Project task

Title: Empirical Testing of a Clustering Algorithm for Large UML Class Diagrams; an Experiment

I store utviklingsprosjekter kan diagrammer bli svært omfattende (f.eks. 100 klasser) slik at de er forholdsvis tidkrevende å finne fram i. Da er det viktig å kunne bruke ulike mekanismer for abstraksjon og filtrering. Vanlige abstraksjonsmekanismer i UML er generalisering (superklasse) og aggregering (komposisjon). En alternativ strukturingsmekasnisme, basert på en analogi til bykart (hvor ulike detaljruter fins på ulike sider), er foreslått i en artikkel av Moody og Sindre som vant Best Paper Award på konferansen ACIS-03. Metoden er foreløpig kun prøvd ut i praksis for ER-diagrammer og det er derfor ønske om en tilsvarende utprøving for UML-diagrammer. Det som ønskes i dette prosjektet er en eksperimentell utprøving av denne strukturingsmetoden, grovt skissert på følgende måte:

- Få tak i ett stort UML klassediagram, helst en modell fra ett reelt industriprosjekt
- Lage ulike presentasjoner av denne modellen (en flat representasjon, som er presentert ihht generalisering og aggregering, og en som er presentert ihht kartanalogien)
- Kjøre et eksperiment for å teste om modellen laget med kartanalogien er lettere å bruke enn det vanlige UML klassediagramet

ABSTRACT

One important part of developing information systems is to get as much insight as possible about the problem, and possible solutions, in an early phase. To get this insight, the actors involved need good and understandable models. One popular modeling approach is UML class diagrams. A problem with UML class diagrams is that they tend to get very large when used to model large-scale commercial applications. In the absence of suitable mechanisms for complexity management, such models tend to be represented as single, interconnected diagrams. Diagrams of this kind are difficult for stakeholders to understand and maintain. There have been developed algorithms for filtering large ER diagrams, and the aim of this project has been to try if one of these algorithms can be used for filtering UML class diagrams as well.

This paper describes a laboratory experiment which compares the efficiency of two different representation methods for documentation and maintenance of large data models. The representation methods compared are the ordinary UML class diagram, and the Leveled Data Model. The methods are compared using a range of performance based and perception based variables. The results show that the Leveled Data Model is not suited for modeling large generalization hierarchies. For other kinds of relations, the efficiency of the two modeling methods is the same. The participants preferred to use the ordinary UML diagram to solve the experimental tasks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This report describes an experiment that compare the use of a classical UML class diagram to the use of a Leveled Data Model. The Leveled Data Model was made using a method developed by Moody and Flitman [12].

The broad research questions addressed by this experiment are:

- How efficient is the suggested Leveled Data Model method compared to an ordinary UML model ?
- Does people perceive the leveled model to be a reasonable way to organize large UML diagrams ?

In the next chapter, Chapter 2, the theoretical foundation for this experiment is explained. It is explained why there is a need for decomposition of large UML diagrams. The Leveled Data Model is explained, and there is an overview of similar existing approaches and previous research.

In Chapter 3, the models used in the experiment are described. First the domain of the models is described, then the UML model and the Leveled Data Model are described. The decomposition in the leveled data model is evaluated, in relation to some principles for a good decomposition.

The next three chapters describes the experiment. First, the design of the experiment is described in Chapter 4. Next, the results is presented in Chapter 5. Finally, the results are analyzed and discussed in Chapter 6.

The last chapter, Chapter 7 contains the conclusions from this experiment. There is also some suggestions for further work in the last chapter.

Chapter 2

Theoretical Foundation

In this chapter the theoretical foundation for this project is discussed. In Chapter 2.1 the need for decomposition of large UML diagrams is discussed. Chapter 2.2 gives an overview of existing approaches in the domain of decomposing large models. The Leveled Data Model is explained in Chapter 2.3. In Chapter 2.4 the conclusions from a previous project, about the same domain, is listed. Finally, in Chapter 2.5, the broad research questions addressed by this experiment is stated.

2.1 Background

In the rapidly changing business environment today the information systems are crucial to the success of most companies. The increasing complexity and evolving nature of organizations mean that systems development approaches must cope with a highly complex and changing reality. Although substantial progress has been made with respects to technology, systems development still suffers from low productivity, high development and maintenance costs and delays in delivering on time. One important part of developing information systems is to get as much insight as possible about the problem, and possible solutions, in an early phase. To get this insight, the actors involved need good and understandable models of those parts of the world that may have an impact on the target information system, as well as on its environment [13].

There are many different kinds of models used for this purpose. One popular approach is UML models. A problem with UML class diagrams is that they tend to get very large when used to model large-scale commercial applications. Class diagrams of hundred or more classes are not unusual. In the absence of suitable mechanisms for complexity management, such models tend to be represented as single, interconnected diagrams. Diagrams of this kind are difficult for stakeholders to understand and maintain [11].

This craves for good mechanisms for abstraction, or filtering, for the models. Filtering

in this context means that you do not see the whole model at once. This can be favorable for several reasons. One reason is that different kinds of users often need to see different parts, or different views, of the model. Another reason is that one kind of users needs to see different parts, or different views, of the model at various times of the development phase. In some modeling languages this is not a problem, because the language itself offers mechanisms for this. The data flow diagrams for instance, are decomposed into many levels, and each level has at most 10 processes. But for other kinds of models, like the class diagram, it could be a need for filtering in addition to the abstraction mechanisms already offered by the modeling language.

2.2 Existing approaches

There has been done some research in the domain of filtering large models. Most of this research have been done within the Entity Relationship (ER) domain. This chapter provides a overview of some of this research.

In an article called "An Ontological Model of an Information System" [15], Wand and Weber propose an ontological model of an information system that provides precise definitions of fundamental concepts like system, subsystem, and coupling. They use this model to analyze some static and dynamic properties of an information system and to examine the question of what constitutes a good decomposition of an information system.

Gandhi, Robertson and Gucht proposed a method called Leveled Entity Relationship Model (LER) [5]. The aim of the LER formalism was to model the complex data in advanced database systems. The LER formalism models structured data by leveling ER diagrams so that deeper layers provide greater structural detail. Moreover, elements deep inside one structure reference elements deep inside another structure. LER cleanly formalizes the relation between such deep structural elements.

Danoch, Shoval and Balaban provide a brief overview of some of the abstraction mechanisms proposed for ER diagrams [3]. They also proposed their own method for creating hierarchical ER diagrams, and did an experiment to compare their decomposed diagrams with flat ER diagrams [4]. From this experiment they found that users preferred the hierarchical model, but they could not prove that working with the hierarchical model gave better results.

Moody and Flitman have developed a quite similar method for filtering large ER diagrams [12]. This method is called a Leveled Data Model, and its usefulness has been verified in several experiments. These experiments are described in the Following articles:

Entity Connectivity vs. Hierarchical Leveling as a Basis for Data Model Clustering

This article [10] describes a series of laboratory experiments which evaluate the validity of connectivity (defined as the number of relationships an entity participates in) as a basis for clustering compared to hierarchical leveling. The first two experiments investigate the relationship between the metrics and perceptions of importance, while the third experiment investigates their relationship to how people intuitively cluster entities. The results show that connectivity provides an empirically valid basis for clustering data models, which closely matches human perceptions of importance.

Dealing With Complexity in Information System Modeling

This article [9] describes the development and empirical validation of leveled data modeling in the ER domain. A combination of research methods were used to validate the method. Action research was first used to test and refine the method in a real-world setting. Eight action research studies were conducted in eight different organizations. Once the method had become stable, two laboratory experiments were conducted to evaluate its effectiveness compared to the standard ER model and methods previously proposed in the literature. Finally, a field experiment was conducted using experienced practitioners to evaluate the likelihood of the method being accepted in practice.

Comparative Evaluation of Large Data Model Representation Methods: The Analyst's Perspective

This paper [8] describes a laboratory experiment which compares the effectiveness of different representation methods for documentation and maintenance of large data models (analyst's viewpoint). The methods are compared using a range of performance-based and perception-based variables, including time taken, documentation correctness, consistency, perceived ease of use, perceived usefulness and intention to use.

2.3 Leveled Data Models

Since an UML class diagram has many similarities to an ER diagram, it would be interesting to find out if the same filtering method could be used for UML class diagrams as well. Using the same filtering method for UML class diagram is proposed in an article by Moody and Sindre [11], where they suggest an algorithm for this. The article does not address the essential differences between ER-diagram and UML-class diagram in great detail. One main difference is that the UML class diagram has some relation types, generalization and aggregation, that ER diagrams does not have.

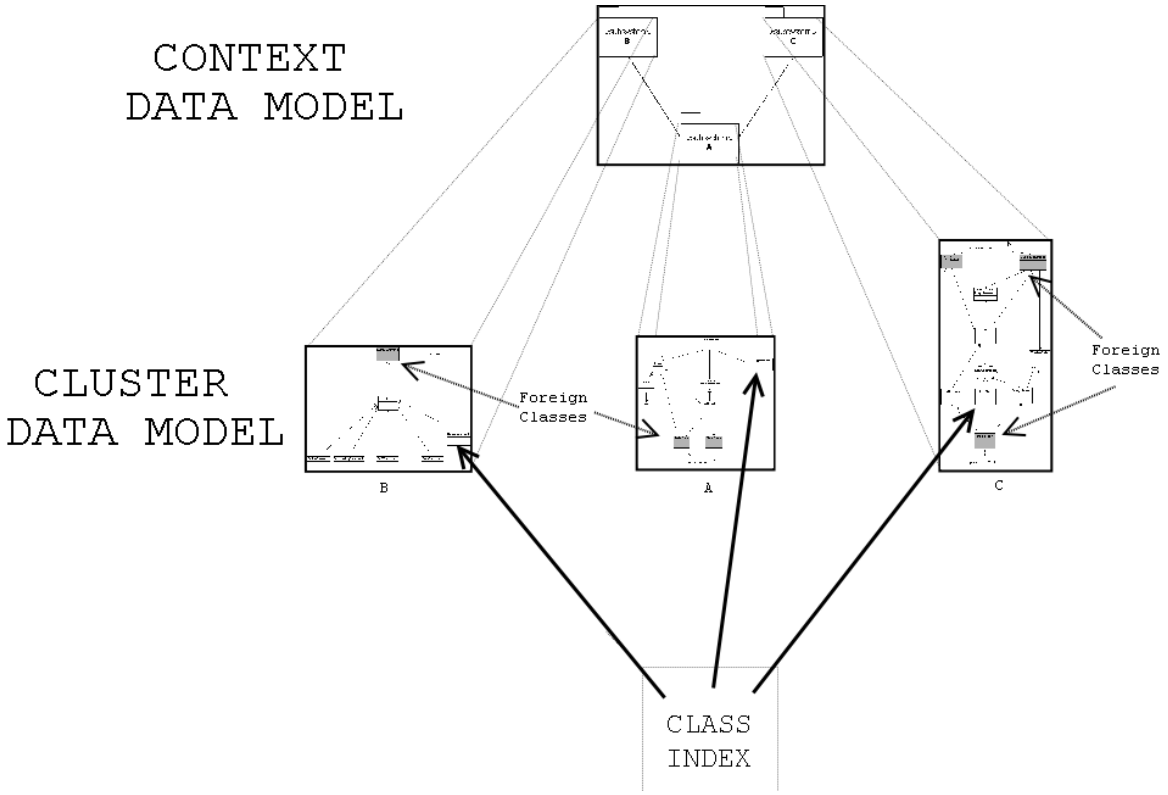

Figure 2.1: Leveled Data Model

The idea of the Leveled Data Model is to use a method for representing large data models based on the organization of a street directory. A Leveled Data Model consists of the following components (Figure 2.1)

- A high level diagram, called the Context Data Model, provides an overview of the model and how it is divided into clusters. This corresponds to the key map in a street directory.

- A set of named Cluster Models show a subset of the data model (a single cluster) in full detail. These correspond to detail maps in a street directory. Foreign classes are used to show cross-references between clusters. These correspond to inter-map references in a street directory.

- An Class Index are used to help locate individual classes within each cluster.

The model may be organized into any number of levels, depending on the size of the underlying data model.

2.4 Previous research

In a project preceding this one [6], I did some research on the Leveled Data Model. The focus was on diagrams that contained relations that are found in UML, but not in ER models, namely generalization and aggregation. An empirical approach was used. The algorithm was tried on some test models, and the decompositions were tested against a set of principles for good decomposition.

The main conclusions from this project were;

- Some kind of filtering is necessary to ease understanding of large UML class diagrams
- The tested algorithm is not suitable for class diagrams modeling a hierarchy, and thereby including a lot of generalization relations
- The algorithm works well on clustering class diagrams with about 40 classes, and without a large amount of generalization relations

2.5 Problem to be solved

The broad research questions addressed by this experiment are:

- How efficient is the suggested Leveled Data Model method compared to an ordinary UML model ?
- Does people perceive the leveled model to be a reasonable way to organize large UML diagrams ?

Chapter 3

The Models Used

The model used in the experiment described in this paper, is an UML diagram from a student project done at Duke University during spring 2001. The aim of the project was to design and implement a much simplified version of the computer programming program **Logo**. As a dialect of LISP, Logo is a complex and powerful language. "Simple Logo", or SLogo, should retain the features most commonly used by beginning users so that it could be used to provide an introduction to computer programming. The Logo computer programming language is further described in Chapter 3.1. The SLogo UML diagram is described in Chapter 3.2. In Chapter 3.3 the SLogo Leveled Data Model is described. This Leveled Data Model is also evaluated empirical with reference to the principles for a good decomposition.

3.1 The Logo computer programming language

Logo is a computer programming language designed to teach programming to children. It is a user-friendly, interpreted language, designed with a "low floor, high ceiling". In other words, the designers of Logo intended for the language to allow novice programmers to get started quickly with writing programs. They also wanted the language to be powerful and extensive for more advanced users.

In the early days, Logo was used to control a simple physical robot, called a turtle. Users could issue commands such as FORWARD 50 to make the turtle advance 50 steps, or RIGHT 90 to make it turn ninety degrees. The turtle robot carried a pen, so users could produce drawings on paper, such as the one shown in Figure 3.1, by controlling the turtle and its pen. The turtle, which has since moved on to the computer screen, has become one of the most familiar and important parts of the Logo language. Figure 3.2 shows a screen shot from the SLogo program.

