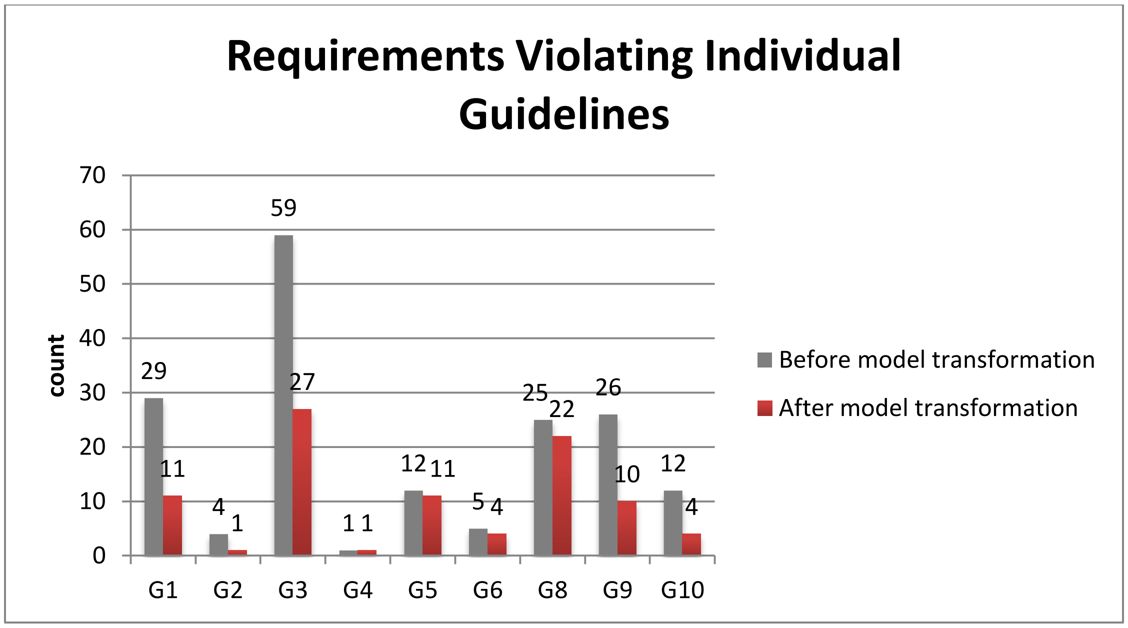


Figure 16 Number of Requirements Violating the Individual Guidelines

The Figure 16 shows the number of requirements violating a particular guideline before and after modifying the requirements for model fragment creation. The graph can be read as initially 59 requirements were violating the guideline G3 and after the requirements were modified due to the fragment creation rules 27 of the requirements were violating the guideline G3. Guideline “R6” has been excluded as no requirement was found violating this guideline before or after the modification. From the Figure 16 above it is visible that the requirements violating the different guidelines vary differently before and after the translation rules have been applied to the requirements. For example the modified requirements do not address the guidelines G5, G7 and G8 and hence there is a very small difference in the before and after count, whereas guidelines G1, G2, G3, G9 and G10 are shown to have a larger difference in the before and after count.

3.5 Interpretation

3.5.1 Distribution of various requirement types

The high percentage of functional requirements as compared to the quality requirements (see Figure 10) can be contributed to the fact that the given LMS is an application system because of which most of its requirements specified in the Specification document would reflect functional aspects of the LMS. Also the developer of the LMS specification document might have focused on just stating the functional requirements of the system. During the process of identifying the type of the various requirements, the author observed that 16% of the total requirements can be specified as having two different requirement types. For example consider the requirement **WEB1c** which states;

> "*When a user enters the login/password in to the WebAccess, a one-time-tag is sent to the user's mobile phone within 10sec*".

This requirement can be classified as both functional and performance requirement. It is desired to have a 1-1 relation between the requirement text and requirement type, because it forces the designer to state every requirement explicitly and not overlook otherwise trivial requirements. Thus WEB1c can be split into two requirements WEB1'c and WEB1'd which states;

> "*When a user enters the login/password in to the WebAccess, a one-time-tag is sent to the user's mobile phone*" and "*Process WEB1'c should take less than 10 sec*" respectively.

By doing so, the functional requirement is made separate from that of the performance aspect. Under normal circumstances this is a better solution as it leads to more concrete and unambiguous requirements. But In case of model fragment generation this is not suitable, firstly because according to the translation rules (Appendix A) where the split functional requirement (WEB1'c) can stand alone and generate fragment, no model fragment can be generated against WEB1'd, which would lead to fewer fragments. One could argue that the rules for model transformation are not correct because of which such a problem arises, but we are assuming that these rules are the best possible model transformation rules out there. A second reason for not splitting requirement WEB1c is that by doing so we explicitly state that WEB1'd is dependent on WEB1'c. Dependency has not been dealt with in the transformation rules. And the requirements are tried to make as independent of each other as possible.

3.5.2 Distribution of various model fragments

The distribution of the different model fragment types (see Figure 11) are mapped with the findings of Dobing and Parsons (2006) which can be viewed in below.

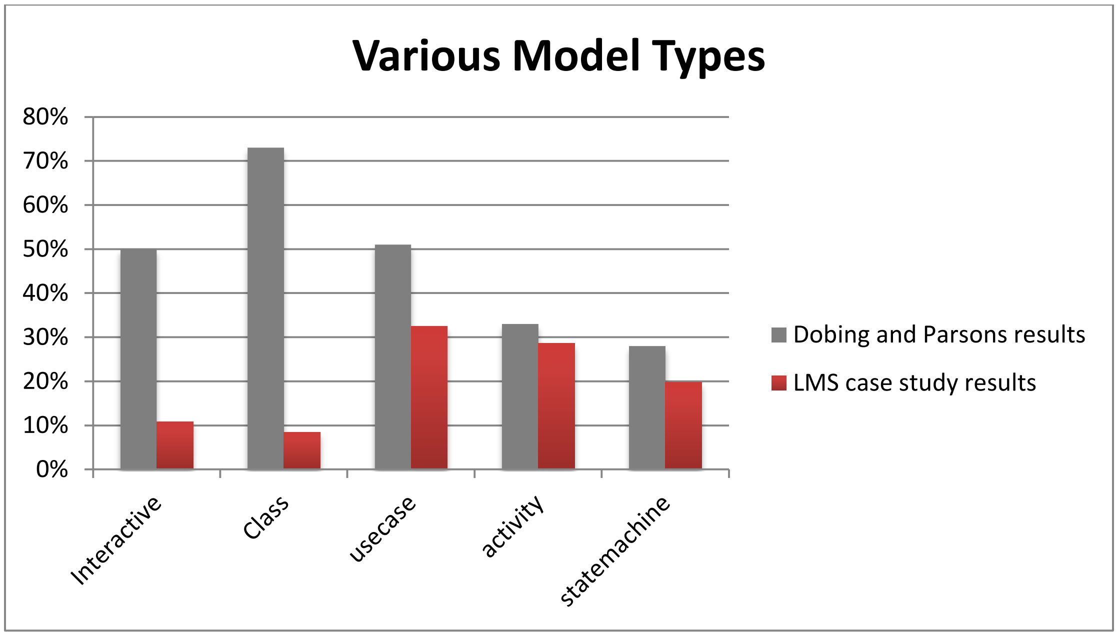


Figure 17 LMS case study results with Dobing & Parsons results, Source (Dobing & Parsons, 2006)

The Figure 17 above maps the usage percentage of the various model types collected by 182 respondents by Dobing and Parsons (2006) and the percentage of the various model types generated by the author for LMS. From the Figure 17 a visible difference is observed in class and interaction (sequence) models. This can be contributed to the fact that a major portion of the LMS requirements represents the behavior aspect of the system. The static relationship between the various concepts of the system was not represented in great detail. Thus the percentage of class diagram is less. For requirements generating 2 model fragments the highest requirement count was found for ad&sd and ad&ud combinations (see Table 3). Use case models describe the functional requirements of a system whereas an activity diagram is often used to elaborate a use case. Thus an activity diagram often is associated with a use case diagram. A similar result is viewed from Table 3 which depicts the highest requirement count for activity and use case combination. Both activity and state machine models depict the behavioral aspects of a system. It is observed that no requirement generate model fragments in the combination of class diagram with any of the other models (id, ad, sd or ud). This indicates that either the requirements worked upon does not include such a requirement or that the static structure models cannot be created with behavioral models.

3.5.3 Relationship between model fragment type and requirement type

It is interesting to see whether there exists a relation between the model fragment type and requirement type. To find out if the model type and requirement type are dependent on each other, a chi square test is performed. The chi test considers the expected and observed counts and has the following formula

$$X^2 = \sum \frac{(O - E)^2}{E}$$

Where O is the observed count and E is the expected count. The hypothesis for the chi square test can be defined as

$$H_0: There\ is\ no\ relation\ between\ the\ model\ fragment\ type\ and\ requirement\ type.$$

$$H_1: There\ exists\ a\ relation\ between\ the\ model\ fragment\ type\ and\ requirement\ type$$

Figure 10 shows the different requirements types the LMS requirements can be classified into. But instead of having all the different types, only two broad classifications of the requirements will be considered namely functional and quality requirements. The reason for merging the requirement types into these two main categories is that there would otherwise be too few observations (low counts) for the test to be reliable (it is an approximate test and X^2 is only properly chi-square distributed if most of the expected counts (E in the formula above) are not too low (one rule of thumb is: no more than 20% less than 5). This will make the calculations easier and also will not neglect the ones with low values. Thus the count of the various model fragment types against the functional and quality requirements have been observed and makes up for our observed count that can be seen in Table 7.

Table 7 Observed Count of the Different Model Diagrams against Functional and Quality Requirements

Observed count	id	cd	ud	ad	sd
Functional	16	17	62	57	40
Quality	15	2	14	13	10

The expected count for the given observed data has been calculated and shown in Table 8.

Table 8 Expected Count of the Different Model Diagrams against Functional and Quality Requirements

Expected Values	id	cd	ud	ad	sd
Functional	24.19512	14.82927	59.31707	54.63415	39.02439024
Quality	6.804878	4.170732	16.68293	15.36585	10.97560976

With the above observed and expected count and α=0.05, the p-value for the chi-square test turned out to be 0.00426. As the chi (p-value) < α, hence the H_0 can be rejected, implying that there exists a relationship between the model fragment type and requirement type.

Table 9 Percentage Deviation Values between the Expected and Observed values

Percentage Deviation	id	cd	ud	ad	sd
Functional	-34%	15%	5%	4%	3%
Quality	120%	-52%	-16%	-15%	-9%

The percentage deviation between the observed and expected count gives us an idea about this relationship. Table 9 represents the percentage deviation of the observed and expected values for the various model types and requirement types. According to this table the probability of having cd, ud, ad and sd against functional requirement is higher than that of having interaction diagrams against functional requirements. But interaction diagrams shows to have the highest probability against quality requirements compared to the other models. From these findings we can formulate the following research question

Q_1: *"Which type of model fragment is created against which requirement type?"*

3.5.4 Requirements with no model fragments

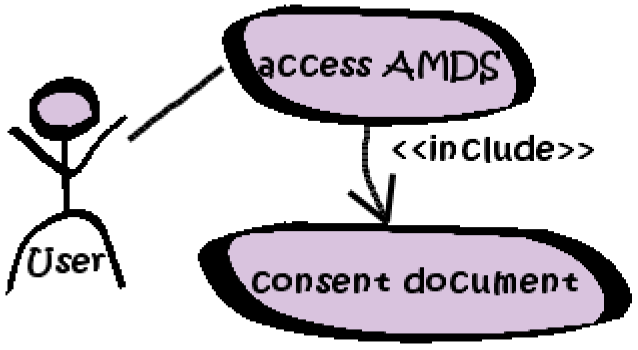
From Figure 12 it is noted that 60% of the requirements have 1 model fragment, 21% have 2 model fragments and only 4% of the requirements have 3 model fragments. The remaining 15% do not have any type of model fragments at all. It will be interesting to know which types of requirements do not create model fragments and whether a requirement of a particular type will never have model fragments against it. On closer inspection it is found that 22% of the requirements with no model fragment were security requirements. Example of a security requirement with no model fragments is given below.

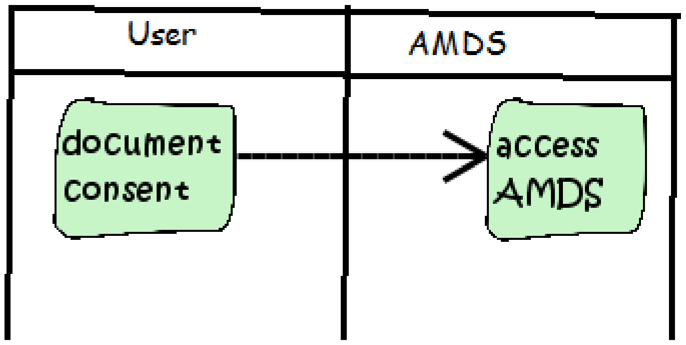
"All passwords must be strong."

This requirement explains that the passwords used for logins/authentications should be strong, so as to prevent reader's profile from being misused. Such a requirement cannot generate model fragments according to any of the transformation rules stated in the Appendix A. 22% of the requirements with no model fragments were that of usability requirements. An example of which is given below

"All functions of the LMS-GUI shall be accessible by keyboard."

This infers that not all types of requirements can have model fragment against them. However this does not mean that requirements of a particular type can never have model fragments against it. This can be justified by considering another security requirement from the LMS.

Requirement ID	**Text + Model Fragment**
AMDS3	Users of the AMDS must document consent in order to access the AMDS
	



The above given example talks about no unauthorized personal accessing the AMDS, hence relates to the security of the system. Where the first security requirement (*"All passwords must be strong"*) did not result in any model fragment it can be seen that this second example (*"Users of the AMDS must document consent in order to access the AMDS"*) has not a single but two different types of model fragments created against it. It might also be possible that the transformation rules for creating model fragments against requirements are not as effective as we thought.

3.5.5 Model fragment size

From the frequency distribution graph of model fragment size (see Figure 13), it can be seen that 78% of the model fragments conform to fragment sizes between 3 and 7, which indicates that usually the fragment sizes are small. An anova test is performed on the fragment sizes of the different types of model fragments with the null hypothesis: "*there is no difference in the average fragment sizes amongst the different model fragment types."* With α= 0.05, the p-value for the anova test turned out to be 9.24413E-09, which suggest that the null hypothesis can be rejected.

The variance for the different types of model fragments (see Table 4) does not show large difference with the exception of class diagrams. This can be explained by considering the outliers for the set of data points of the various model fragments. An outlier is a data point that lies below Q_1-(1.5*IQR) or above Q_3 + (1.5*IQR) where Q_1 and Q_3 are the first and third quartile respectively and IQR is the inter quartile range. In addition to this the extreme outlier values have also been calculated. All this information is represented in a modified version of Figure 14, such that the outliers for the various model types are illustrated by a black dot and the extreme outliers by a red dot.

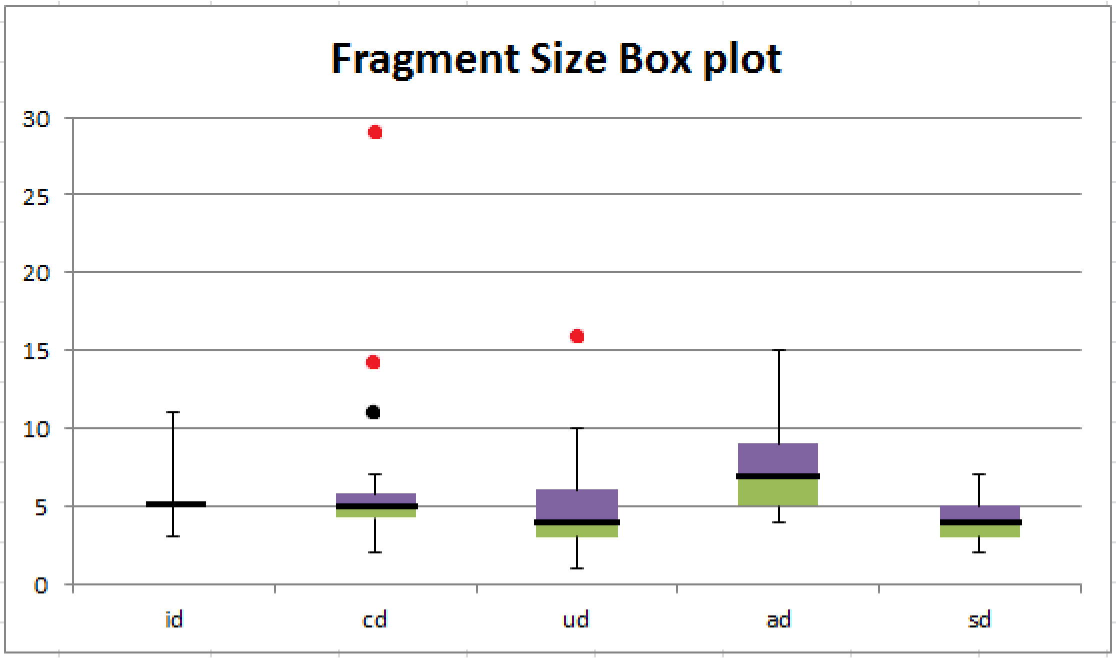


Figure 18 Box plot illustrating the Fragment Sizes of the Various Model Diagrams with their Outliers

By ignoring the outliers, the variance for the different model fragment types is calculated to be not so large. This is represented in Table 10. From these findings another research question can be formulated

Q$_2$: Does a particular model diagram always create model fragments in a specific size range?

Table 10 Variance values for the Different Model Types (excluding the outliers)

Model Fragment Type	Variance
Interaction	3.847619
Class	1.917582
Use case	3.717788
Activity	7.324675
State machine	1.330159

3.5.6 Push back of model creation

One of the research questions RQ$_3$ stated in section 3.1was "what is the push back of model fragment creation on the requirements?". It has already been stated in section 3.4.6 that during the model fragment creation some of the requirements were modified so as to better fit the transformation rules. The requirements have been checked (before and after modification) against 10 guidelines used for reinforcing the requirements in natural language. The LMS case study has 178 requirements in total, 85 out of which have been modified to better suit the fragment formulation rules. And due to this modification a 47% decrease in requirements violating the guidelines is seen (see Figure 19). This means that the model fragment creation rules pushes the requirements to better fit with guidelines.

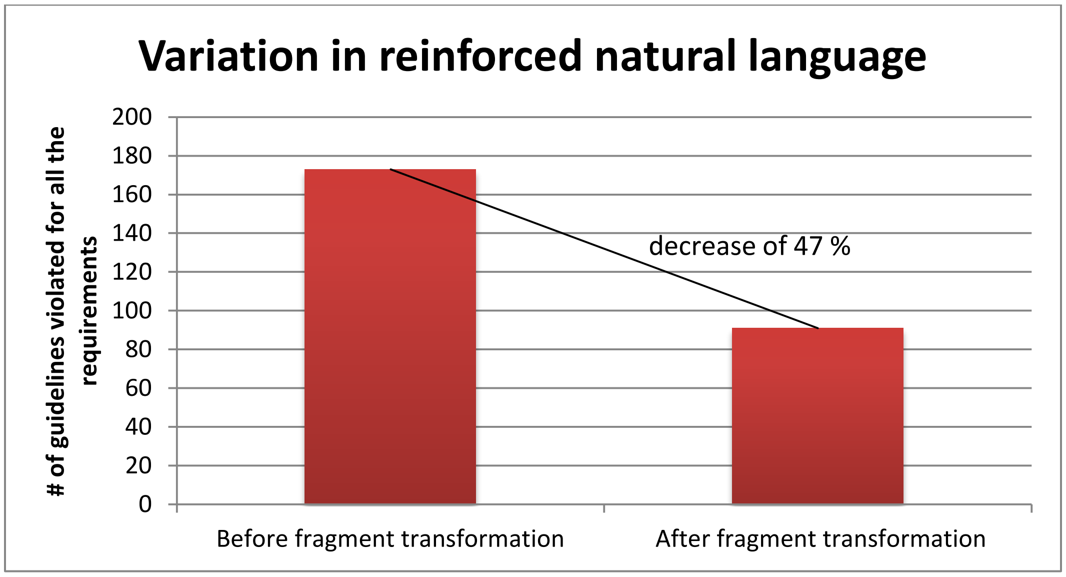


Figure 19 Variation in Reinforced Natural Language Requirements

An example from the LMS case study can be considered to demonstrate how the requirement is modified so that it better fits with a guideline. Let us consider requirement **CDAT6** which states that *“The catalog may be updated by a librarian manually.”* According to the transformation rules, such a requirement would result in the generation of use case model. It was found with a slight modification this requirement could better fit with the rules of use case model fragment. Hence the requirement **CDAT6** was modified to *“A librarian can manually update the catalog.”* It can be seen that the initial requirement (*The catalog may be updated by a librarian manually)* violates **Guideline 1**, which states that the requirements should be in active tense rather than passive. But the modified text i.e. *a librarian can manually update the catalog*, does not violate this rule. This however does not imply that all the requirements violating **Guideline 1** will always be corrected. Let us consider another requirement **LCP8** which says *“Librarians may manually modify the state of a lease, prolongation, or reservation.”* This requirement will result in the generation of a use case fragment without modifying the original text. However it is seen that this requirement violates **Guideline 8.** According to this guideline universal quantifiers should be explicitly stated when intended. But the model fragment rules do not enforce such a change and hence the requirement text remains the same, with **Guideline 8** still being violated.

The percentage change for each of the individual guideline before and after model fragment creation is shown in the Table 11 below. The negative value of the percentage change indicates a decrease in violating guidelines, thus implying an improvement in the requirements.