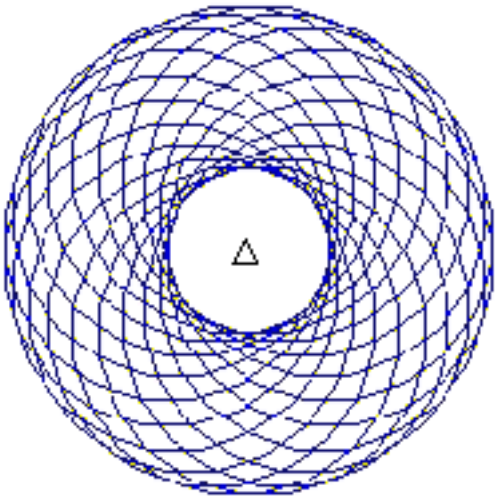

Figure 3.1: Logo drawing example

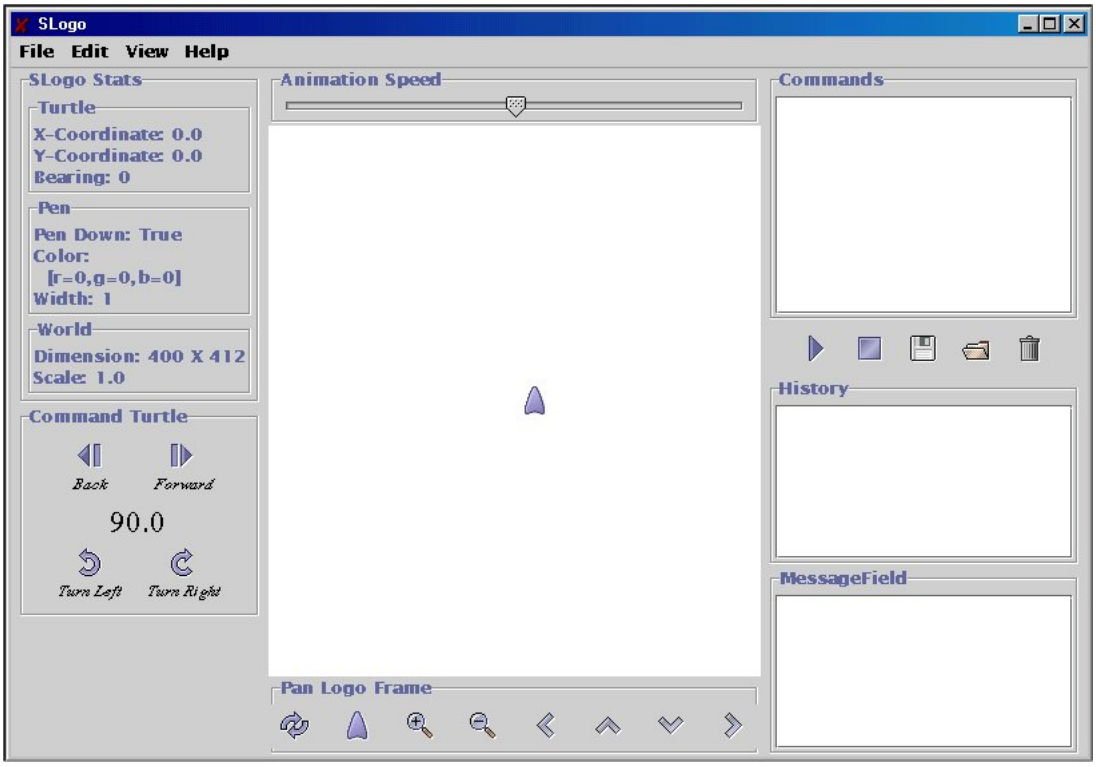

Figure 3.2: The SLogo GUI as it appears when it is first loaded

3.2 The SLogo UML diagram

The SLogo UML diagram (see Figure B.1) is developed using Java StructureBuilder. It consists of 89 classes and 3 interfaces. It contains a total of 95 relations, 55 of these are generalizations. There are 12 classes in the diagram that are not connected. The full picture can be printed in a readable size using one A1 sheet.

3.3 The SLogo Leveled Data Model

This chapter describes the Leveled Data Model used in the experiment. I have included the chapter about principles for a good decomposition from the pre-project [6] as an appendix, and refer to this in the description of the Leveled Data Model. Chapter 3.3.1 describes the highest level of decomposition. Chapters 3.3.2 and 3.3.3 describes the clusters of the second level of decomposition.

3.3.1 Level 1

The first level of the Leveled Data Model is shown in Appendix C. On this level the UML diagram is divided into 2 clusters, named SLogoCommand and SLogoController. These clusters are further divided into new clusters on the next level. The SLogo-Command cluster consists mainly of generalization relations, and the SLogoController cluster consists mainly of other relations. Table 3.1 describes the two clusters . Here is a description of the different columns.

1. Shows the name of the cluster, the same as the name of the central sub-cluster. (See Appendix A.4)

2. Shows the number of sub-clusters inside the cluster. (see Appendix A.5, A.6 and A.7)

3. Shows the number of internal relationships. (See Appendix A.9)

4. Shows the number of boundary relationships. (See Appendix A.8)

5. Shows where the figure can be found.

Both Clusters on this level are within the boundaries of the principles for a good decomposition (Appendix A).

1. Name	2. #sub-clusters	3. #int. rel.	4. #ext. rel.	5. Figure
SLogoCommand	6	5	2	D.1
SLogoController	5	4	2	D.2

Table 3.1: Clusters from the first level of decomposition

3.3.2 Level 2; The SLogoCommand Cluster

The SLogoCommand Cluster consists of six sub-clusters. The type of relations inside this cluster is mainly generalization. The six sub-clusters are described in table 3.2. The different columns are mainly the same as for the table in Chapter 3.3.1. The only difference is that column 2 contains the number of classes, instead of number of sub-clusters, in the cluster.

1. Name	2. #classes	3. #int. rel.	4. #ext. rel.	5. Figure
BooleanCommand	8	7	1	E.1
MathCommand	8	7	1	E.2
MovementCommand	10	9	1	E.3
PositionCommand	8	7	1	E.4
ViewCommand	10	9	1	E.5
SLogoCommand	4	3	7	E.6

Table 3.2: Sub-clusters from the SLogoCommand Cluster

It can be seen from the table that the clusters BooleanCommand, MathCommand and PositionCommand follows the principles for a good decomposition nicely. The clusters MovementCommand and ViewCommand violate the principle; "Cognitively Manageable" (Appendix A.5). This principle states that the maximum number of clusters/classes on each level should be nine. The reason to have ten classes in these clusters is that this makes the model less complex.

The problem cluster here is the SLogoCommand cluster. This cluster violates the principles; "Balanced" (Appendix A.7), "Loosely Coupled" (Appendix A.8) and "Highly Cohesive" (Appendix A.9). The principle "Balanced" states that the minimum number of clusters/classes on each level should be five. The reason I only have four classes her is that more classes here would have caused an even worse violation of the two other principles. The violation of the other two principles comes from the low number of internal relations, and the high number of external relationships. This is not good, and the reason for this is that the Leveled Data Model is not well suited for decomposing a hierarchy [6].

3.3.3 Level 2; The SLogoController Cluster

The SLogoController Cluster consists of five sub-clusters. The five sub-clusters are described in Table 3.3. The different columns are mainly the same as for the table in Chapter 3.3.1. The only difference is that column 2 contains the number of classes, instead of number of sub-clusters, in the cluster.

1. Name	2. #classes	3. #int. rel.	4. #ext. rel.	5. Figure
SLogoGui	9	15	3	F.1
SpecialCommand	7	6	2	F.2
TurtleAnimator	8	9	8	F.3
Utils	10	0	0	F.4
SLogoController	9	7	10	F.5

Table 3.3: Sub-clusters from the SLogoController Cluster

It can be seen from the table that the clusters SLogoGui and SpecialCommand follows the principles for good a decomposition nicely. The TurtleAnimator cluster has almost as many external as internal relations. This is not ideal, but it is acceptable. The Utils cluster is actually just a collection of classes that has no connections. Therefore, this cluster can not be evaluated by the principles for a good decomposition. I think this must be the best solution for this kind of classes, instead of spreading them around in the other clusters. That would cause all clusters to violate the good decomposition principles.

The SLogoController cluster violates the "Loosely Coupled" (Appendix A.8) principle, because it has so many external relations. This is impossible to avoid, since the SLogoController class is the most central class in the UML diagram, and have very many connections.

Chapter 4

Research Design

In this chapter the different aspects of the research design are discussed. In Chapter 4.1 the reasons to use a controlled experiment is explained. The characteristics of the experimental design is explained in Chapter 4.2. The independent and dependent variables in the experiment is discussed in the Chapters 4.3 and 4.4. In Chapter 4.5 the hypotheses are listed, and in Chapter 4.6 the participants are described. The experimental treatment is described in Chapter 4.7, and the materials used is described in Chapter 4.8. The last part of this chapter, Chapter 4.9, describes the tasks of the experiment.

4.1 Method Selection

There is a wide variety of research methods which may be used in conducting IS research. Different research methods are appropriate in different situations, depending on the research question and the stage of knowledge in the area being studied.Prior to this study, the proposed method had been tested in practice using an empirical approach [6].

A controlled experiment provides the most effective way to evaluate how efficient the proposed method is because:

- It allows direct comparisons to be made between different methods under controlled conditions through manipulation of experimental treatments
- It enables the method to be evaluated using objective and quantitative data
- It enables the method to be evaluated using independent participants

Since there are no decomposition methods widely used for UML, the Levelled Diagram is only compared to an ordinary UML diagram.

4.2 Experimental Design

The context of the experiment can be characterized according to four dimension [17]. The characteristics of this experiment are:

- **Off-line;** it is not executed in a real software project
- **Student;** the project is staffed with students, not proffessionals
- **Toy;** a constructed, not a real problem is used
- **Specific;** the studies are only valid in a specific contex

Furthermore, the experiment design can be described as a two group, post-test only design, with one active between-groups factor (representation method). One experimental groups is treatment groups, the other is control group.

4.3 Independent Variable

There are two independent variables, representation method and time limit. The representation method variable has two levels; ordinary UML model and Levelled model. The time variable was equal for both experimental groups for each individual group of questions. A smal experiment with 4 participants was held in advance to find a suitable time limit for each question group.

4.4 Dependent Variables

We distinguish between two types of dependent measures:

- Performance based (objective) measures: How effectively are the subjects able to perform the experimental task?
- Perception based (subjective) measures: How efficient do the subjects perceive the method to be?

Performance Based Measure

One performance based dependent variable were used to evaluate the methods:

A1: Documentation correctness: measured by the number of correct answers.

Perception Based Measures

From a scientific point of view, objective measures generally provide much more convincing evidence than subjective measures. However in decisions about whether to use a particular method, perceptions play a much more important role, because of the element of free will or intentionality in human behaviour [8].

I define two perception based variables for evaluating the methods:

A2: Perceived Ease of Use: Mesured by which model the participants prefered to use.

A3: Perceived Usefulness of the levelled model: Mesured by the degree to which a person believes that the levelled model representation method will be efficient in achieving its objectives.

4.5 Hypotheses

The two research questions from Chapter 2.5 is broken down into several hypotheses, each relating to a particular combination of independent and/or dependent variables.

- H0: Participants will make the same amount of errors using the Levelled Data Model as using the ordinary UML model
- H1: Participants will make fewer errors using the Levelled Data Model than using the ordinary UML model
- H2: Participants will make fewer errors using the ordinary UML model than using the Levelled Data model
- H3: Participants do not prefer one model type over the other
- H4: Participants prefer to use the Levelled Data Model over the ordinary UML model
- H5: Participants prefer to use the ordinary UML model over the Levelled Data Model

- H6: Participants think that the Levelled Data Model is a unreasonable way to present a large UML model
- H7: Participants think that the Levelled Data Model is a reasonable way to present a large UML model

4.6 Participants

There were 7 participants in the experiment, all of whom were final year Information Systems students at NTNU. They were expected to enter the work force 2-3 months after the experiment. All subjects participated voluntarily and were paid 150 NOK to particpy in the experiment. The 4 participants with the best score was payed additionally 50 NOK after the experiment. This was done to ensure that the participants was motivated to do their best. Subjects were randomly assigned to experimental groups.

4.7 Experimental Treatment

All participants were given a seven minute lecture about each of the representation methods being evaluated. They were also given a note about each representation method.

4.8 Materials

- **Large UML Model:** The SLogo UML diagram, the diagram can be found in Figure B.1. The model was printed on a 100 cm * 60 cm sheet. More details about the model can be found in Chapter 3.2
- **Levelled Data Model:** Contains; Contents, 3 levels of decomposition and an alfabetic index. It was printed on 18 A4 pages. More information about this model can be found in Chapter 3.3
- **Lecture notes about the model types**
- **Six groups of questions:** Each group consisting of 25 relations that should be verified or falsified
- **Post-Task Survey:** With questions about Perceived Ease of Use (A2) and Perceived Usefulness (A3

The printed size of the classes was the same in the Levelled Data Model as in the ordinary UML model. Details about the domain of the UML model can be found in Chapter 3.1.

4.9 Experimental Task

The participants where randomly divided into 2 groups (group A and group B), Group A started of using the ordinary UML diagram, group B started with the levelled model. They where given 3 groups of questions related to the model. Each question-group (QG) had an individual time limit. Each QG contained 25 claims about relations in the model that the participants had to verify or falsify. QG1 adressed generalization relations, QG2 adressed direct relations and QG3 adressed indirect relations. When the three first QGs where finished, the groups switched model, and the process was repeated with similar questions in QG4, QG5 and QG6. The time limits was different in the second half of the experiment.

At the end of the experiment all the participants filled out a post-task survey. This survey containd questions related to the perception of the preeceding tasks.

Chapter 5

Results

This chapter contains the results from the experiment. Chapter 5.1 contains the results from the performance based part of the experiment. Chapter 5.2 contains the results from the post-task survey: Analyses and discussion of all the results can be found in chapter 6.

5.1 Results; Performance Based Measure

This chapter contains results from the 6 groups of questions (QG) in the experiment. QG1 (Table 5.1) and QG4 (Table 5.4) had questions concerning generalization relations (super-classes and sub-classes). QG2 (Table 5.2) and QG5 (Table 5.5) had questions concerning direct connections. QG3 (Table 5.3) and QG6 (Table 5.6) had questions concerning indirect connections.