Table 11 Percentage Change for the Individual Guidelines

	Guidelines	%change
Guideline1	Use Active tense	-62%
Guideline2	Avoid empty words	-75%
Guideline3	Avoid incomplete verb forms	-54%
Guideline4	Avoid negation & double negation	0%
Guideline5	Explicitly state all branches of a condition	-8%
Guideline7	Use definite articles	-20%
Guideline8	Specify Universal quantifiers	-12%
Guideline9	Move conditions to the front	-62%
Guideline10	Avoid complex terms	-67%

The push back of model creation is not only the modification of the textual requirements but it also helps in recognition of the missing trivial requirements. Usually during the requirements gathering phase, the client or the requirement engineer overlooks the trivial requirements and do not include them in the system specification document. This neglect may happen on behalf of either of the parties. These requirements can be trivial but that does not make them any less significant compared to other ones and should therefore be explicitly stated in the specification document of the developing system. The model fragments can help in identifying these trivial requirements. For example the model fragment for a requirement **(RAC10)** *"a reader can sign in, update, reserve medium, prolong medium from a reader account"* will look something like

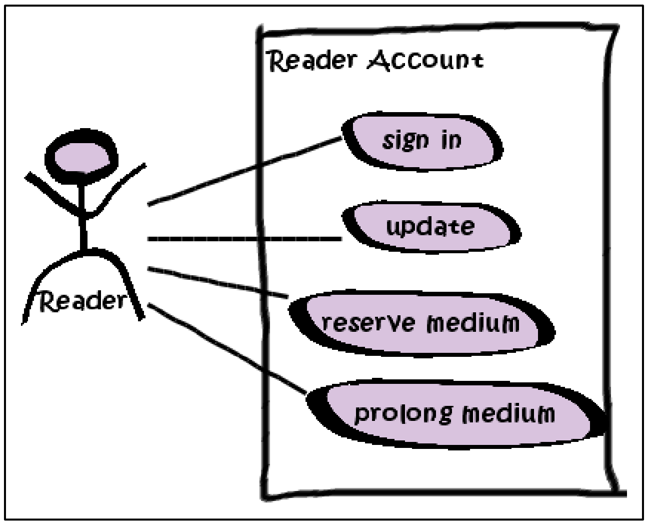


Figure 20 Use case Model Fragment for Requirement RAC10

This model fragment indicates that a reader can sign in to a reader account, update his/her account and send a request for a medium or prolong the lease of a medium. An obvious activity of the reader should be to sign out from the account. This activity is not mentioned in the textual requirement and can be overlooked, but creating the model fragment helps in identifying this otherwise obvious activity and hence leading to the addition of a new requirement stating "*the reader should be able to sign out of the reader account at any given time".*

Chapter 4 Experiment

4.1 Experiment Goal & Hypothesis

The goal for conducting an experiment is to basically confirm that the transformation rules to create model fragments are understandable, applicable and simple for other students to use. This can formally be expressed by using a Goal Question Metric (Koziolek, 2008) format. According to which the goal of the experiment can be stated as

Table 12 Goal of the Experiment by Using Goal Question Metric Approach

Purpose	To validate
Issue	The transformation rules
Object	Of requirement specified in natural language to model fragments
Viewpoint	From researcher viewpoint.

In addition to validating the transformation rules, other empirical insights will also be gathered from the experiment. One research question to answer is

RQ$_4$: Does model fragment size contribute to the effort put in creating a model fragment?

4.2 Methods and Materials

40 students from the 02264 "Requirements Engineering" course taught at DTU Lyngby participated in the experiment. The experiment was conducted at the end of the term and no incentive was given to the students. The students were also informed that their participation in this experiment will have no effect on their final grade. 3 students were female while the remaining 37 students were male. 90% of the participants were up to 30 years of age, 8% students were in the range of 31 to 50 and 1% above the age of 51.

All the participants were asked to rate their different capabilities and skills at the beginning of the experiment from a scale of 0 to 4 (0 being very low and 4 being very high). According to which the average skill level of the participants in English, Requirements Engineering, Unified Modeling Language (UML), practical modeling and computer programming turned out to 3.15, 2.57, 2.39, 2.17 and 2.72 respectively.

The questionnaires included of a textual requirement, blank space (where the students were to add model fragments), two subjective questions and a space to record the time after completing the task (Appendix B) .There were two different types of questionnaires. The layout, basic format and level of the requirements were similar however the content varied. One set of questionnaires had requirements 1-8 and the other set had requirements 9-15 (Appendix B). Half of the students were given the questionnaire with requirements 1-8 in it while the other half were given the questionnaire with requirements 9-15. The distribution of the questionnaire amongst the students was done randomly.

For this experiment each student was provided with an instruction set, stating the rules for creating model fragments from textual requirements. Each student was also provided with a questionnaire stating the requirements against which model fragments were to be created. No electronic device (i.e. computers, tablets or scientific calculators) were required for the experiment. A pen/pencil and open mind on behalf of the students was required.

Before starting the experiment the participants were informed about why the experiment was being conducted and advised to read the instruction sheet carefully as it included crucial information about how to apply the various rules. They were also asked to sign a consent form to allow the researchers to use the findings from the experiment for academic purpose. The students were asked to create model fragments for as many of the requirements as possible with no time limit, but the participants were requested to note down the time in the questionnaire after creating a model fragment for a requirement, before moving on to the next requirement. Not all the participants were able to complete the entire questionnaire, others opted to skip some of the requirements maybe because they found those requirements difficult or confusing. Therefor the requirements against which model fragments were created by the participants are only considered during data analysis. The quantitative data of the experiment was then transcribed by the author into Microsoft Excel.

4.3 Observation

This section includes the observations and results found after conducting the experiment. The various data analysis and illustrations presented in the coming sections are done using Microsoft Excel.

4.3.1 Distribution of model fragment type

Rules for creating model fragments of the type activity, class, interaction, state machine and use case models were given to the participants. Thus the participants were restricted to create model fragment of the above mentioned types. The distribution of the different type of model fragments can be viewed in Figure 21. From the distribution chart it can be seen that activity model fragments has the highest count with a percentage of 34% followed by interaction diagrams with a percentage of 22%. Use case, State machine and Class model fragments are found to be in the percentage 20%, 16% and 8% respectively.

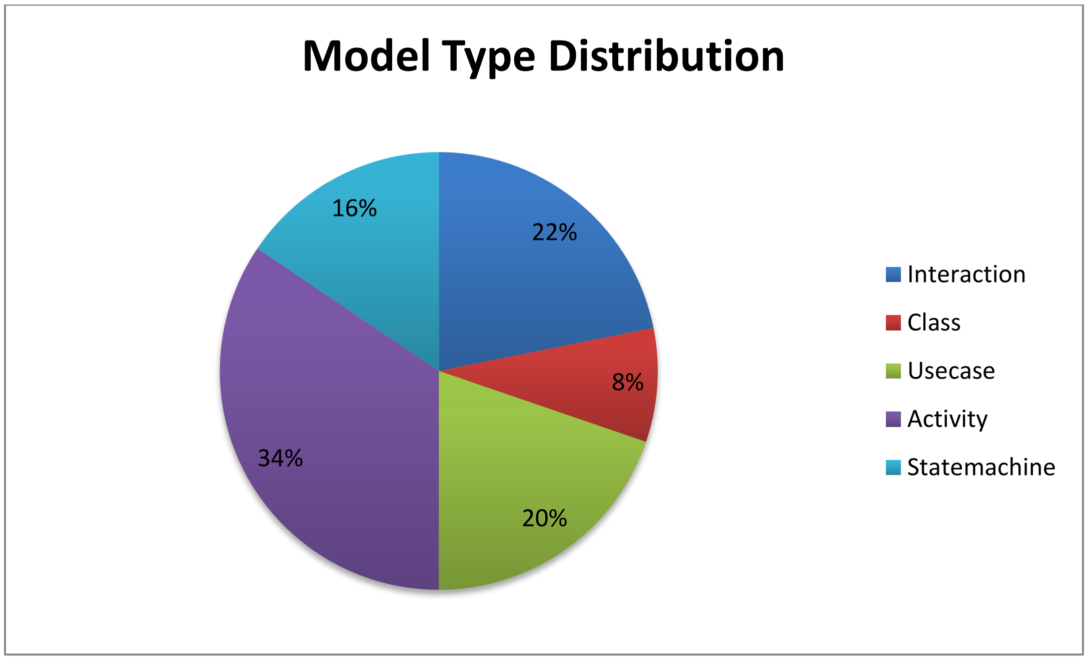


Figure 21 Model Type Distribution (Experiment)

4.3.2 Number of Model Fragments per Requirement

The model fragment generated per requirement is interesting to observe and it can be seen from Figure 22 that 96% of the requirements have only 1 model fragment associated to it. 4%

of the requirements results in the creation of 2 different types of model fragments. No participant generated more than 2 model fragments for a single requirement.

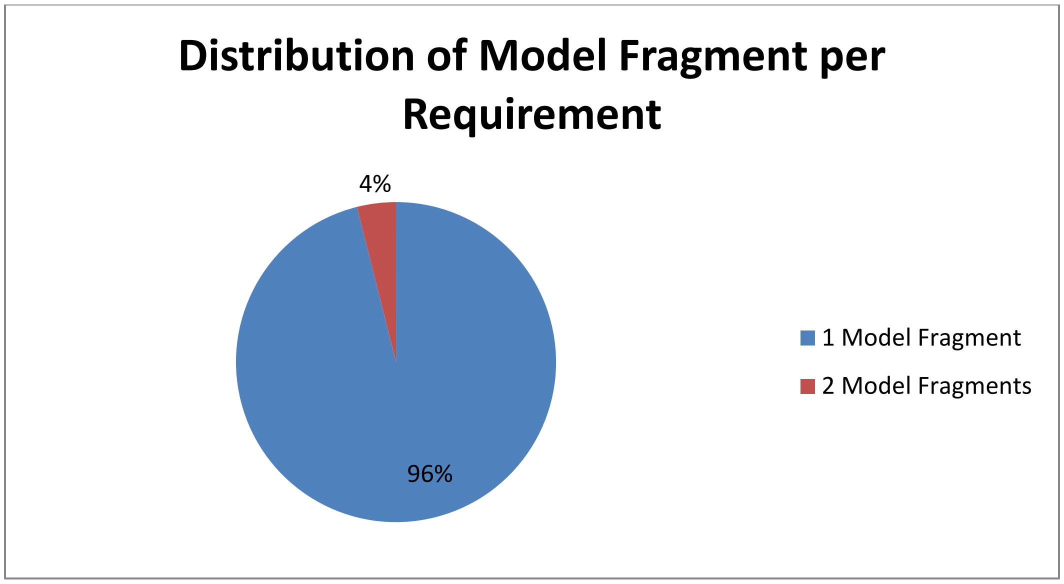


Figure 22 Distribution of Model Fragments per requirement (Experiment)

4.3.3 Model diagrams created against the different requirements

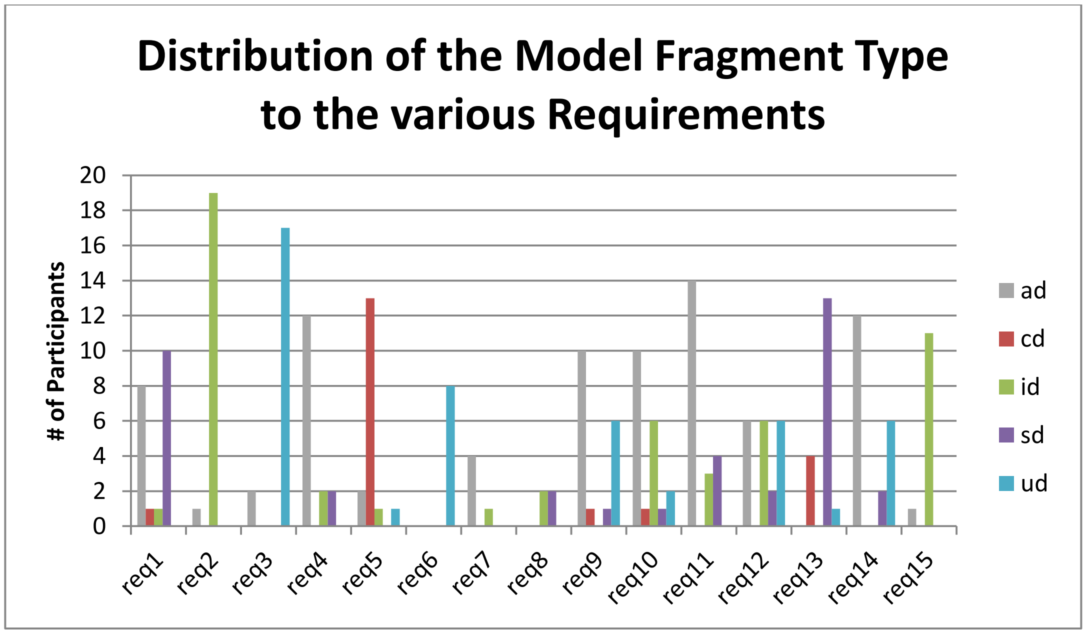


Figure 23 Distribution of the Model Type against the Various Requirements

The Figure 23 represents the distribution of different type of model fragments the participants were able to create against each requirement given in the questionnaire. It should be remembered that a single participant did not create these model fragments against a requirement rather multiple participants worked on the same requirement (independently). For one out of the fifteen requirements (req10) the participants have created all the five types of model diagrams. For four out of fifteen requirements (req1, req5, req9, req12) the participants were able to create four different types of model diagrams. Four out of fifteen requirements (req4, req11, req13 and req14) the participants were able to create three types of model diagrams. Five requirements (req2, req3, req7, req8 and req15) have two different model diagrams created against them. Req6 is the only requirement with a single type of model diagram created against it by the participants.

4.3.4 Model Fragment Size of the different model diagrams

The fragment size corresponds to the number of elements in a model fragment. The fragment size for all the model fragments created by the participants has been manually calculated by the author. The mean model fragment size is 8.7 with a standard deviation of 4.22. Table 13 represents the statistical information for the fragments sizes of the different model diagrams. It can be seen from this table that the variances between the various model diagrams vary slightly. Activity and class diagrams depict the highest variance Figure 24

Table 13 Various Statistical Information for the Different Model Diagrams

Groups	Count	Sum	Average	Variance
Interaction	51	278	5.450980392	1.09254902
Class	20	176	8.8	13.11578947
Use case	47	339	7.212765957	6.997224792
Activity	81	1022	12.61728395	15.16419753
State machine	36	229	6.361111111	7.723015873

From Table 14 it can be seen that the 75th and median values of class and State machine is the same, thus the box plot (Figure 24) only visualizes the median by the horizontal black line. For interaction diagrams the 25th percentile and median has the same values. This is also illustrated in the box plot (Figure 24) as a horizontal black line. The right most box plot labeled “Average” represents the average of the different values (minimum, 25th percentile, median, 75th percentile and maximum) for all the different model diagrams.

Table 14 Summary Table for Figure 24

	Interaction	Class	Use case	Activity	State Machine	Average
Minimum	4	5	3	5	3	4
25th Percentile	5	6.75	5	10	4.75	6.3
Median	5	9	6	12	7	7.8
75th Percentile	6	9	8.5	15	7	9.1
Maximum	9	22	16	27	16	18

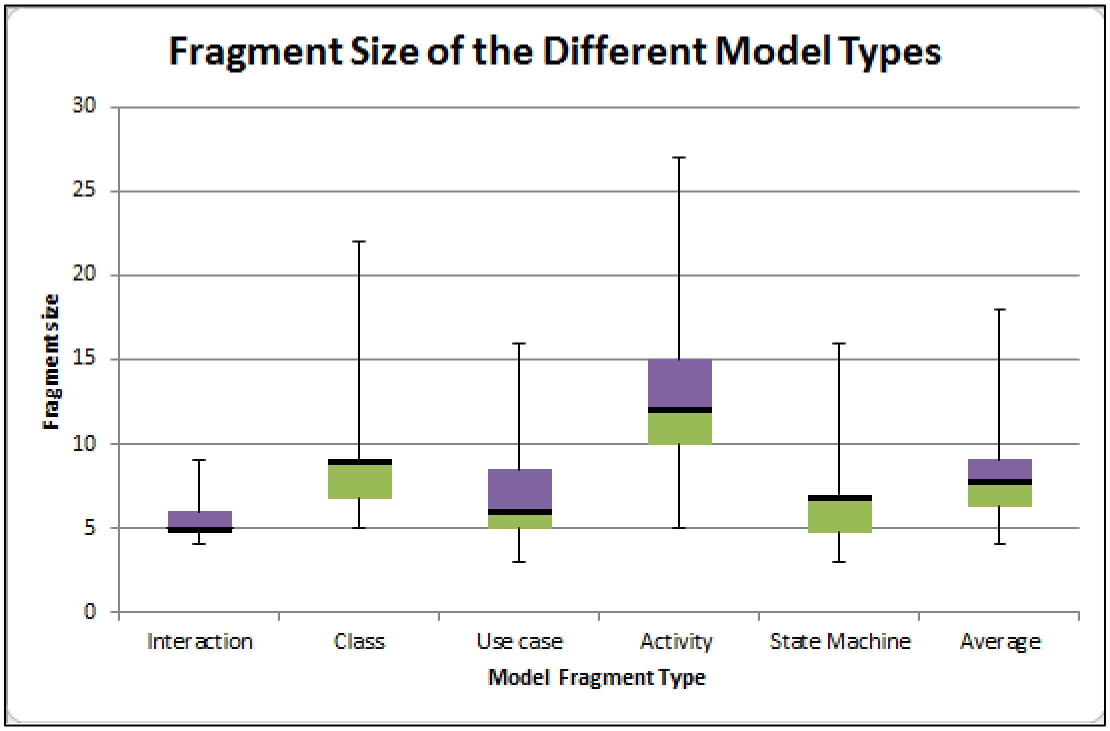


Figure 24 Model Fragment Size for the Different Model Fragment Types (Experiment)

4.3.5 Difficulty level of the different model diagrams

The questionnaire included two questions that were to be answered by the participants; one of the questions was how difficult they found the process of creating that model fragment. They were asked to rate the effort put in creating a model fragment on a scale of 0-4 (0 being easy and 4 being very difficult). With the help of that data, the author has made use of box plot to visualize the effort put in creating the different model fragments according to the participants.

The Table 16 represents a summary of the box plot shown in Figure 25. The variance and other statistical information for the different model diagrams are shown in Table 15 which depicts a low difference in variance.

Table 15 Statistical Information of the Perceived Effort for the Different Model Diagrams

Groups	Count	Sum	Average	Variance
id	50	83	1.656	0.61924898
cd	20	30	1.485	0.676078947
ud	47	76	1.610638298	0.720101758
ad	81	166	2.043209877	0.594234568
sd	35	73	2.071428571	0.537394958

The area of a box plot not only corresponds to the 25th percentile, median and 75th percentile for the difficulty level of a model diagram, but also depicts the number/count of data points for that model diagram.

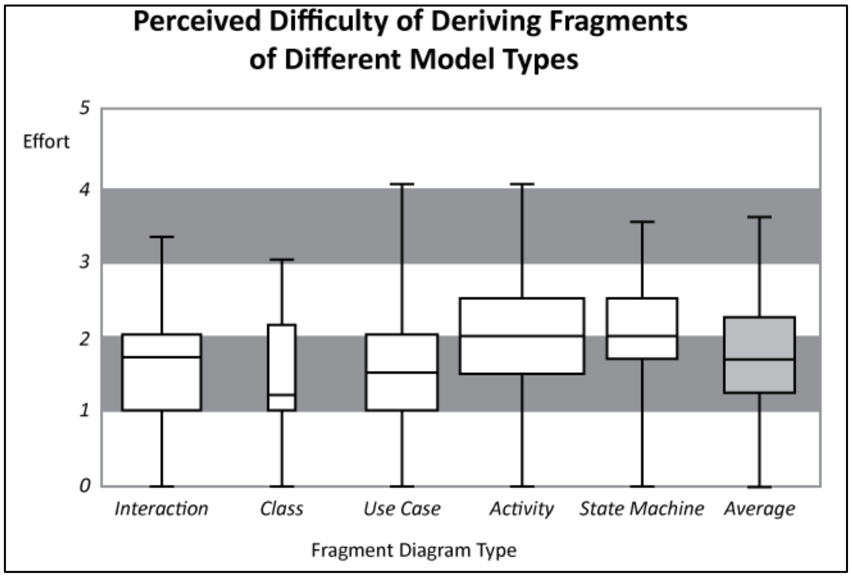


Figure 25 Difficulty Level for the Different Model Fragments (Experiment)

Table 16 Summary Table for Figure 25

	Interaction	Class	Use case	Activity	State machine	Average
minimum	0	0	0	0	0	0
25th percentile	1	1	1	1.5	1.7	1.24
median	1.7	1.2	1.5	2	2	1.68
75th percentile	2	2.125	2	2.5	2.5	2.225
maximum	3.3	3	4	4	3.5	3.56

4.4 Interpretation

4.4.1 Distribution of model fragment type

The high percentage of activity, use case and interaction model fragments suggest one of the two things. Either the transformation rules for these model diagrams are easier to apply or the participants of the experiment are more familiar with these types of model diagrams. Both of these propositions are possible, but given that all the participants of the experiment were students of the course "Requirements Engineering" and had worked with such models throughout the semester, it would be justifiable to accept the latter proposition. The author selected all the requirements to be worked on in the experiment keeping in mind the different model fragments it would generate. Even though the distribution of the model diagrams is not exactly equal it is close to what the author had expected.

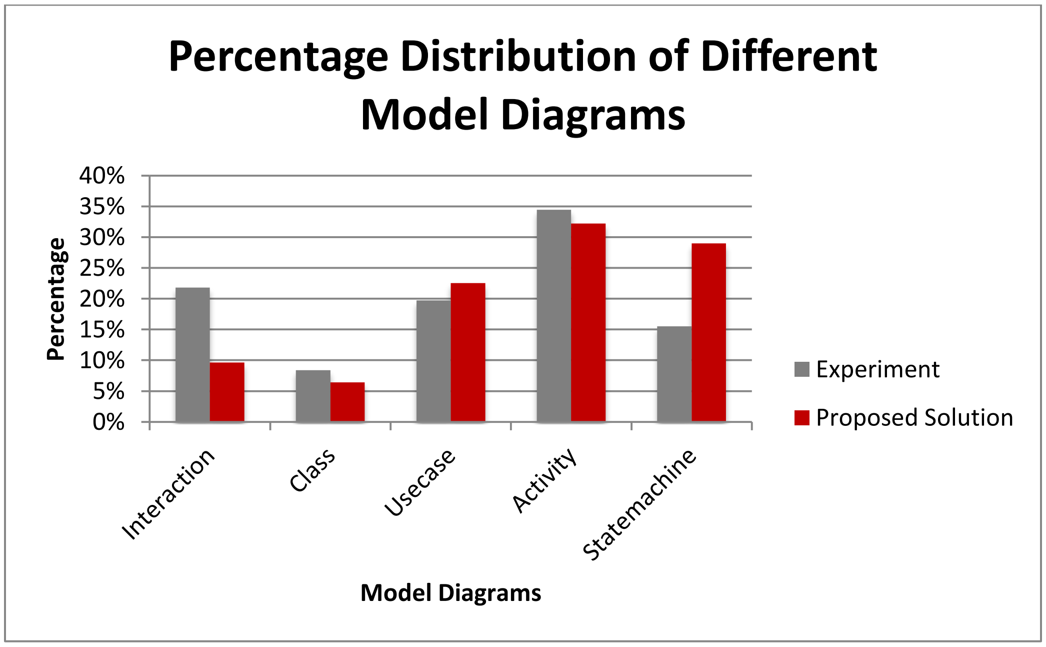


Figure 26 Percentage Distribution of Model Diagrams for Proposed Solution with Percentage Distribution of Model Diagrams for Experiment results

As seen from Figure 22 the participants were only able to create at most 2 model fragments for a single requirement. Also the ration of a single model fragment to two model fragments against a requirement is very high (25:1). The percentage of 2 fragments per requirements is very small almost negligible. This suggests that the students were not able to use the transformation rules correctly. But if we consider the information shown in Figure 23 it depicts that for 53% of the requirements at least three different model diagrams can be created against them. This implies that the fragment creation rules for the different model diagrams are applicable to a single requirement and there must be another reason as to why a single student was not able to create more fragments. One possible explanation as to why 96% of the students created only a single fragment per requirement could be that the students had not read the instruction set carefully and were not aware to create as many fragments as possible for a requirement. Another reason could be that the layout of the questionnaire did not allow for more than one fragment to be drawn against a requirement. The questionnaire for the experiment can be found in Appendix B from where it can be seen that a single empty box is given under a requirement. If the questionnaire had multiple empty boxes under a single requirement, it might have been clearer for the students as to what needs to be done. The third reason could be that the students did not care to perform well and created a single fragment thinking it was enough.