The tables has the following data for ordinary UML (Ordinary) and Leveled Data Model (Leveled):

- **Mean;** the average number of correct answers
- **Variance;** the variance of the answers
- **Observations;** the number of participants

The two last fields in the tables have the following data about the QG;

- **Max score;** the maximum number of correct answers that could be achieved
- **Time;** the time the participants could use

	Ordinary	**Leveled**
Mean	9	8,75
Variance	0	0,25
Observations	3	4
Max score	9	
Time	4 min	

Table 5.1: Results; QG 1

	Ordinary	**Leveled**
Mean	10	9,25
Variance	0	0,92
Observations	3	4
Max score	10	
Time	5 min	

Table 5.2: Results; QG 2

	Ordinary	**Leveled**
Mean	10,67	6,75
Variance	6,33	4,92
Observations	3	4
Max score	13	
Time	10 min	

Table 5.3: Results; QG 3

	Ordinary	**Leveled**
Mean	10,75	6,33
Variance	3,58	4,33
Observations	4	3
Max score	12	
Time	3 min	

Table 5.4: Results; QG 4

	Ordinary	**Leveled**
Mean	7,5	7,33
Variance	0,33	1,33
Observations	4	3
Max score	8	
Time	3 min	

Table 5.5: Results; QG 5

	Ordinary	**Leveled**
Mean	11,5	9
Variance	7	7
Observations	4	3
Max score	14	
Time	9 min	

Table 5.6: Results; QG 6

5.2 Results; Perception Based Measures

This chapter contains the results from the post-task survey. The first chapter, Chapter 5.2.1, contains results about preferred model type. In the second chapter, Chapter 5.2.2, the participants meaning about the Leveled Data Model is shown. The last chapter, Chapter 5.2.3, shows how satisfied the participants where with the pre-experiment lecture.

5.2.1 Participants Preferred Model Type

The participants where asked which model type they found easiest to use. The answers are presented in Table 5.7.

Answer	**Observations**
Ordinary UML Much easier	3
Ordinary UML easier	2
The same	1
LDM easier	1
LDM Much easier	0

Table 5.7: Participants prefered model type

5.2.2 Participants Opinion about the Leveled Data Model

The participants where asked if they could see the need for decomposition of large UML models, and if the Leveled Data Model seems like a reasonable way of doing this. All participants answered that they saw a need, and that the Leveled Data Model seems like a reasonable way of doing this.

5.2.3 Participants opinion about the pre-experiment lecture

All the participants had previous experience with ordinary UML diagrams. This was their first experience with the Leveled Data Model. They were asked if they thought they got enough education in the Leveled Data Model. The answers are shown in Table 5.8.

Answer	**Observations**
No, I did not understand anything	0
No, but I understood most of it	1
Yes, I understood enough to answer the questions	5
Yes, I got a very good understanding	1

Table 5.8: Participants opinion about the pre-experiment lecture

Chapter 6

Data Analysis and Discussion

This chapter is divided into two main parts; **Data Analysis** and **Discussion**. The Data analysis part is further divided into analysis of the performance based measures in Chapter 6.1, and analysis of the perception based measures in Chapter 6.2. In the last chapter, Chapter 6.3, the results from the analysis are discussed.

6.1 Analysis; Performance Based Measure

In this chapter the results from the 6 groups of questions from the experiment are analyzed (the results are fond in Chapter 5.1) . The first part, Chapter 6.1.1, provides the analysis from the QGs concerning generalization relations, QG1 and QG4. The second part, Chapter 6.1.2, provide analysis from QG2 and QG4, that address direct connections. In the last part, 6.1.3, the QGs concerning indirect connections, QG3 and QG6, are analyzed.

The analysis methods used here are F-test and t-test [14] [7] [17]. The F-test is used to compare the variance of the two independent samples. For the F-test I use two hypotheses;

- **Hy:** The answers from the Leveled Data Model and the ordinary UML have the same variance
- **Hn:** The answers from the Leveled Data Model and the ordinary UML have different variance

Based on the results from the F-test, one of two alternative t-tests is used to reject or accept the null hypothesis, H0. One of the t-tests is assuming equal variance, the other is assuming unequal variances. A two tailed version of both t-tests is used. The

confidence interval is set at the 0.05 level of significance in all t-tests. The Alternative hypothesis, from Chapter 4.5, are:

- H0: Participants will make the same amount of errors using the Leveled Data Model as using the ordinary UML model
- H1: Participants will make fewer errors using the Leveled Data Model than using the ordinary UML model
- H2: Participants will make fewer errors using the ordinary UML model than using the Leveled Data model

6.1.1 Generalization Relations

	Q1	**Q4**
F	0	0,8269
P(F<=f) one-tail	0	0,4120
F Critical one-tail	0,0522	0,1047

Table 6.1: F-Test Two-Sample for Variances, QG1 and QG4

The results from the F-test are shown in Table 6.1. For QG1 the F-test can not reject Hy, a t-test assuming equal variance is therefore used to test H0 for QG1. For QG4 the F-test rejects Hy, therefore Hn is chosen, and a t-test assuming unequal variance is used to test H0 for QG4.

	Q1	**Q4**
t Stat	0,8452	2,8871
P(T<=t) two-tail	0,4366	0,0447
t Critical two-tail	2,5706	2,7765

Table 6.2: t-Test, QG1 and QG4

The results from the t-tests are shown in Table 6.2. For QG1 it is impossible to reject H0. For QG4, H0 is rejected, and H2 is accepted.

6.1.2 Direct Connections

The results from the F-test are shown in Table 6.3. For QG2 the F-test can not reject Hy, a t-test assuming equal variance is therefore used to test H0 for QG2. For QG5

	Q2	**Q5**
F	0	0,25
P(F<=f) one-tail	0	0,1424
F Critical one-tail	0,0522	0,1047

Table 6.3: F-Test Two-Sample for Variances, QG2 and QG5

	Q2	**Q5**
t Stat	1,3241	0,2294
P(T<=t) two-tail	0,2428	0,8333
t Critical two-tail	2,5706	3,1824

Table 6.4: t-Test, QG2 and QG5

the F-test rejects Hy, therefore Hn is chosen, and a t-test assuming unequal variance is used to test H0 for QG5.

The results from the t-tests are shown in Table 6.4. It is impossible to reject H0 for both Q2 and Q5.

6.1.3 Indirect Connections

	Q3	**Q6**
F	1,2881	1
P(F<=f) one-tail	0,3946	0,4648
F Critical one-tail	9,5521	0,1047

Table 6.5: F-Test Two-Sample for Variances, QG3 and QG6

The results from the F-test are shown in Table 6.5. For QG3 the F-test can not reject Hy, a t-test assuming equal variance is therefore used to test H0 for QG3. For QG6 the F-test rejects Hy, therefore Hn is chosen, and a t-test assuming unequal variance is used to test H0 for QG6.

The results from the t-tests are shown in Table 6.6. It is impossible to reject H0 for both Q3 and Q6.

	Q3	**Q6**
t Stat	2,1900	1,2372
P(T<=t) two-tail	0,0801	0,2837
t Critical two-tail	2,5706	2,7765

Table 6.6: t-Test, QG3 and QG6

6.2 Analysis; Perception Based Measures

Because the participants meanings was quite unanimous in this part of the experiment the results can be interpreted using only **descriptive statistics** [17]. This means that the results are plotted in a reasonable way, and the central tendency is visualized. In Chapter 6.2.1 the preferred model of the participants is analyzed. Chapter 6.2.2 address the participants meaning about the Leveled Data Model. The last chapter, Chapter 6.2.3, address the question of training in the Leveled Data Model. The hypothesis , from Chapter 4.5, used here are:

- H3: Participants think that it is the same which model they use
- H4: Participants prefer to use the Leveled Data Model over the ordinary UML model
- H5: Participants prefer to use the ordinary UML model over the Leveled Data Model
- H6: Participants think that the Leveled Data Model is an unreasonable way to present a large UML model
- H7: Participants think that the Leveled Data Model is a reasonable way to present a large UML model

6.2.1 Participants Preferred Model Type

The participants were asked which model type they found easiest to use. The answers are visualized in Figure 6.1. It can be seen from the figure that H3 has to be rejected, and H5 has to be accepted. In other words; the participants found the ordinary UML diagram easiest to use for the experimental tasks. It should not be necessary to provide any statistical tests to prove this.

6.2.2 Participants Opinion about the Leveled Data Model

All participants agreed that there is a need for decomposition of large UML diagrams. They also agreed that the Leveled Data Model seems like a reasonable way of doing this. Therefore hypothesis H6 is rejected, and H7 is accepted.

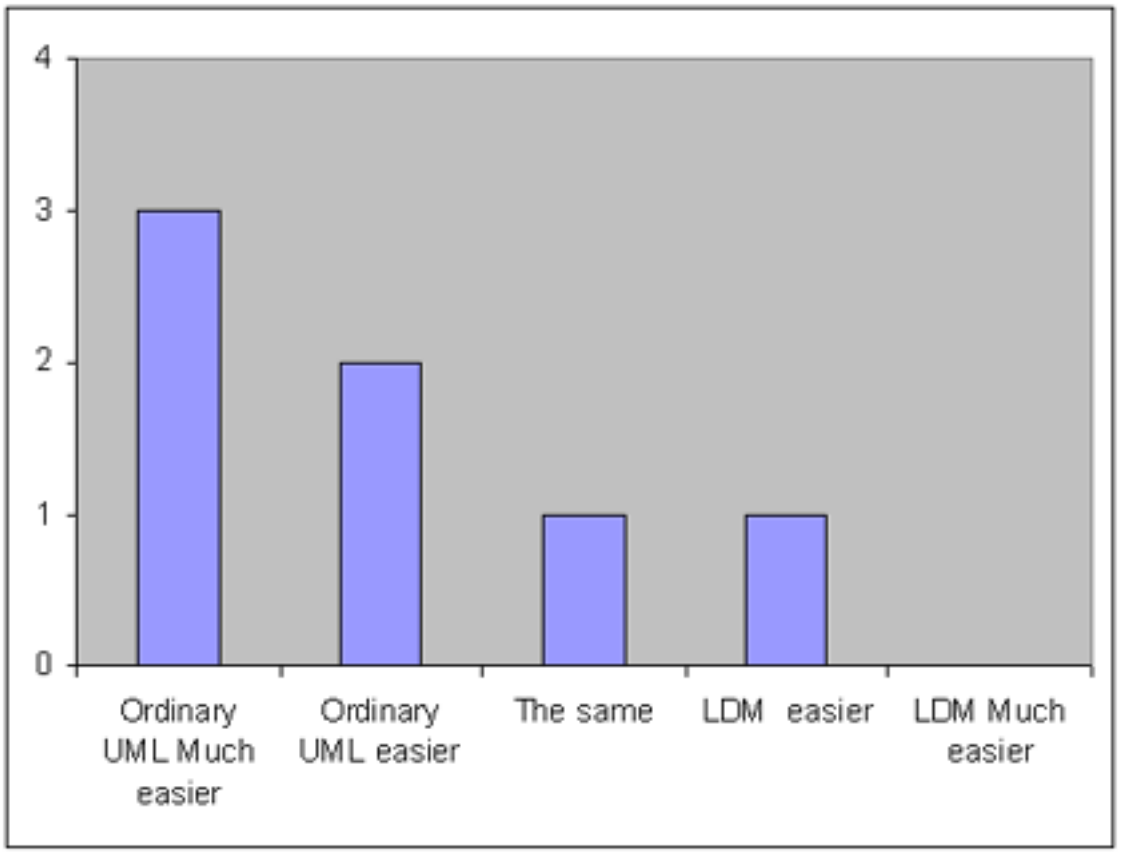

Figure 6.1: Model type preferred by participants

6.2.3 Participants opinion about the pre-experiment lecture

The participants were asked if they thought they got enough training in the Leveled Data Model. Using the results shown in Figure 6.2, I conclude that there is agreement that they did.

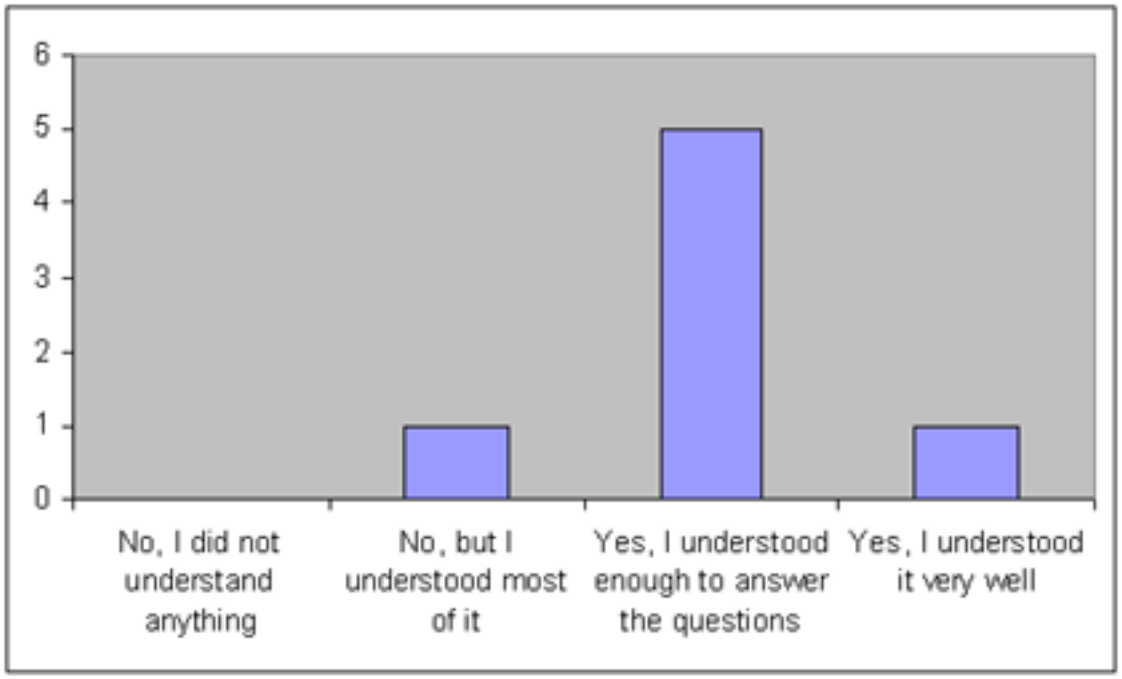

Figure 6.2: Participants opinion about the pre-experiment lecture

6.3 Discussion

I start this discussion with a summary of the results from the analysis. In the rest of this chapter I will discuss these points. The summary is numbered, this is only to refer to the points more easily in the discussion, not to apply importance.