How close were the participant's answers to the solutions?

The author created model fragments for the requirements given in the experiment which will be considered as the correct/proposed solutions. The list of the different type of model fragments created by the author against a requirement can be seen in Table 17. According to this table a requirement can at max have 4 different types of model fragments associated to it and at minimum 1 model fragment.

Table 17 Solution Proposed by the Author

Requirement ID	Fragment Type 1	Fragment Type 2	Fragment Type 3	Fragment Type 4
req1	sd	ad		
req2	ud	sd	id	ad
req3	ud			
req4	sd	ad		
req5	sd	cd		
req6	ud			
req7	sd	ad		
req8	sd	id	ad	

req9	ud		
req10	ud	sd	ad
req11	sd	ad	
req12	ud	ad	
req13	cd		
req14	ud	sd	ad
req15	id	ad	

Now with the proposed solution given, the work of the participants can be evaluated to see how correct they were in creating the right type of model fragment. This will give an idea about how effectively the participants were able to apply the fragment creation rules. For this purpose a correctness score is given to the participants against the individual requirements they have worked on. The correctness score checks the model type of a fragment (created by the participant) against the model type(s) of the proposed solution for a given requirement.

The participant is given a correctness score of 0 if none of the participants' model type matches the model type in the proposed solution. If *n* number of fragments matches, then the correctness score will be *n/k* where n is the number of matched model fragments by the participant and k is the total number model fragments in the proposed solution. The correctness score will be 1 if the model fragment of the participant and proposed solution is exactly the same both in type and number. For example let us consider the **req8** from Table 17. The proposed solution has 3 model fragments of type sd, id and ad (k=3). For the same requirement let us say **Participant 1** creates a model fragment of type cd. This would result in a correctness score of 0. **Participant 2** creates fragments sd and id. As both the model fragments created by **Participant 2** are matched with the proposed solution model diagrams, thus the participant will get a correctness score of 2/3=0.67 against r**eq8**. **Participant 3** creates three model fragments of the type sd, id and ad and hence will have a correctness score of 1. A complete table for the values of correctness score for the participants can be found in Appendix C.

The distribution of the participants on the basis of correctness score recorded is depicted in Figure 27. If we assume correctness score of 0.5 as the threshold value then participants with correctness score of greater than or equal to 0.5 can be gathered into one group and participants with correctness score less than 0.5 into another group. According to this grouping 56% of the participants have been successful in effectively understanding and applying the transformation rules while 44% of the participants have not been quite successful. Such a small difference in percentages suggests that the transformation rules are not as effective as anticipated.

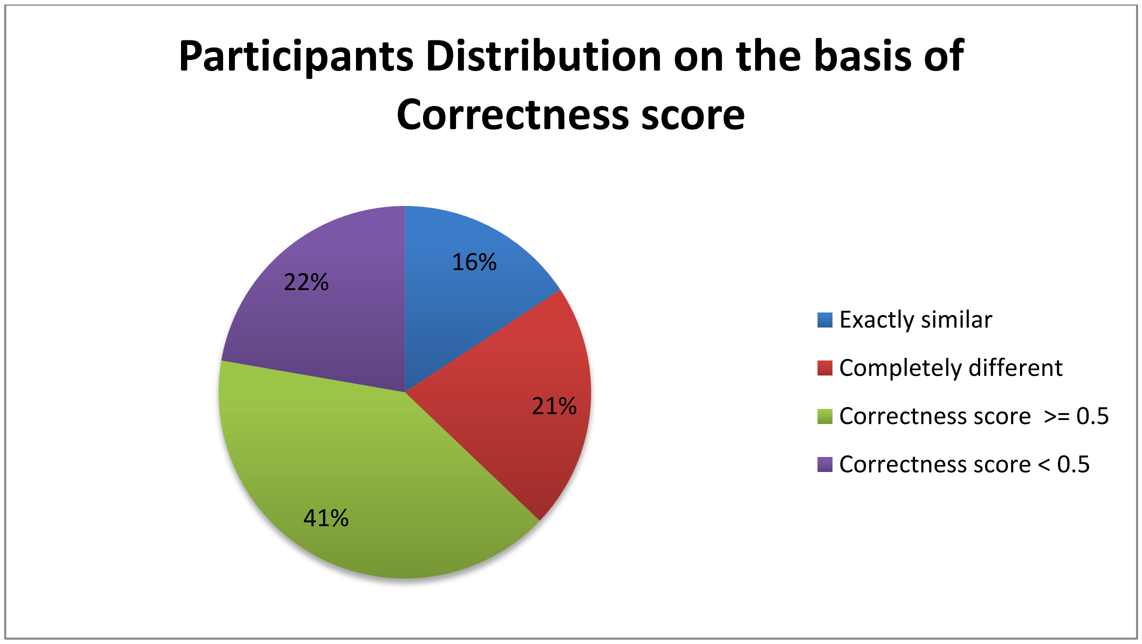


Figure 27 Distribution of Participants on the Basis of Correctness Score

4.4.2 Relation between effort and size of model fragments

A regression analysis can be used to present the relationship between a dependent and an independent variable. In this section the effort executed by the students in creating a model fragment is considered to be the dependent variable whereas the fragment size is the independent variable. This will assist the author in drawing some conclusions for the research question QR_2: *Does model fragment size contribute to the effort put in creating a model fragment?*

A simple linear regression equation is $Y = b + mx$, in which Y is the dependent variable, and x represents the independent variable. The author has modified this equation to better explain her work. The equation for linear regression becomes

$$Effort = \alpha + \beta * size + M(type)$$

Equation 1 Linear regression equation

Where α represents the intercept value, β is the model size co-efficient and $M(type)$ represents the co-efficient for model type. This latter co-efficient has been introduced so as to represent the different model types (which are class, interaction, activity, use case and state machine). For each model diagram, the effort is calculated twice (once for the minimum value of fragment size and then for the maximum value of fragment size) according to Equation 1. These values are represented in Table 18 for all the model diagrams.

Table 18 The values of Effort against the Minimum and Maximum Fragment Size for the Different Model Diagrams

	Fragment size	Activity	Class	Interaction	State machine	Use case
Min	5	1.95				
Max	27	2.21				
Min	5		1.44			
Max	22		1.64			
Min	4			1.64		
Max	9			1.70		
Min	3				2.05	
Max	16				2.20	
Min	3					1.56
Max	16					1.71

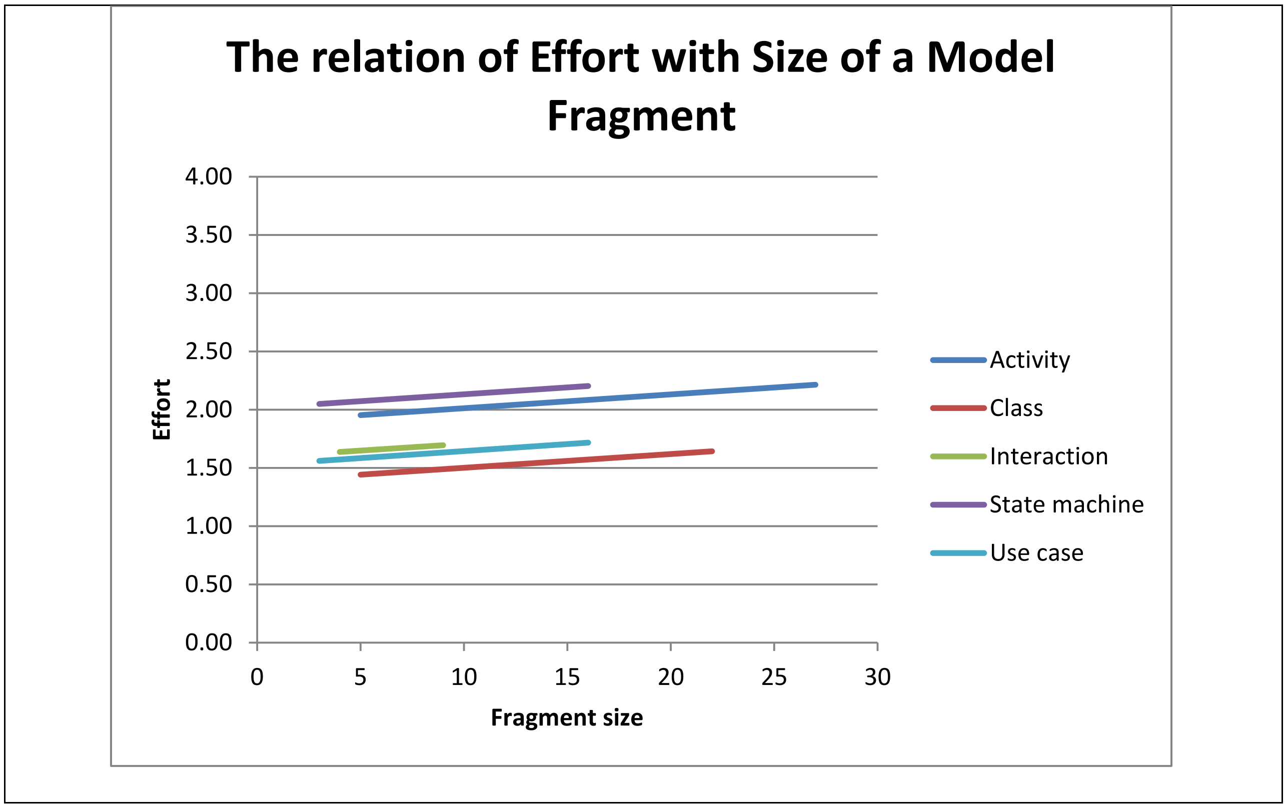


Figure 28 Relationship between the Effort in Creating a Model Fragment against the Size of a Model Fragment (for all the model diagrams)

In Figure 28 the fragment size values are shown on the x-axis. The effort is represented on the y-axis. The length of a line corresponds to the difference between maximum and minimum fragment sizes for a model diagram. The author expected to see an increase in the effort as the model size increased. But according to Figure 28 none of the model diagrams depict an increase in effort as the model size increased. It appears as if the effort does not depend on the fragment size rather the model type. According to the figure the participants found state machine models to be the most difficult model diagram to create as compared to the others. The class model fragments turned out to be the easiest to create for the participants.

4.5 Threats to Validity

Three case studies representing different software systems are chosen to collect requirements to be used in the questionnaires of the experiment. Two sets of questionnaires are made in which the textual requirements are of equal quantity and at the same abstraction level. The two sets of questionnaires have been randomly distributed amongst the 40 students. Students are used as subjects for the experiment. Most of them are master students except for 5 students which are at bachelor level, but with a good knowledge of requirements engineering concepts. Using students as subjects is acceptable as the experiment was designed for academic students and not working professionals. According to Kitchenham et al. (2002)

> "*Using students as subjects is not a major issue as long as you are interested in evaluating the use of a technique by novice or non-expert software engineers. Students are the next generation of software professionals so, are relatively close to the population of interest.*"

The experiment to validate the transformation rules has been conducted once, thus does not have a rich history for data comparisons. The effort level specified by the students is subjective to individual students.

Chapter 5 Discussion

5.1 Deriving model fragments from requirements

In this section the individual observations and results from the case study or experiment are not discussed rather the general impact of model fragment creation in translation of requirements to formal models is described. The idea behind this thesis was to explore the possibility of transforming textual requirements into formal models by adding model fragments. A set of rules for creating model fragments from requirements in natural language has been proposed. At present these transformation rules appear to work well for creating model fragments, but expanding these transformation rules might increase the quantity of model fragments against requirements.

The transformation rules proposed in this thesis considers both static and dynamic aspects of a system creating both structure and behavioral model diagrams (fragments), whereas most of the previous related work ((Moreno & van de Riet, 1997), (Overmyer, Lavoie, & Rambow, 2001), (Gelhausen & Tichy, 2007)) deals with the transformation of requirements into class models. Another conclusion to be drawn from this thesis is that the textual requirements are not restricted to any framework for transformation in contrast to some of the previous related work ((Kealey & Amyot, 2006), (Fatwanto A. , 2012)). The proposed method of transformation (textual requirements to formal models by adding model fragments) is exploited on a case study to determine if it is applicable on a large set of requirements. Figure 12 and Figure 22 show that majority of the requirements generates at least one model fragments, suggesting that the transformation rules (to create model fragments from requirements) are applicable on a large set of requirements. According to the author the transformation rules are simple and easy to use, but the process of manual fragment creation is time consuming. Thus the automation of this method can be suggested as future work. The notion of the type of model fragment to be created against a requirement type was inspected. It was observed that a dependency relation does exist between model type and requirement type. This observation leads to a research question (*"Which type of model fragment is created against which requirement type?"*) which can be interesting to work on in the future. One of the results from the case study showed that the different model diagrams depict variation in fragment sizes which lead the author to the formulation of another research question "*Does a particular model diagram always create model fragments in a specific size range?"*

Not all the requirements in the case study generate model fragments suggesting there are some short comings to the transformation rules. The size of the model fragments turned out to be relatively small, which was expected because the transformation rules considers transforming the requirements into model fragments with a few basic model elements. The small fragment size of the models was also observed for the model fragments created by the students in the experiment. These small model fragments are woven together through fragment weaving iteratively to obtain larger fragments which ultimately generate formal models.

5.2 Tracing from models to requirements

Model fragments not only support the forward generation of formal models from requirements but also supports the backward tracing from models to textual requirements. It is important to have requirements validated by the clients before starting the development of the system. The large complex UML diagrams are hard to understand by clients not familiar with these models and notations. Thus small pieces of model fragments will assist the requirement engineer to validate requirements from the clients by visually tracing a requirement from the UML model through the smaller pieces of model fragments to the original requirement text.

The process of creating model fragments sometimes results in the modification of the original requirement text. This modification does not change the scope of a requirement rather it changes the structure of the text. Such a modification also pushes the designer to add information to the text which might otherwise have been neglected. The requirements are modified so that the transformation rules can easily be applied to the requirement. As a result of which the requirements better fit with the guidelines of reinforcing natural language. Figure 15 show that 70% of the original requirements violate at least one of the guidelines. After modifying the requirements (during fragment creation) 42% violates at least one of the guidelines, which indicates an improvement in the quality of the textual requirements.

An unexpected result observed during the process of creating model fragments for the LMS case study was the addition of new requirements as a result of model fragment creation. The visual representation of a requirement assists the designer in identifying missing activities which could otherwise have been overlooked if the requirements were only in textual form. These missing requirements are often the trivial requirements, but nevertheless crucial for the development of a good system.

Chapter 6 Conclusion

The purpose of this thesis is to exploit whether model fragments can be used as an intermediate step in transforming textual requirements to formal models. The thesis briefly discusses how model fragments can be woven together to obtain formal models. This approach helps in bridging the communication gap between clients and Requirement Engineers and also supports forward and backward traceability.

For the purpose of creating model fragments from textual requirements, transformation rules are formalized and included in this thesis. A case study is selected to see whether or not model fragments can be created against a large set of requirements. From the case study it was observed that the transformation rules are easy and simple to use. Also model fragments can be created for a large set of requirements, but the manual process of model creation is a very time consuming process. Various empirical analyses are done on the model fragments to obtain some insights. The first insight obtained was that there is not an even distribution of the various model diagrams for a given set of requirements. Secondly the relationship between the requirement type and model type was considered which led to the formulation of a research question (Q_1: *"Does the requirement type determine the type of model fragment to be created against it?"*). Thirdly it was observed that on average the size (number of model elements) of the fragments was small and most of the model fragments lied in the range of 3-15 model elements.

The push back of the model fragments creation on the textural requirements was also observed for the case study. It is interesting to see that some of the original requirements are modified during the process of fragment creation. It was noted that after modifying the requirements (to better fit the transformation rules) the textual requirements better fit with the guidelines of reinforced natural language. The creation of model fragments also resulted in identifying some of the trivial requirements that otherwise might have been neglected by the clients or requirement engineer.

To validate the transformation rules an experiment was conducted on 40 students of the course 02264 "Requirements Engineering" at DTU Lyngby. The correctness score recorded for the individual requirements showed that 55% of the participants were able to create the correct model type fragment. This indicates that there is room for improving and expanding the transformation rules. It was observed that according to the participants the effort put in creating a model fragment did not depend on the fragment size rather it depends on the model diagram.

6.1 Future Work and Limitations

In the previous section the major results and observations inferred from this thesis was described. This section comprises of the various limitations this study has and proposes some recommendation that can be done as future work. At present the process of creating model fragments from textual requirements is done manually. Even though the author worked on a single case study for this thesis, the process of model creation was a very time consuming one. Hence as future work, the development of a tool to automate this process is suggested. Instead of having a single case study to draw conclusions from, having more case studies would help in comparing results to draw conclusions.

The experiment was conducted on students of Requirements Engineering, but the same experiment can be performed on Professionals or Academic Scholars/Researchers of the Software Engineering field to see whether the results deviate or not. For future experiments to be performed, the layout of the questionnaire can be modified so as to encourage the subjects to create more than one model fragment.

Chapter 7 Bibliography

A Guide to the Project Management Body of Knowledge. (2008). Pennsylvania: Project Management Institute, Inc.

(2013). *S2 Analysis Project Sample [LMS 3.1].* Lyngby.

Al-Rawas, A., & Easterbrook, S. (1996). *Communication Problems In Requirements Engineering: A Field Study.* London: Proceedings of the First Westminister Conference on Professional Awareness in Software Engineering, Royal Society.

Arlow, J., & Neustadt, I. (2004). *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML.* Boston: The Addison-Wesley Object Technology Series.

Balzar, R., Goldman, N., & Wile, D. (1978). Informality in Program Specification. *IEEE Trans. On software Engineering, vol. 4, no. 2*.

Cockburn, A. (2000). *Writing Effective Use Cases.* USA: Addison-Wesley Professional.

Cysneiros, L. M., & do Praito Leite, J. S. (2004). Non-Functional REquirements: From Elicitation to Conceptual Models. *IEEE Trans. On Software Engineering , vol. 30, no. 5*, 328-350.

Davis, A. M. (2005). *Just Enough Requirements Management: Where Software Development Meets Marketing.* New York: Dorset House Publishing.

Dobing, B., & Parsons, J. (2006). How UML is Used. *Communication of the ACM*, 109-113.

Ergorov, A., Hansen, K., Pol, A., Sukosd, A., Shan, L., & Kalmus, J.-L. (2012). *Sjællands Banken Customer Kiosk: System Analysis Document.* Lyngby.

Fatwanto, A. (2011). *A Concern-Aware Requirements Engineering Framework.Ph.D Thesis.* Canberra: The Australian NAtional University.

Fatwanto, A. (2012). Software Requirements Translation from Natural Language to Object-Oriented Model. *Control Systems & Industrial Informatics*, 191-195.

Fowler, M. (2003). *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* Addison Wesley.

Gelhausen, T., & Tichy, W. F. (2007). Thematic Role Based Generation of UML Models from RealWorld. *International Conference on Semantic Computing*, 282-289.

Gotel, O., & Finkelstein, A. (1994). An Analysis of the Requirements Traceability Problem. *Proc. of First International Conference on Requirements Engineering*, 94-101.

Haramain, H., & Gaizauskas, R. (2000). CM-Builder: An Automated NL-Based Case Tool. *Proceeding of the 15th International Conference on Automated Software Engineering, IEEE Computer Society*, 45-53.

Hasegawa, R., Kitamura, M., Kaiya, H., & Saeki, M. (2009). Extracting Conceptual Graphs from Japanese Documents for Software Requirements Modeling. *Asia-Pacific Conference on Conceptual Modelling (APCCM). Australian Computer Society*, 87-96.

Kealey, J., & Amyot, D. (2006). Towards the Automated Conversion of Natural-Language use Cases to Graphical use Case Maps. *Canadian Conference on Electrical and Computer Engineering*, 2377-2380.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., et al. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Trans. Softw. Eng.*, 721-734.

körner, S. J., & Brumm, T. (2010). Natural Language Specification Improvement with Ontologies. *International Journal of Semantic Computing (IJSC)*, 445-470.

Koziolek, H. (2008). Goal, Question, Metric. *Lecture Notes in Computer Science*, 39-42.

Kragelund, J. (2012). *Advanced Tool Support for Requirements Engineering.* Lyngby.

Kroha, P. (2000). *Database and Expert Systems Applications.* Berlin: Springer.

Landhäußer, M., körner, S. J., & Tichy, W. F. (2014). From Requirements to UML Models and Back: How Automatic Processing of Text can Support Requirements Engineering. *Software Qual J*, 121-149.

Lange, C. F., Chaudron, M. R., & Muskens, J. (2006). In Practice: UML Software Architecture and Design Description. *IEEE Computer Society*, 40-46.

Miriala, K., & Harandi, M. T. (1991). Automatic Derivation of Formal Specification from Informal Description. *IEEE Trans. On software Engineering, vol. 17, no. 10*.

Montes, A., Pacheco, H., Estrada, H., & Pastor, O. (2008). Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach. *Preceeding of the 13th International Conference on Application of Natural Language to Information Systems, Lecture Notes in Computer Science*, 325-326.

Montes, A., Pacheco, H., Estrada, H., & Pastor, O. (2008). Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach. *Preceeding of the 13th International Conference on Application of Natural Language to Information Systems, Lecture Notes in Computer Science*, 325-326.

Moreno, A. M., & van de Riet, R. P. (1997). Justification of the equivalence between linguistic and conceptual patterns for the object model.

Overmyer, S. P., Lavoie, B., & Rambow, O. (2001). Conceptual Modeling through Linguistic Analysis Using LIDA. *Proceedings of the 23rd International Conference on Software Engineering*, 401-410.

Pedersen, J., Ali, S., Ellavarason, E., & Haamann, M. (2012). *Mobile Match Maker System: System Analysis Document.* Lyngby.

Ramesh, B., & Jarke, M. (2001). Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 58-93.

Reubenstein, H. B., & Waters, R. C. (1991). The Requirements Apprentice: Automatic Assistance for Requirements Acquisition. *IEEE Trans. On software Engineering, vol. 17, no. 3*.