6.3.1 Summary of Results

1. The participants made fewer errors on average, using the ordinary UML model than using the Leveled Data model, on all the six groups of questions (Chapter 5.1)
2. On QG4, H0 was rejected and H2 was accepted. This means that the conclusion from this part of the experiment is: Participants will make fewer errors using the ordinary UML model than using the Leveled Data model (Chapter 6.1.1)
3. On all the other QGs H0 could not be rejected. This means that the conclusions from these parts of the experiment are: Participants will make the same amount of errors using the Leveled Data Model as using the ordinary UML model (Chapter 6.1.1 , 6.1.2 and 6.1.3)
4. Participants prefer to use the ordinary UML model over the Leveled Data Model (Chapter 6.2.1)
5. Participants thinks that the Leveled Data Model is a reasonable way to present a large UML model (Chapter 6.2.2)
6. The participants thought they got enough training in the Leveled Data Model (Chapter 6.2.3)

6.3.2 Number of Errors

Even though the analysis only found a significant difference in one of the QGs, it is a fact that the participants using the ordinary UML diagram made fewer errors than those using the Leveled Data model, in every part of the experiment. This does not necessarily mean that the ordinary UML is always better. There can be several other reasons for this result. The reasons I believe to be most likely is;

- Experience with the model type
- The type of questions in the experiment
- The models used

- The participant's motivation

This is discussed in more detail in the following.

Experience with the model type

The participants thought they got enough training in the Leveled Data model, point 6 in the summary, but they have not actually worked with this type of models before. All the participants have used ordinary UML in several compulsory subjects through their study. It is possible that the participants would have got fewer errors using the Leveled Data Model if they had more training with this type of models in advance.

The type of questions in the experiment

All the questions in the experiment was about verifying or falsifying some kind of connections. For such tasks both model types have both advantages and disadvantages. For the ordinary UML diagram, the main task is to locate the class that starts the connection. Once this is done, it should be quite easy to find the valid connections. For the Leveled Model locating one class is easy, because of the alphabetic index. Here the main task would be to follow the connections. This is easy for connections inside one cluster, but it takes more effort if the connection goes through several clusters.

In a real system development project there are lots of different tasks that have to be done using a model. I could have used other kinds of questions in this experiment, but there are some good reasons why I used the kind of questions I did. The main reasons are listed here:

- The answers are easy to check. Either the answer is right, or the answer is wrong. There is nothing in between
- Both model types have both advantages and disadvantages when used for this kind of questions. None of the models are favored
- Tasks similar to this will have to be done many times as part of bigger tasks in a real project

Another thing, that might affect the results, is that I used the ordinary UML diagram while making the questions. This was a deliberate choice. There would always have been a possibility to affect the results, and I think the risk would have been bigger if I used the Leveled Data Model.

The models used

There are many things about the models that can affect the results. I found it very hard to get a suitable model for the experiment. The one that was used is the best I could find, but the model has some properties that are not optimal for this experiment. The "negative" properties of the model are listed here:

- **Amount of generalization relations in the model;** 55 of 95 relations are generalization, this is far too much
- **The size of the model;** The model consists of 89 classes and 3 interfaces. This could have been a reasonable size. But because of the great share of generalizations, and because there are 12 classes in the diagram that are not connected, it is too small
- **The model is made by students;** It would have been better to have a diagram from a real commercial project

The Leveled Data Model could have been different. The decomposition was done using the algorithm proposed by Moody and Flitman [12]. I also tried to follow the principles for a good decomposition [6]. But there are still a lot of choices to take, and it is not certain that the Leveled Model used here is the optimal decomposition of this model. An empirical evaluation of the Leveled Data Model is presented in Chapter 3.3.

The participant's motivation

It is essential for the outcome of an experiment that the participants make their best effort on all tasks. There was a reward of 50 NOK for the four participants that had the most correct answers. Even though this is not a high sum, I think this made the participants motivation higher. I also think that the fact that they were paid to participate, and that they were observed during the experiment increased their effort. From the observer's point of view, it seemed like all the participants made their best effort throughout the experiment.

6.3.3 Generalization Connections

On the first group of questions concerning Generalization Connections the two model types came out the same. On the second, the group using the ordinary UML diagram scored significantly better. The questions in these two groups were quite similar, the main difference was the time they could use. On the first QG, six of the seven participants had all the answers correct. This indicates that the time limit was too high.

Because of this, I think the answers from the second QG are more relevant. I therefore conclude that an ordinary UML diagram is easier to use for hierarchical structures than a leveled Data Model. This largely agrees with the conclusion from the preceding project [6]. There I used an empirical approach, and found that the Leveled Data Model is not a suitable way to decompose tree structures.

6.3.4 Direct and Indirect Connections

On the questions concerning direct and indirect connections the use of the two different models scored the same. Using the Leveled Data Model did not seem to be better for any of the QGs. Considering the participants experience with the different model types, this is still a pretty good result for the Leveled Data Model. It proves that the Leveled Data Model can be used in practice. It also proves that this model is easy to learn for someone that now UML in advance. It would be interesting to run a big experiment, where the two different models were used for a real project.

6.3.5 Perception Based Measures

Even though the participants think that the Leveled Data Model is a reasonable way to present a large UML model, they preferred to use the ordinary UML model over the Leveled Data Model. These results are not surprising, considering the discussion about the performance based measures in this chapter. It seems like the perception reflects the performance. For obvious reasons, I think the perception is also affected by the same reasons as the performance.

Chapter 7

Conclusions and Further Research

This chapter contains the conclusions from the experiment, and suggestions for further research. In Chapter 7.1 the conclusions of the experiment is listed, and strengths and limitations of the research is discussed. Chapter 7.2 describes some suggestions for further research.

7.1 Conclusions

The implications of this study should be considered in light of its limitations. First, like all laboratory experiments, generalizations should be made cautiously. Second, the results are for only one experiment and a small sample. Replications would be desirable. Strengths and limitations of the research are further discussed in Chapter 7.1.2. The conclusions from the experiment is listed in Chapter 7.1.1

7.1.1 Summary of Findings

This experiment has conducted an empirical comparison of the efficiency (both actual and perceived) between an ordinary UML diagrams and a Leveled Data Model. The conclusions from this experiment are listed here:

- The ordinary UML diagram is more efficient to use, than the Leveled data model, for large generalization hierarchies
- The two different types of diagrams are equally efficient to use for ordinary connections
- People that know UML can use the Leveled Data Model efficient after only a brief introduction

- The participants found the ordinary UML diagram easier to use than the Leveled Data model for the case in the experiment
- The participants perceived the Leveled Data Model to be a reasonable way to decompose large UML diagrams

7.1.2 Strengths and Limitations of the Research

Internal Validity

To guard against problems of internal invalidity, all variables other than the independent variable should be held constant between groups [17]. The following variables were controlled as part of this experiment:

- **Participant Characteristics:** individual differences between participants in different experimental groups were controlled by the randomization procedure.
- **Task Complexity:** the same data model was used by participants in both groups for all tasks.
- **Training:** the same amount of training and similar training materials was given to each experimental group.
- **Experimental Setting:** the location, time of day, time of year, the experimenter and instructions given to subjects were consistent across experimental groups.

External Validity

The greatest threat to the generalisability of the findings of this study was the use of students as experimental subjects. There were also very few participants, and they came from a quite uniform group; they were all from the same class on the same curriculum.

In general, the population from which one selects subjects for the experiment should be representative of the population to which the researchers wishes to generalize results. Because the participants in this study had completed several system development units, and were about to enter the workforce, they were considered as reasonable proxies for practitioners. Most previous experimental studies on data modeling have used undergraduate students as proxies for analysts (e.g. [4], [8], [10], [9]). However, clearly their level of knowledge and expertise in UML modeling would be significantly less than practitioners (the target population). While level of expertise and experience may have affected overall performance on the task, the fact that this was equalized between experimental groups means that comparative findings should still be valid.

7.2 Further Research

As discussed under External Validity, the greatest weakness of this experiment was the nature of the sample population used. For this reason, it would be useful to conduct a field experiment using experienced practitioners as a check on the external validity of the results of this experiment.

It would also be useful to conduct a new experiment quite similar to this, but with some differences;

- **More participants:** The ideal would be to use experienced practitioners, but final year IT master students is a good substitute.
- **A different model type:** There should be less generalization relations, and more ordinary relations than there is in the model used here. A model from a real industrial project would be preferable.
- **More training:** There should be more training in the Leveled Data Model before the experimental tasks. There could e.g. be a tutorial, and some practice questions for each model type.
- **Other experimental tasks:** In addition to questions, there should be some more practical tasks that the participants had to solve, using the different model types.

An experiment, like the one sketched above, would require more time and resources than the experiment described in this paper. The advantage would be that the results would be more generalisable.

It would also be interesting to develop tool support for the Leveled Data Model, and run an similar experiment on computers instead of on paper. It is very likely that the Leveled Data Model would have more advantages compared to ordinary UML on the computer screen than on paper. The reason is that one page in the Leveled Data Model would fit the size of the screen, but it is impossible to see the whole large UML diagram in a readable size on the screen.

Appendix A

Principles for a Good Decomposion

There have been done much research on the subject of what makes a decomposition good. Wand and Weber have written several articles on the subject, like [15] and [16], with a very theoretical approach. They especially stress the principles of loose coupling (see Chapter A.8) and high cohesion (see Chapter A.9), but they also address many of the other principles explained later in this chapter. Burton-Jones and Meso [1] supports Wand and Webers' theories through an empirical test. Carter and Freyberg [2] also base their work on the work of Wand and Weber, but they stress the importance of looking at the relation between the principles of coupling and cohesion. Because coupling and cohesion interfere with one another, it is important to look at the ratio between them.

Moody and Flitman states 9 principles for a good decomposition [12]. These are mainly the same principles as those discussed by the articles above. Even if these principles are made for ER-diagrams, most of them are important for decomposition in general. In this chapter I will list these principles. Each subchapter gives a short explanation and a metric for one principle. I will use these principles to evaluate the different models I work on during this project.

A.1 Complete

This principle requires that each entity is included in at least one subject area. This ensures that all entities and relationships in the original model are preserved in the decomposition, so that the decomposition is lossless. This principle should be applied at each level of the hierarchy.

Metric: Union of Subject Areas.
This principle can be verified by ensuring that the union of subject areas at each level of the model equals the set of elements at the next level down.

A.2 Non-Redundant

This principle requires that each entity is included in at most one subject area. This ensures that subject areas form non-overlapping (disjoint) subsets of E and therefore minimizes redundancy. This principle should also be applied at each level of the hierarchy.

Metric: Intersection of Subject Areas.
This principle can be verified by ensuring that the intersection between subject areas at each level of the model is null.

A.3 Self Contained

This principle requires that each subject area forms a fully connected sub graph of the original model (D). This assists understanding by making sure that each subject area forms an integrated whole.

Metric: Fully Connected.
This principle can be verified by ensuring that for any pair of entities on the same subject area, a path exists between them consisting only of internal relationships.

A.4 Unified

Each subject area should be named after one of the entities on the subject area, called the central entity. This helps to ensure that all entities on the subject area relate to a single concept. Identifying appropriate central entities is the key to identifying meaningful subject areas. Central entities act as "nuclei" around which other entities are clustered to form subject areas. Central entities should be chosen as the entities of highest business importance - the "core" business concepts in the model. Of course, business importance is quite a subjective concept, and requires human judgment. However a useful heuristic for identifying central entities is to identify entities with the most relationships. Usually the most highly connected entities are also the most important entities from a business viewpoint.

Metric: Connectivity.
We define the connectivity of an entity as the number of relationships it participates in. The entities with the highest connectivity should therefore be used as central entities. At higher levels of the model (Level 2 and above), central subject areas can also be identified based on their connectivity. The connectivity of a subject area is defined as the number of boundary relationships it has.

A.5 Cognitively Manageable

The maximum size of subject areas as the upper limit of human cognitive capacity (nine concepts). This should be used as the limit at all levels of the model, to ensure that each subject area forms a cognitively "digestible" unit of information.

Metric: Maximum Size of Subject Areas.
The size of a subject area is defined as the number of distinct concepts it contains. (This may be either entities or subject areas).

A.6 Flexible

Data models tend to grow in size over time, as new requirements are added or systems expand in scope. The partitioning of the data model into subject areas should therefore allow flexibility for growth. A data model which consists of subject areas that are all of the maximum size (nine) will have to be repartitioned if even a single entity is added.

Metric 1 : Capacity for Growth.
Flexibility can be measured by calculating the percentage of growth possible before the model needs to be re-partitioned. This is defined as: ((No. of Level 1 subject areas*9) - (No. of entities in E))*100/ (No. of entities in E) As a rule of thumb, at least 20

Metric 2 : Optimal Size of Subject Areas.
The optimal size of subject areas is defined as seven. This allows for growth of two entities per subject area, or about 30

Metric 2.1 : Optimal Number of Subject Areas.
The optimal size of subject areas can be used to calculate the optimal number of subject areas for each level of decomposition. This can be used to guide the decomposition process from the beginning. The optimal number of subject areas at Level n is defined as the number of entities in E divided by 7n, rounded to the nearest whole number. For example, for a model with 125 entities, the optimal number of subject areas at Level 2 will be 3 (125/49 = 2.55).

Metric 2.2 : Optimal Number of Decomposition Levels.
The optimal size of subject areas can also be used to calculate the optimal number of levels of decomposition. This is defined as the number of entities in E log to base 7, rounded down to the next lowest whole number. For example, for a model with 125 entities, two levels will be required (125 log 7 = 2.48).

A.7 Balanced

Each subsystem should be approximately equal in size.

Metric 1 : Standard Deviation in Subject Area Size.
Balancing can be measured by calculating the standard deviation in the size of subject areas.

Metric 2 : Minimum Size of Subject Areas.
The minimum size of subject areas is defined as five concepts.

A.8 Loosely Coupled

Coupling is defined as the strength of interconnections between different subsystems (inter-molecular forces). Coupling is widely accepted to be one of the most important measures of the quality of a decomposition and should be minimized to increase the independence of the parts of the system. In the case of a Leveled Data Model, coupling corresponds to the number and strength of relationships between subject areas (boundary relationships). Minimizing boundary relationships:

- Improves understanding by ensuring that subject areas can be understood independently of each other and reducing the need to navigate between subject areas
- Simplifies documentation by reducing the need to show cross-references between subject areas (via foreign entities)
- Simplifies maintenance by ensuring that subject areas can be maintained relatively independently and by minimizing redundancy between them (by reducing the number of foreign entities)

Metric 1 : Number of Boundary Relationships.
The simplest measure of the coupling of a decomposition is the number of relationships between subject areas (boundary relationships).

Metric 2 : Sum of Boundary Relationships Strengths.
A finer resolution level measure of coupling can be obtained by assigning different weights to different types of relationships to indicate their relative semantic strength. The coupling of the decomposition can then be calculated as the sum of the strengths of boundary relationships.

A.9 Highly Cohesive

The complementary concept to coupling is cohesion, which is defined as the strength of associations within subsystems (intra-molecular forces). Cohesion should be maximized in order to increase independence of subsystems. Subsystems which are highly cohesive are likely to be more independent of each other. Also, subsystems that are highly cohesive will be easier to understand because they can be encoded as a single integrated "chunk" of information rather than a number of relatively independent "chunks".

Metric 1 : Number of Internal Relationships.
The simplest measure of the cohesion of a decomposition is the number of relationships within subject areas (internal relationships).

Metric 2 : Sum of Internal Relationships Strengths.
As with coupling, a finer resolution measure of cohesion can be obtained by assigning different weights to different types of relationships to indicate semantic "strength". The cohesion of the decomposition can then be calculated as the sum of the strengths of internal relationships.

Appendix B

Original UML Diagram

This appendix contains the original UML diagram.

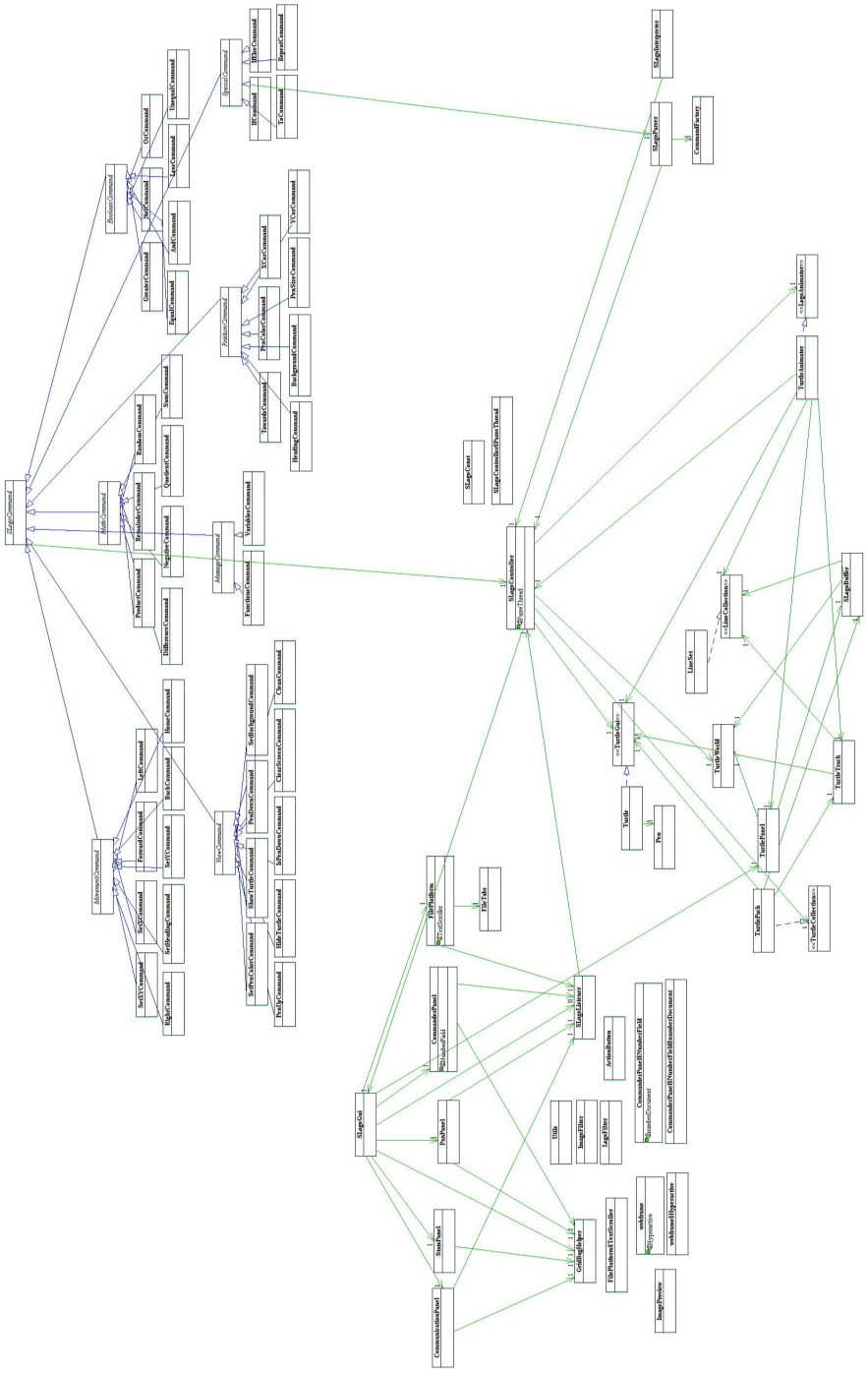

Figure B.1: Original UML diagram

Appendix C

Decomposition Level 1

This appendix contains the first level of decomposition.

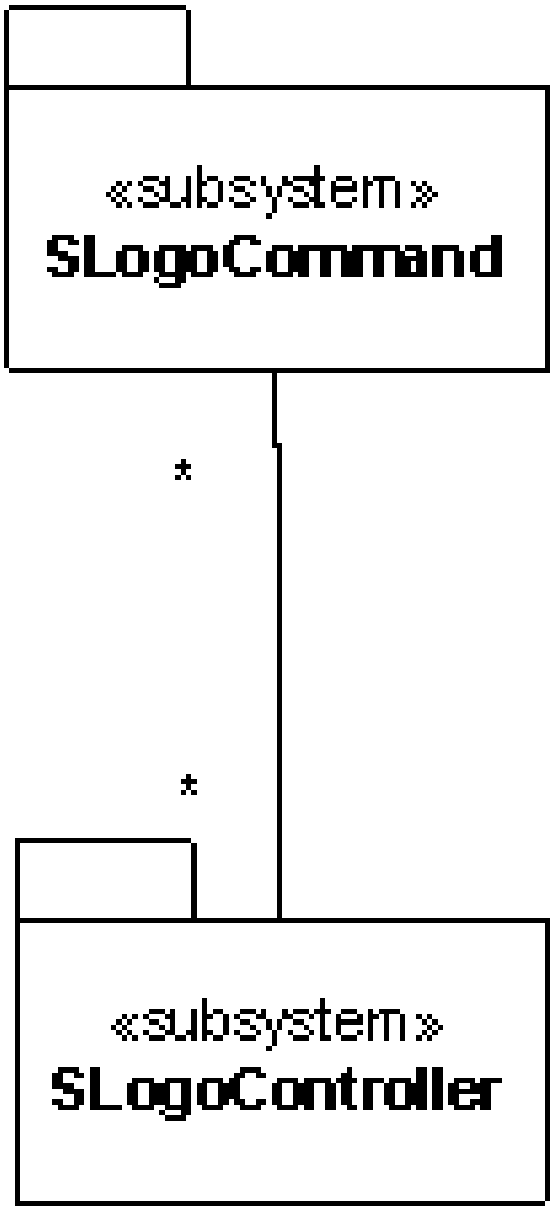

Figure C.1: Decomposition Level 1

Appendix D

Decomposition Level 2

This appendix contains the clusters from the second level of the decomposition of the SLogo diagram in Figure B.1. These diagrams are the second level in the leveled model, the first level is shown in Figure C.1.

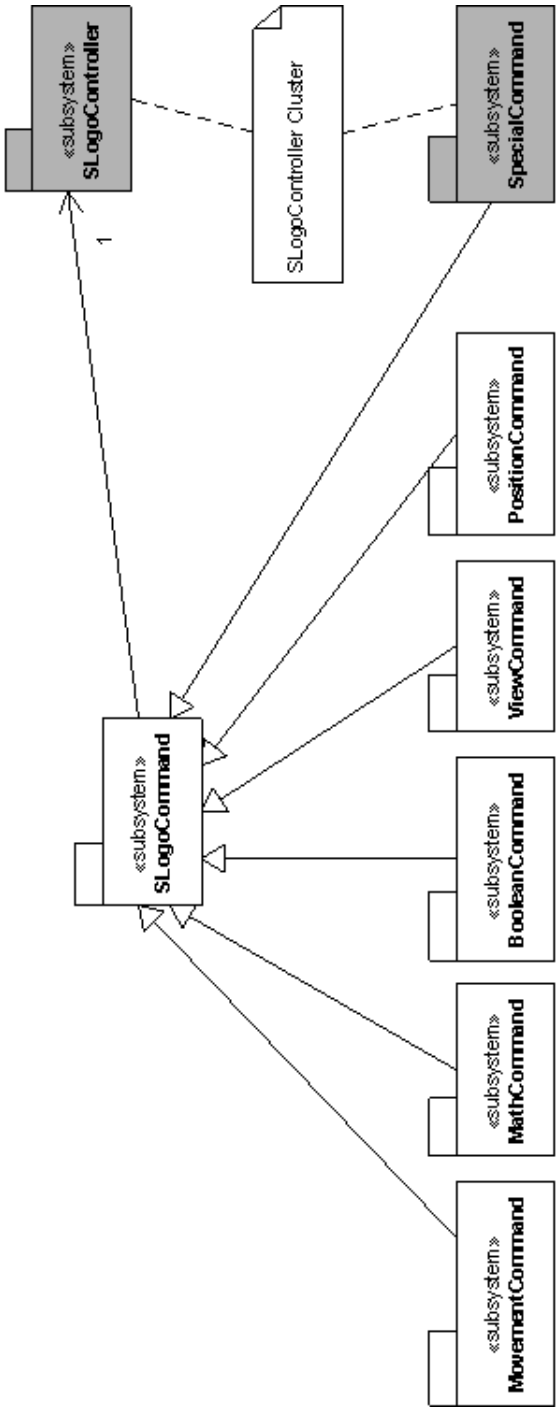

Figure D.1: Decomposition Level 2; SLogoCommand Cluster

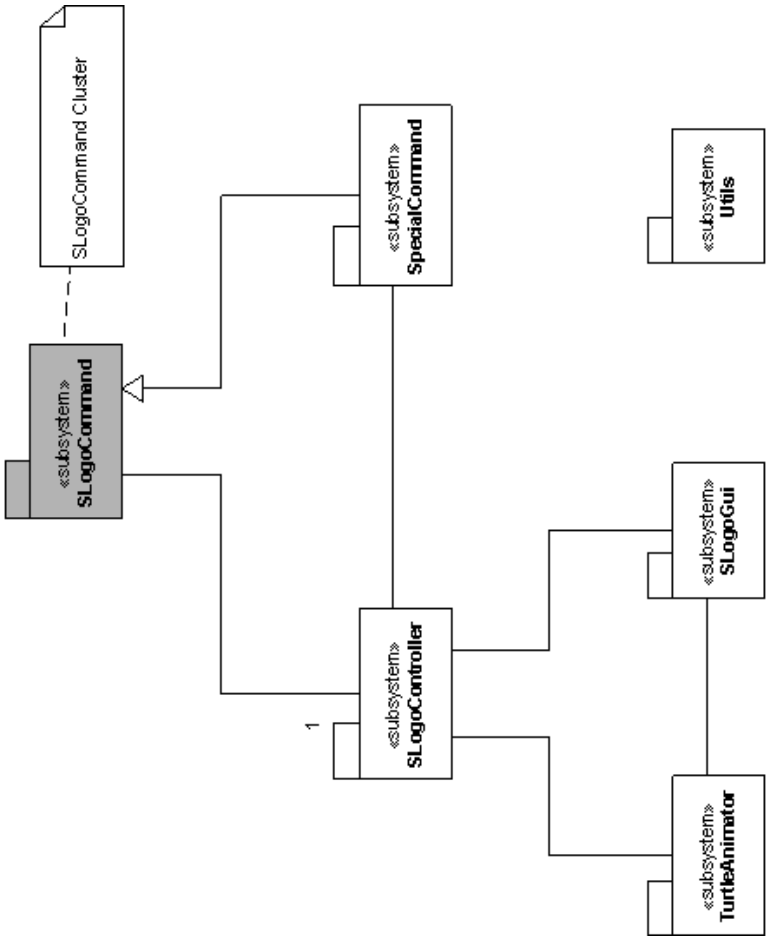

Figure D.2: Decomposition Level 2; SLogoController Cluster

Appendix E

Decomposition Level 3; SLogoCommand Cluster

This appendix contains the clusters from the decomposition of the SLogoCommand diagram in Figure D.1. These diagrams are on the third level in the leveled model, the first level is shown in Figure C.1.