Siikarla, M., Peltonen, J., & Koskinen, J. (2006). Towards Unambiguous Model Fragments. *Nordic Jounal of Computing*.

Störrle, H. (2011). *Library Managment System: System Analysis Document.* Lyngby.

Wieringa, R. (1995). An Introduction to Requirements Traceability. *Technical Report IR-389 Faculty of Mathematics and Computer Science*.

Appendix

A. Transformation Rules (from textual requirements to model fragments.

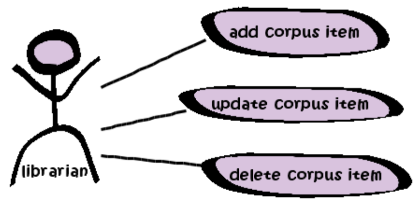
1. If a requirement specifies a role with activities, create a use case diagram.

1a Subjects and nouns indicate actors or components.

Librarians may add, update, and delete corpus items manually.

1b Predicates (strong verbs) indicate use cases.

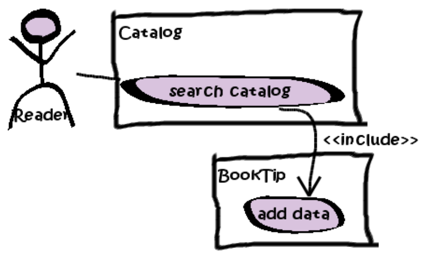
1c If a predicate relates to a subject, the corresponding actor and use case shall be connected by an association.



1d Prepositions indicating location or ownership (“in”, “within”) indicate the container of use cases.

1e Relational prepositions (“from”, “to”, “via”, “by”) indicate relationships between use cases.

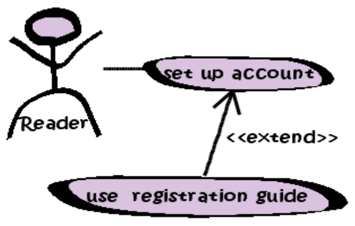
The BookTip system will add data to the result of a search in the catalog.



1f Unrestricted relations (“might”, “may”, “extends”) indicate “extends” relations.

An online registration guide might be used by the reader to set up a new account.

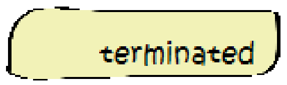
Restricted relations (“must”, “includes”, “will”) indicate “includes” relationships.



1. If the requirement specifies one or more alternative situations, create a state machine diagram

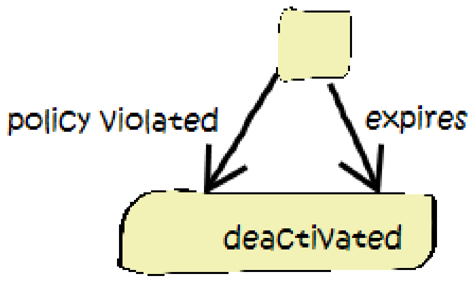
2a Adverbs, adjectives, and passive verbforms indicate states.

A terminated account does not offer any actions.

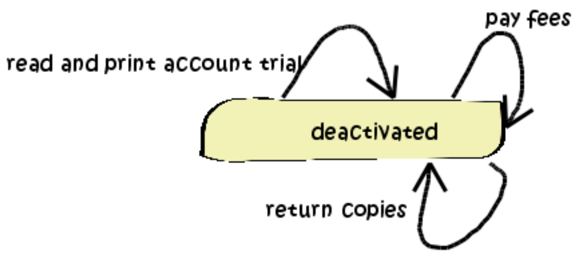


2b Verbforms in active tense, progressive tense and as nouns-forms indicate transitions.

If a reader account expires, or violates policy, the account is deactivated automatically.

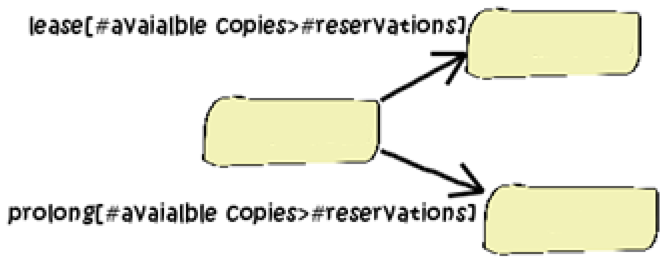


A deactivated reader account allows readers only payment of fees, reading and printing the account action trail, returning of copies, and termination of the account.



2c Restrictions and limitations (“only if”, “unless”, “may not”) of transitions indicate guards.

A reader may only lease or prolong a medium if there are more copies available than there are reservations for the medium.


lease[#avaialble copies>#reservations]
prolong[#avaialble copies>#reservations]

2. If the requirement contains a temporal aspect, create a sequence diagram.

Keywords include contain concrete durations “5sec” or temporal prepositions (“before”,” after”, “within”, “less than”).

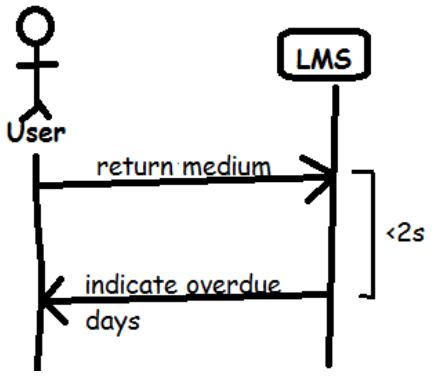
3a Subjects and nouns in phrases with active tense indicate lifelines.

If an overdue copy is returned to the library, then the LMS indicates the overdue days to the user within 2s.

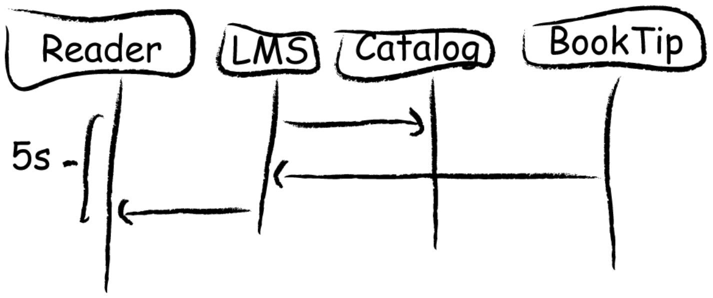
3b Phrases in passive tense indicate messages.

3c Sequencing between sub-phrases ("if/then", "afterwards", "next") + directional prepositions indicates sequential ordering between messages.

3d Temporal specifications ("in", "within", "after", "2s") indicate time constraints.



LMS will provide the reader between 3 and 5 reading suggestions from BookTip within 5s after issuing the query to the Catalog.



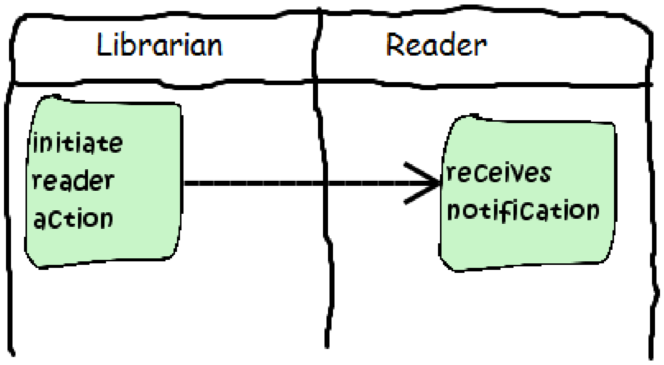
3. **If the requirement describes a sequence of steps and/or case distinctions, create an activity diagram.**

4a Subjects indicate lifelines.

If a librarian initiates reader actions on behalf of a reader, the reader receives notification.

4b Action verbs indicate actions.

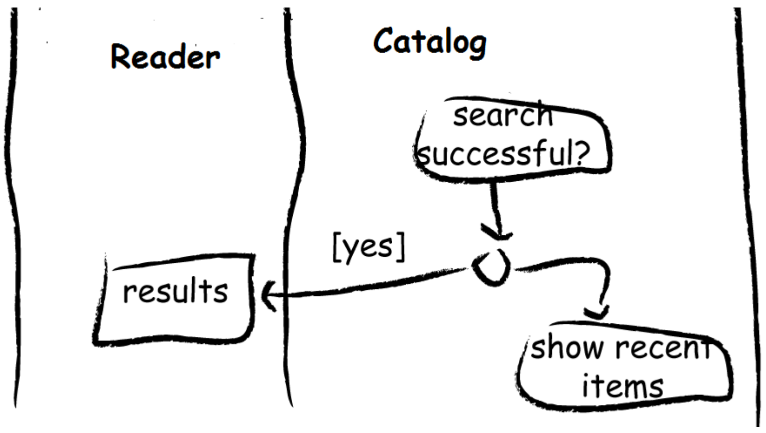
The subject of a predicate indicates the lifeline of an action.



4c Objects indicate data stores or data entities.

If the search was successful, the results are returned to the reader; otherwise, recent catalog items are shown.

4d Directional predicates (“from”, “to”, “of”) and specific verbs (“transmit”, “return”, “send”) indicate data flow.



4e Case distinctions (“if/else”, “otherwise”) indicate Decision/Merge nodes.

4f Control flow is indicated by temporal expressions (“then”, “afterwards”) or past tense present perfect.

4. If the requirement uses expressions describing static relationships, create a class diagram.

Keywords include “contains”, “has”, “consists”, “owns”.

5a Nouns not described as parts of other nouns indicate classes.

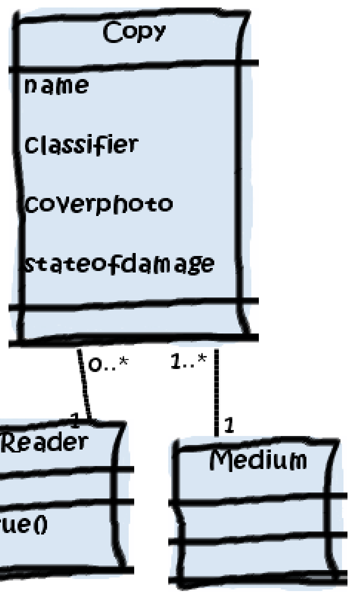
Each reader can issue multiple copies of a medium, with a copy consisting of a name, classifier, cover photo and state indicating its degree of damage.

5b Nouns described as parts of other nouns indicate class properties.

5c Action verbs indicate class methods.

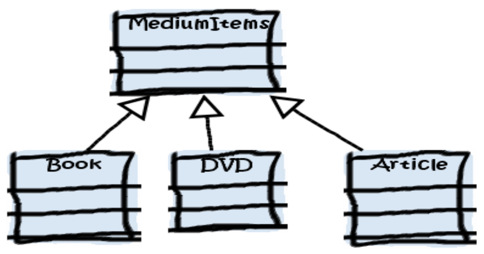
5d Structural verbs (“has”, “owns”, “contains”) indicate composition relationships.

5e Existential or universal quantifiers indicate general associations and a multiplicity. Numeric values or quantity key words (“each”, “many”, “several”, “multiple”, “a single”) indicate a multiplicity.



5f Verb forms like “is a” or “kind of” and concrete examples (“like”, “such as”) indicate generalizations.

There are three kinds of MediumItems, Books, DVDs, and Articles.



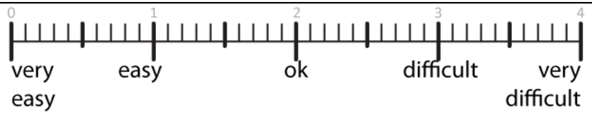
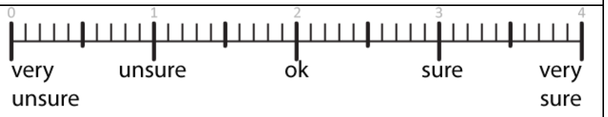
Appendix

B. Questionnaire for the Experiment

Requirement 1

If a reservation defaults, it is removed from the list of reservations for the medium it refers to and the medium enters the state available if no other reservations are pending.