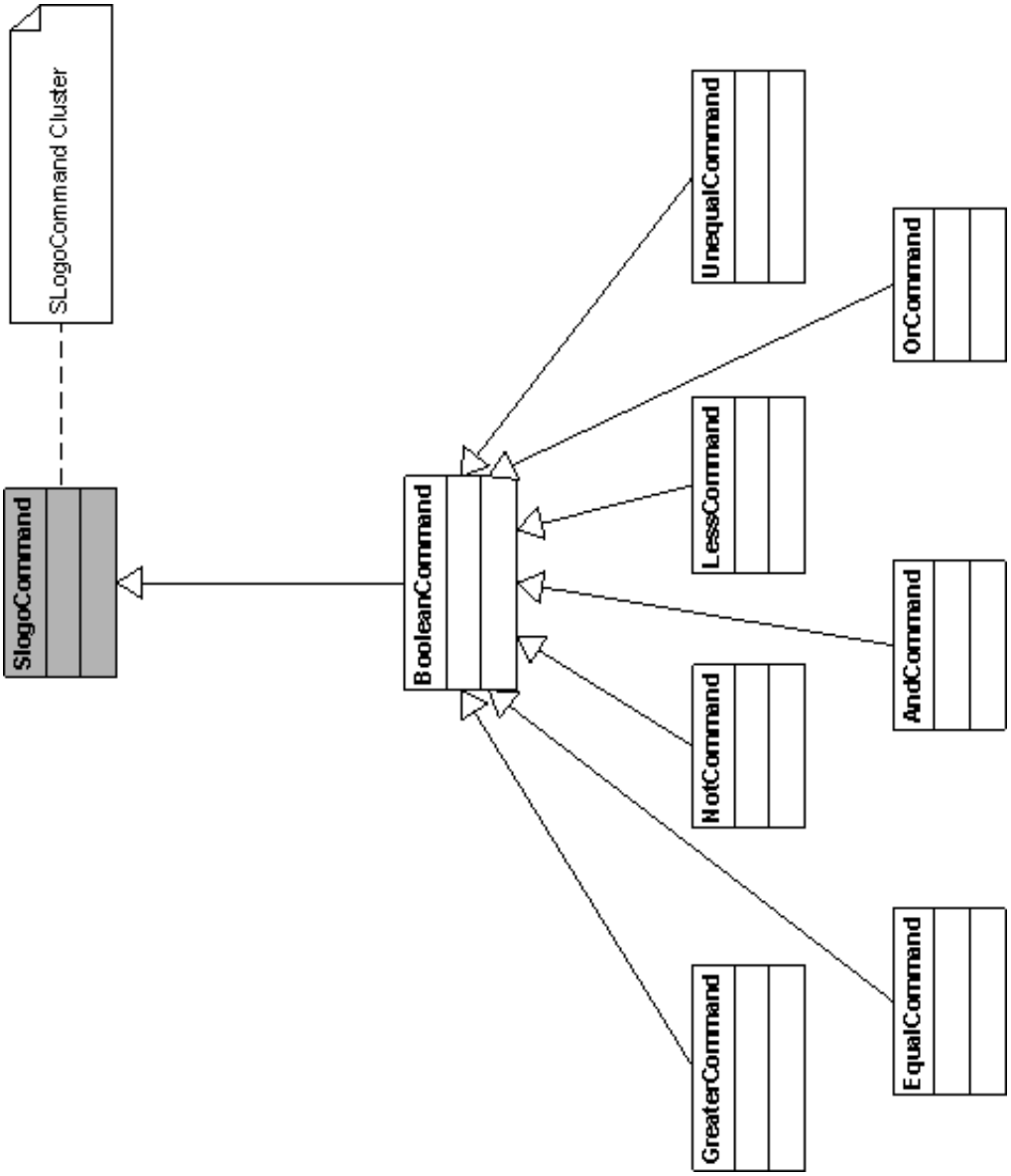

Figure E.1: Cluster - BooleanCommand

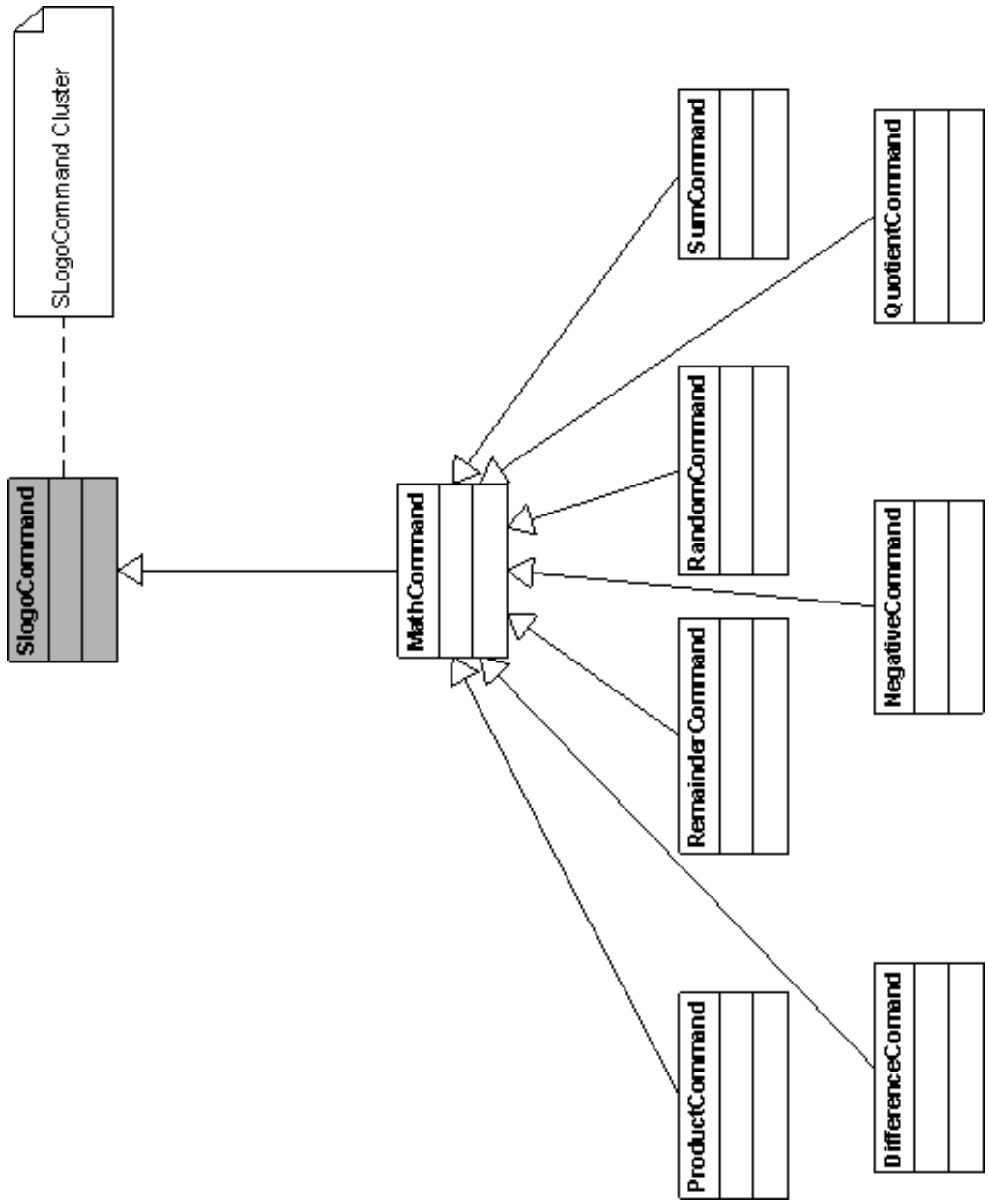

Figure E.2: Cluster - MathCommand

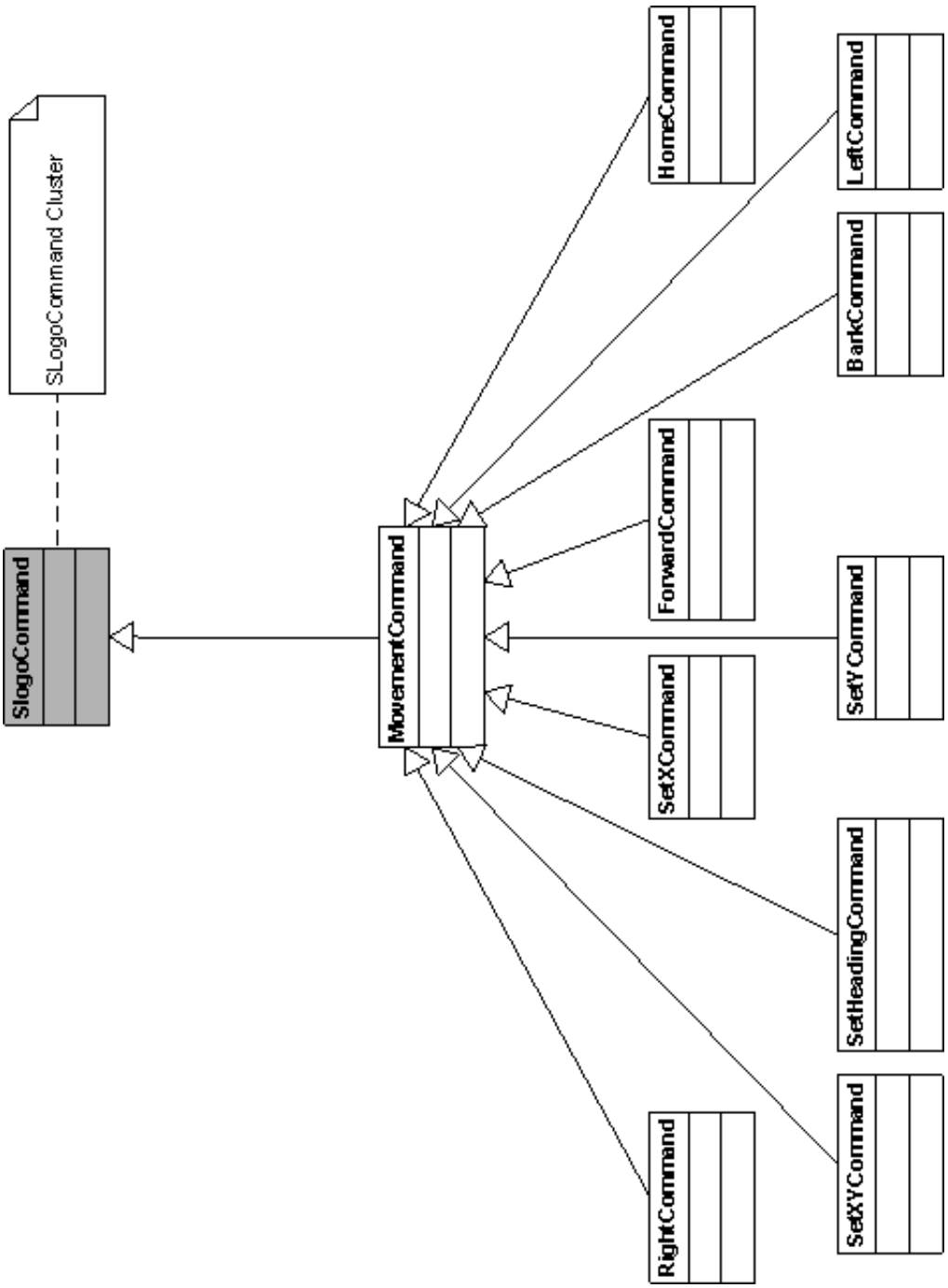

Figure E.3: Cluster - MovementCommand

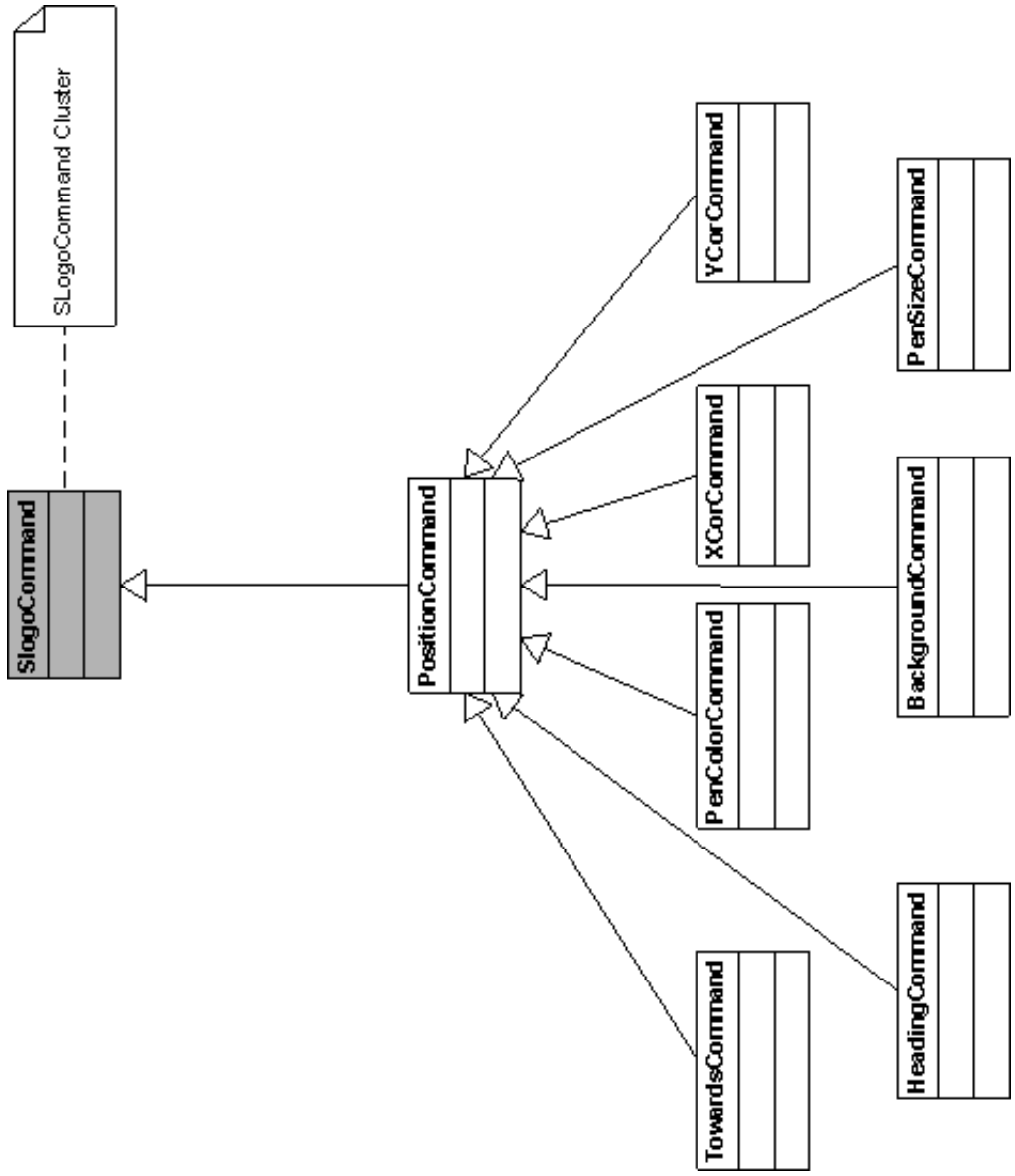

Figure E.4: Cluster - PositionCommand

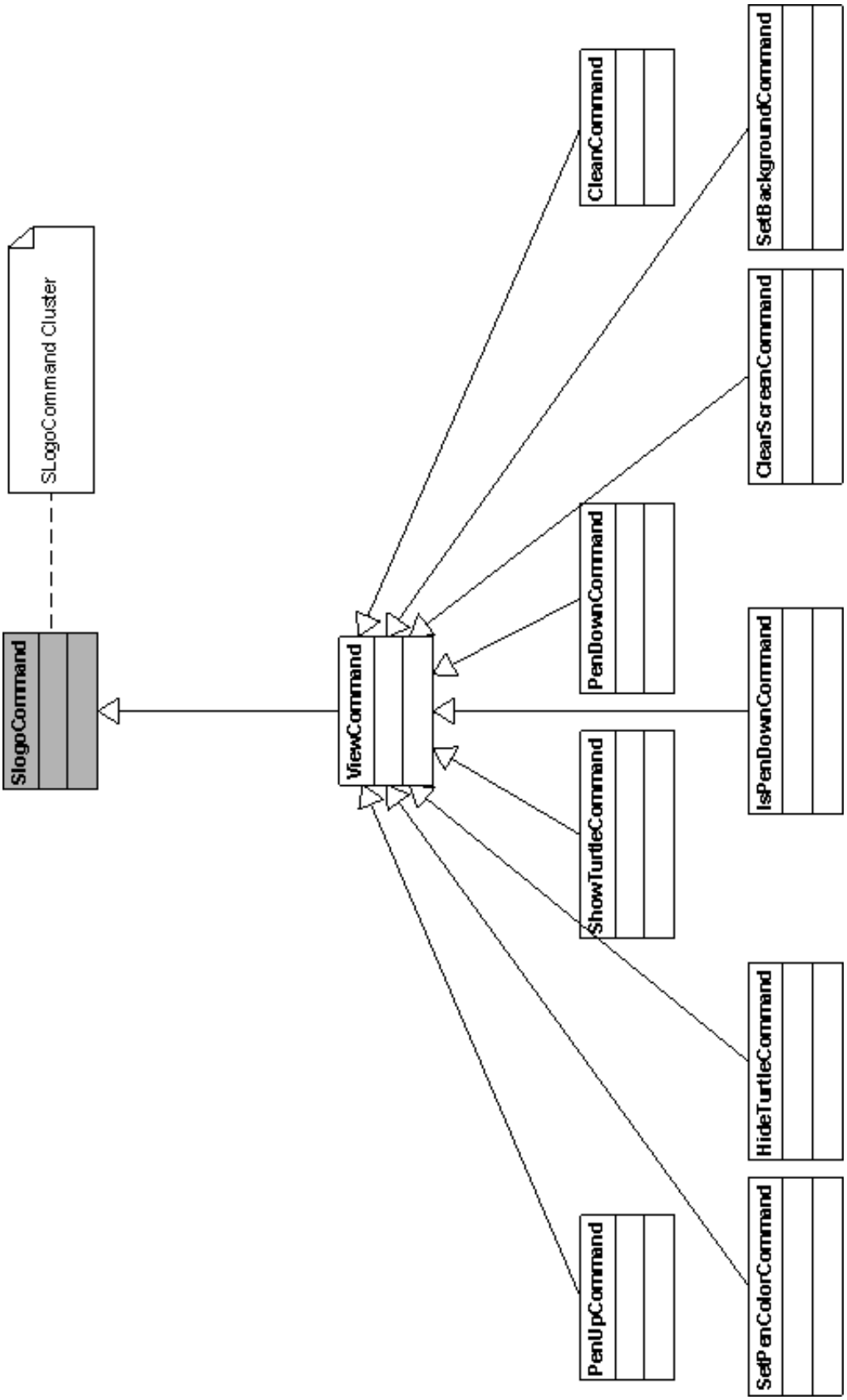

Figure E.5: Cluster - ViewCommand

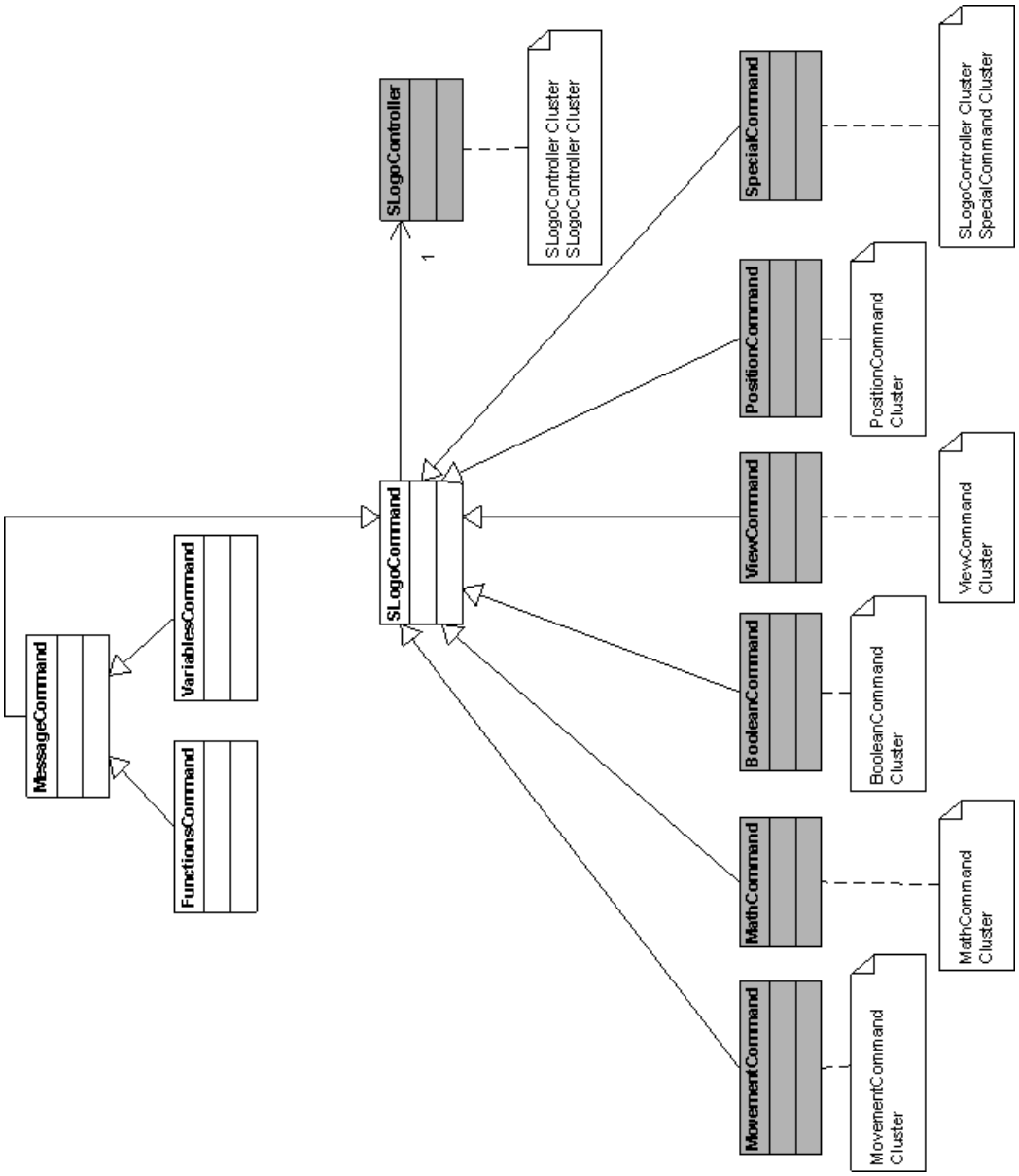

Figure E.6: Cluster - SLogoCommand

Appendix F

Decomposition Level 3; SLogoController Cluster

This appendix contains the clusters from the decomposition of the SLogoController diagram in Figure D.2. These diagrams are on the third level in the leveled model, the first level is shown in Figure C.1.