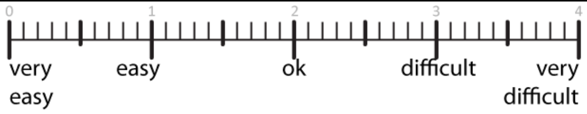
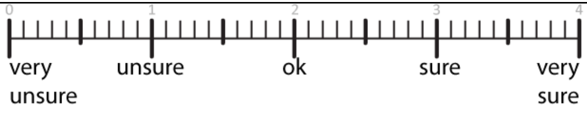
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	
3. How sure are you, that the model fragment really captures the requirement?	
4. Time after completing this task	

Requirement 2

When a copy is returned at the front desk, the librarian present there is notified within 1s if there is currently a reservation for the medium of the copy.

Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	 0 1 2 3 4 very easy easy ok difficult very difficult
3. How sure are you, that the model fragment really captures the requirement?	 0 1 2 3 4 very unsure unsure ok sure very sure
4. Time after completing this task	

Requirement 3

Librarians and Readers may post and inspect media they think should be acquired by the library to

a public “wish list”.	
Model Fragment	
1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 4 If a fee calculation yields a non-zero result; the reader is notified of the computation, its result and provided with an explanation.

Model Fragment	
1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 5
A search yields a list of overview results which consists of the medium category, author, title and publication year.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 6 The personal user might also receive suggestions from the MMM (Mobile Match Maker) system in addition to manually searching for restaurants/stores/attractions/hotels.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy, easy, ok, difficult, very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure, unsure, ok, sure, very sure
4. Time after completing this task	

Requirement 7 If the PMMD (Portable Medical Monitoring Device) of a personal user receives data exceeding the safety threshold value then the EMT (Emergency Medical Technician) is notified.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy, easy, ok, difficult, very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure, unsure, ok, sure, very sure
4. Time after completing this task	

Requirement 8 With the violation of MMM (Mobile Match Maker) policies, an account is deactivated automatically in less than 10secs.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 9 The MMM (Mobile Match Maker) system suggests other users to a personal user on the basis of common interests or at least one mutual friend.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy, easy, ok, difficult, very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure, unsure, ok, sure, very sure
4. Time after completing this task	

Requirement 10 When a personal user subscribes to a store, he/she gets notified about discounts and new arrivals.	
Model Fragment	
1. Which rules have you applied for deriving this fragment?	

2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 11

When the money transaction is successful, the customer is notified through email or with a paper receipt.

Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult

3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

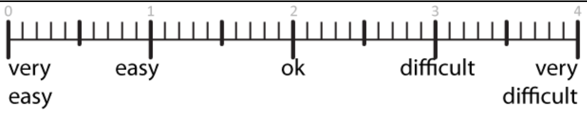
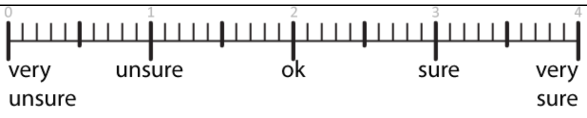
Requirement 12

A customer can transact money by authenticating him/herself.

Model Fragment

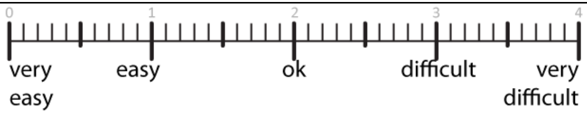
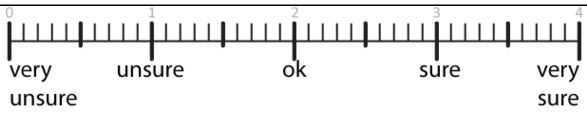
1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure

4. Time after completing this task	

Requirement 13 Activated, deactivated or closed are the states of a bank account, of which a bank account can only be in one state at any given time.	
Model Fragment	
1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	
3. How sure are you, that the model fragment really captures the requirement?	
4. Time after completing this task	

Requirement 14 A bank employee should be able to review a loan request and offer a loan in case in case the loan request amount is higher than €5000.
Model Fragment

1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	0 1 2 3 4 very easy · easy · ok · difficult · very difficult
3. How sure are you, that the model fragment really captures the requirement?	0 1 2 3 4 very unsure · unsure · ok · sure · very sure
4. Time after completing this task	

Requirement 15 Scanning a valid customer id with the readable ID device should display name and photo of the customer within 5 seconds.	
Model Fragment	
1. Which rules have you applied for deriving this fragment?	
2. How difficult was it to derive this fragment?	
3. How sure are you, that the model fragment really captures the requirement?	
4. Time after completing this task	

Appendix

C. Table including Requirements with their Correctness Score (Experiment)

Participant ID	REQID	Fragment Type	Correctness Score	*Participant ID*	REQID	Fragment Type	Correctness Score
6	1	ad	0.5	*22*	9	sd	0
7	1	ad	0.5	*32*	9	ud,cd	1
8	1	ad	0.5	*23*	9	ud	1
12	1	ad	0.5	*30*	9	ud	1
16	1	ad	0.5	*37*	9	ud	1
27	1	ad	0.5	*39*	9	ud	1
28	1	ad	0.5	*40*	9	ud	1
26	1	id	0.5	*3*	10	ad	0.33
13	1	sd, ad	1	*4*	10	ad	0.33
29	1	sd,cd	0.5	*14*	10	ad	0.33
11	1	sd	0.5	*15*	10	ad	0.33
17	1	sd	0.5	*19*	10	ad	0.33
18	1	sd	0.5	*22*	10	ad	0.33
24	1	sd	0.5	*23*	10	ad	0.33
25	1	sd	0.5	*32*	10	ad	0.33
31	1	sd	0.5	*37*	10	ad	0.33
33	1	sd	0.5	*40*	10	ad	0.33
35	1	sd	0.5	*1*	10	cd	0
31	2	ad	0.25	*2*	10	id	0
5	2	id	0.25	*20*	10	id	0
6	2	id	0.25	*21*	10	id	0
7	2	id	0.25	*34*	10	id	0
8	2	id	0.25	*36*	10	id	0
10	2	id	0.25	*39*	10	id	0
11	2	id	0.25	*38*	10	sd	0.33
12	2	id	0.25	*9*	10	ud	0.33
13	2	id	0.25	*30*	10	ud	0.33
16	2	id	0.25	*1*	11	ad	0.5
17	2	id	0.25	*2*	11	ad	0.5
18	2	id	0.25	*3*	11	ad	0.5
24	2	id	0.25	*9*	11	ad	0.5
25	2	id	0.25	*20*	11	ad	0.5

Participant ID	REQID	Fragment Type	Correctness Score	*Participant ID*	REQID	Fragment Type	Correctness Score
26	2	id	0.25	*21*	11	ad	0.5
27	2	id	0.25	*22*	11	ad	0.5
28	2	id	0.25	*23*	11	ad	0.5
29	2	id	0.25	*30*	11	ad	0.5
33	2	id	0.25	*32*	11	ad	0.5
35	2	id	0.25	*34*	11	ad	0.5
13	3	ad	0	*38*	11	ad	0.5
27	3	ad	0	*39*	11	ad	0.5
5	3	ud	1	*40*	11	ad	0.5
6	3	ud	1	*4*	11	id	0
7	3	ud	1	*14*	11	id	0
8	3	ud	1	*36*	11	sd,id	0.5
11	3	ud	1	*15*	11	sd	0.5
12	3	ud	1	*19*	11	sd	0.5
16	3	ud	1	*37*	11	sd	0.5
17	3	ud	1	*1*	12	ad	0.5
18	3	ud	1	*14*	12	ad	0.5
24	3	ud	1	*20*	12	ad	0.5
25	3	ud	1	*34*	12	ad	0.5
26	3	ud	1	*39*	12	ad	0.5
28	3	ud	1	*40*	12	ad	0.5
29	3	ud	1	*2*	12	id	0
31	3	ud	1	*9*	12	id	0
33	3	ud	1	*15*	12	id	0
35	3	ud	1	*21*	12	id	0
12	4	ad	0.5	*30*	12	id	0
16	4	ad	0.5	*36*	12	id	0
17	4	ad	0.5	*3*	12	sd	0
24	4	ad	0.5	*32*	12	sd	0
25	4	ad	0.5	*4*	12	ud	0.5
26	4	ad	0.5	*19*	12	ud	0.5
27	4	ad	0.5	*22*	12	ud	0.5
28	4	ad	0.5	*23*	12	ud	0.5
29	4	ad	0.5	*37*	12	ud	0.5
31	4	ad	0.5	*38*	12	ud	0.5
33	4	ad	0.5	*1*	13	cd	1
35	4	ad	0.5	*23*	13	cd	1
13	4	id	0	*32*	13	cd	1

Participant ID	**REQID**	**Fragment Type**	**Correctness Score**	***Participant ID***	**REQID**	**Fragment Type**	**Correctness Score**
18	4	id	0	*39*	13	cd	1
5	4	sd	0.5	*9*	13	sd	0
8	4	sd	0.5	*14*	13	sd	0
31	5	ad	0	*15*	13	sd	0
13	5	cd,ad	0.5	*19*	13	sd	0
5	5	cd	0.5	*20*	13	sd	0
8	5	cd	0.5	*21*	13	sd	0
10	5	cd	0.5	*22*	13	sd	0
16	5	cd	0.5	*30*	13	sd	0
17	5	cd	0.5	*34*	13	sd	0
25	5	cd	0.5	*36*	13	sd	0
26	5	cd	0.5	*37*	13	sd	0
27	5	cd	0.5	*38*	13	sd	0
28	5	cd	0.5	*40*	13	sd	0
33	5	cd	0.5	*4*	13	ud	0
35	5	cd	0.5	*19*	14	ad,ud	0.67
18	5	cd,id	0.5	*1*	14	ad	0.33
12	5	ud	0	*4*	14	ad	0.33
5	6	ud	1	*9*	14	ad	0.33
8	6	ud	1	*22*	14	ad	0.33
18	6	ud	1	*36*	14	ad	0.33
26	6	ud	1	*37*	14	ad	0.33
28	6	ud	1	*38*	14	ad	0.33
31	6	ud	1	*39*	14	ad	0.33
33	6	ud	1	*40*	14	ad	0.33
35	6	ud	1	*20*	14	ad,ud	0.67
8	7	ad	0.5	*34*	14	ad,ud	0.67
28	7	ad	0.5	*23*	14	sd	0.33
31	7	ad	0.5	*30*	14	sd	0.33
35	7	ad	0.5	*14*	14	ud	0.33
5	7	id	0	*15*	14	ud	0.33
28	8	id	0.33	*21*	14	ud	0.33
35	8	id	0.33	*21*	15	ad	0.5
8	8	sd	0.33	*4*	15	id	0.5
31	8	sd	0.33	*9*	15	id	0.5
1	9	ad	0	*22*	15	id	0.5
2	9	ad	0	*23*	15	id	0.5
3	9	ad	0	*30*	15	id	0.5

Participant ID	**REQID**	**Fragment Type**	**Correctness Score**
4	9	ad	0
9	9	ad	0
14	9	ad	0
15	9	ad	0
19	9	ad	0
20	9	ad	0
34	9	ad	0
34	15	id	0.5
36	15	id	0.5
37	15	id	0.5
38	15	id	0.5
39	15	id	0.5
40	15	id	0.5