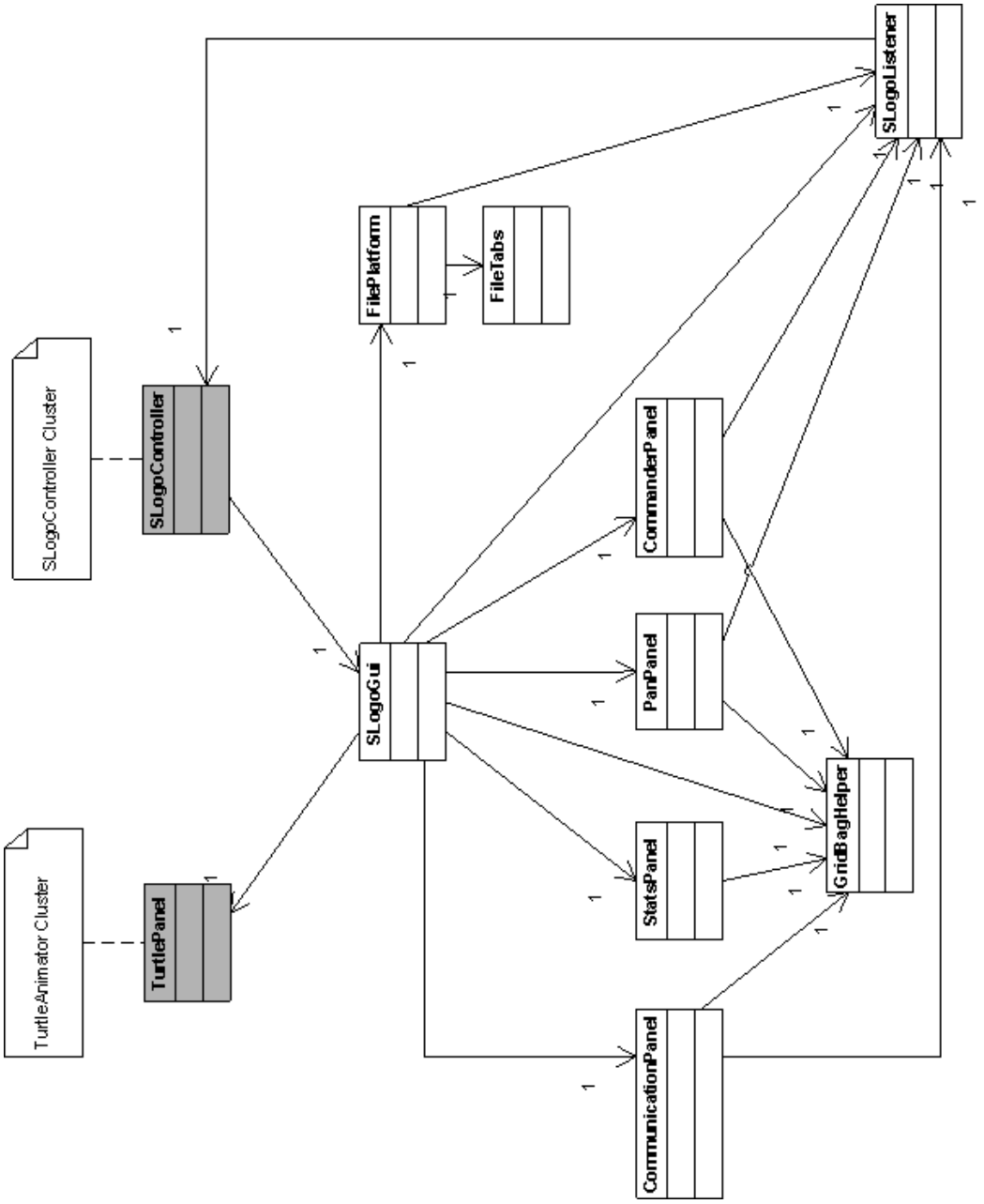

Figure F.1: Cluster - SLogoGui

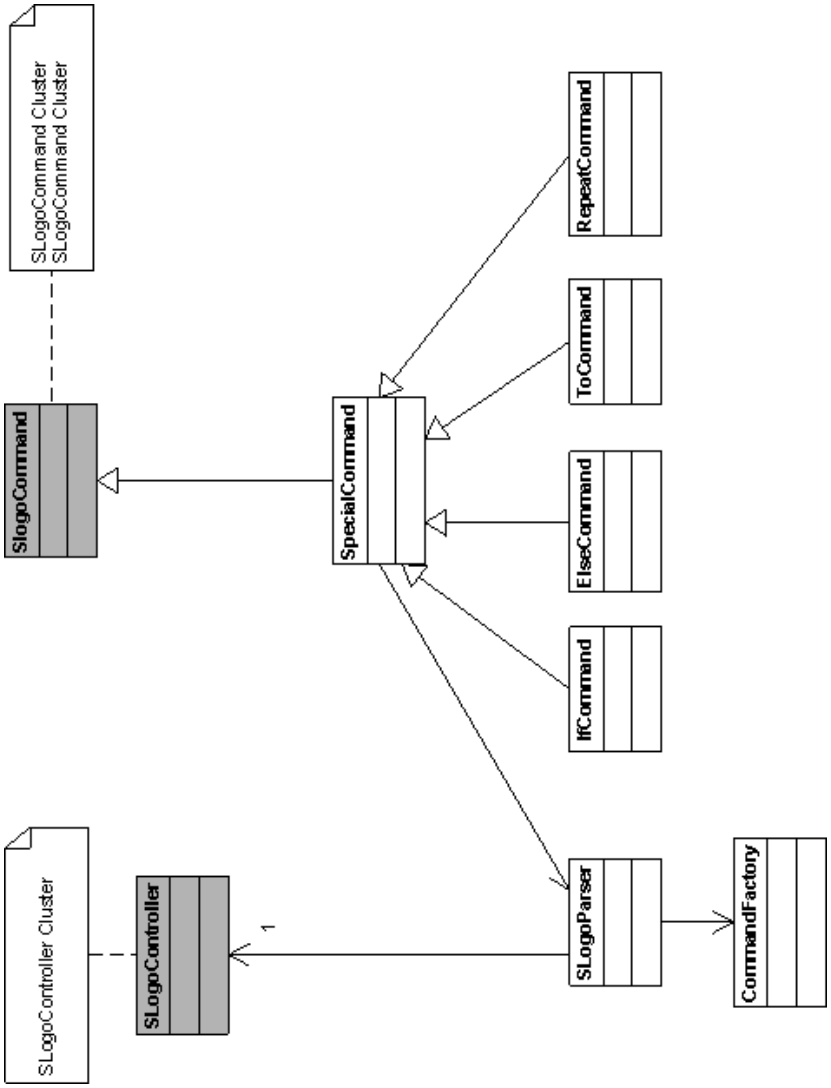

Figure F.2: Cluster - SpecialCommand

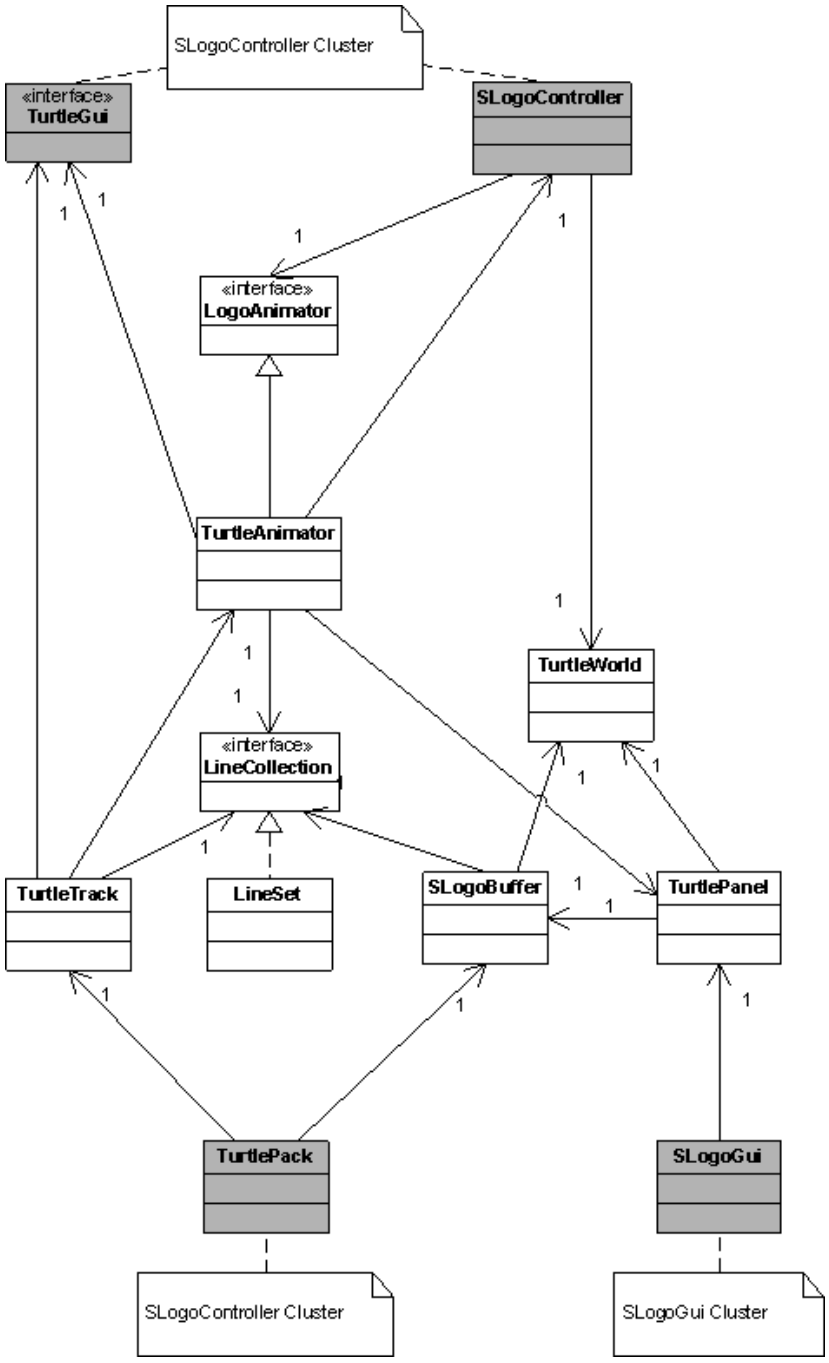

Figure F.3: Cluster - TurtleAnimator

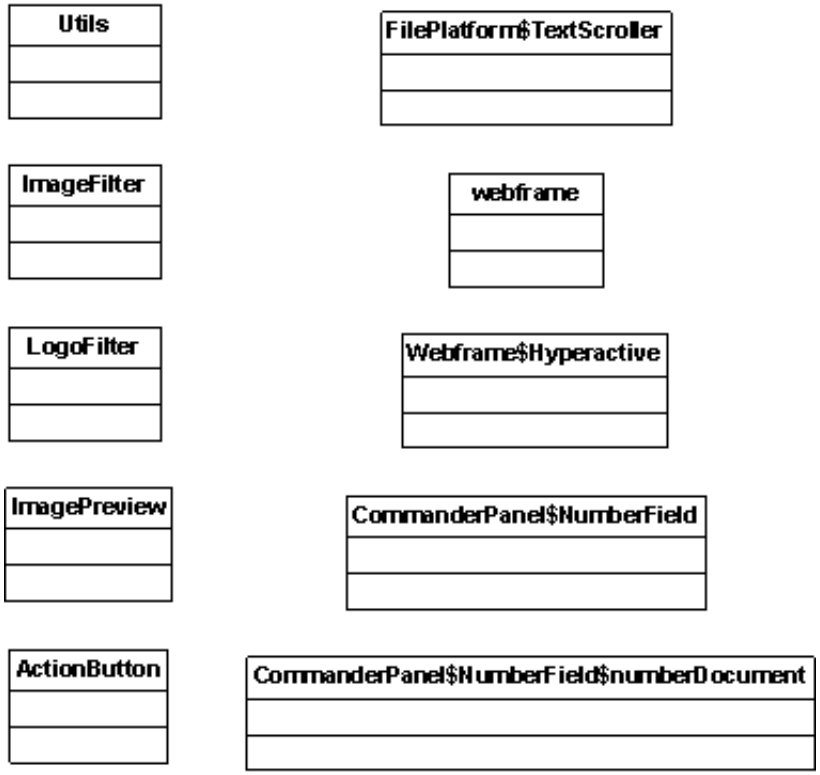

Figure F.4: Cluster - Utils

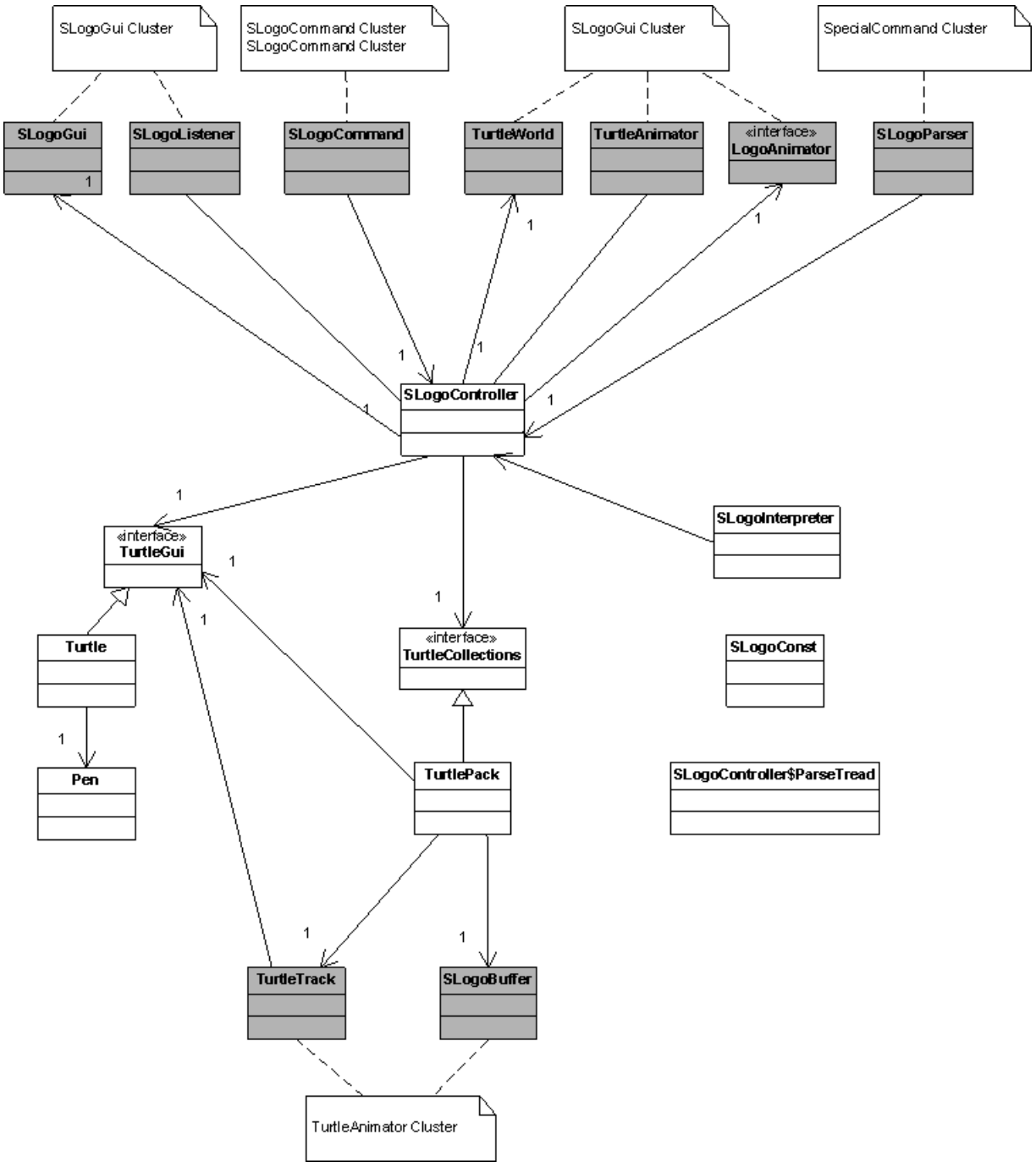

Figure F.5: Cluster - SLogoController

Appendix G

Sorted List

This appendix contains sorted lists of the classes in the decomposed diagram. Table G.1 and Table G.2 shows the clusters of the SLogoCommand supercluster, and Table G.3 shows the clusters of the SLogoController supercluster. For each cluster, there is a reference to the appendix where the figure can be found.

Super Cluster	**Cluster**	**Class**
SLogoCommand	BooleanCommand	AndCommand
D.1	E.1	BooleanCommand
		EqualCommand
		GreaterCommand
		LessCommand
		NotCommand
		OrCommand
		UnequalCommand
	MathCommand	DifferenceCommand
	E.2	MathCommand
		NegativeCommand
		ProductCommand
		QuotientCommand
		RandomCommand
		RemainderCommand
		SumCommand

Table G.1: SLogoCommand - Sorted List

Super Cluster	**Cluster**	**Class**
SLogoCommand	MovementCommand	BackCommand
D.1	E.3	ForwardCommand
		HomeCommand
		LeftCommand
		MovementCommand
		RightCommand
		SetHeadingCommand
		SetXCommand
		SetXYCommand
		SetYCommand
	PositionCommand	BackGroundCommand
	E.4	HeadingCommand
		PenColorCommand
		PenSizeCommand
		PositionCommand
		TowardsCommand
		XCorCommand
		YCorCommand
	SLogoCommand	FunctionsCommand
	E.6	MessagesCommand
		SlogoCommand
		VariablesCommand
	ViewCommand	CleanCommand
	E.5	ClearScreenCommand
		HideTurtleCommand
		IsPenDownCommand
		PenDownCommand
		PenUpCommand
		SetBackgroundCommand
		SetPenColorCommand
		ShowTurtleCommand
		ViewCommand

Table G.2: SLogoCommand - Sorted List

Super Cluster	**Cluster**	**Class**
SLogoController	SLogoController	Pen
D.2	F.5	SLogoConst
		SLogoController
		SLogoController$ParseTread
		SLogoInterpreter
		Turtle
		TurtleCollections
		TurtleGui
		TurtlePack
	SLogoGui	CommanderPanel
	F.1	CommunicationPanel
		FilePlatform
		FileTabs
		GridBagHelper
		PanPanel
		SLogoGui
		SLogoListener
		StatsPanel
	SpecialCommand	CommandFactory
	F.2	ElseCommand
		IfCommand
		RepeatCommand
		SLogoParser
		SpecialCommand
		ToCommand
	TurtleAnimator	LineCollection
	F.3	LineSet
		LogoAnimator
		SLogoBuffer
		TurtleAnimator
		TurtlePanel
		TurtleTrack
		TurtleWorld
	Utils	ActionButton
	F.4	CommanderPanel$NumberField
		CommanderPanel$NumberField$numberDocument
		FilePlatform$TextScroller
		ImageFilter
		ImagePreview
		LogoFilter
		Utils
		webframe
		Webframe$Hyperactive

Table G.3: SLogoController - Sorted List

Bibliography

[1] Andrew Burton-Jones and Peter Meso. How good are these uml diagrams? an empirical test of the wand and weber good decomposition model. *Proceedings of the 23rd International Conference on Information Systems 2002, L. Applegate, R. Galliers, and J. DeGross, Eds. Barcelona, Spain,*, December 2002.

[2] Michael R. Carter and Chris A. Freyberg. Coupling and incoherence of a decomposition. *Proceedings of the Information Systems Foundations Workshop, Ontology, Semiotics and Practice*, January 1999.

[3] Revital Danoch, Peretz Shoval, and Mira Balabaan. Hierarchical evolution of entity-relationship diagrams - a bottom-up approach. *Proceedings of the Sixth CaiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD'01), Interlaken, Switzerland*, January 2001.

[4] Revital Danoch, Peretz Shoval, and Mira Balabaan. Comprehension of hierarchical er diagrams compared to flat er diagrams. *Information Modeling Methods and Methodologies. Hershey, PA: Idea Group Publishing. ISBN: 1-59140-375-8*, November 2003.

[5] M. Gandhi, E. Robertson, and D. Gucht. Leveled entity relationship model. *In: Proc. of the 13th Intl. Conference on Entity-Relationship Approach (ER'94). Volume 881 of LNCS., Springer-Verlag (1994) 420-436"*, 1994.

[6] Vidar Haugen. Empirical testing of a clustering algorithm for large uml class diagrams, December 2004.

[7] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley and Sons Inc., New York, US, eight edition, 1999.

[8] Daniel L. Moody. Comparative evaluation of large data model representation methods: The analyst's perspective. *Proceedings of the 21st International Conference on Conceptual Modeling Tampere, Finland, pp. 214 - 231*, October 2002.

[9] Daniel L. Moody. Dealing with complexity in information systems modeling: Development and empirical validation of a method for representing large data models.

Proceedings of the International Conference on Information Systems, ICIS 2003, Seattle, Washington, USA. Association for Information Systems 2003, December 2003.

[10] Daniel L. Moody. Entity connectivity vs. hierarchical levelling as a basis for data model clustering: An experimental analysis. *Database and Expert Systems Applications, ISBN: 3-540-40806-1, pp. 77 - 87*, October 2003.

[11] Daniel L. Moody and Guttorm Sindre. Managing complexity in object oriented analysis: Adapting uml for modelling large scale information systems. *Proceedings of the 14th Australasian Conference on Information Systems (ACIS 2003), Perth, Australia*, November 2003.

[12] David L. Moody and Andrew Flitman. A methology for clustering entity relationship models: A human information processing approach. *Proceedings of the 18th International Conferance on Entity-Relationship Approach (pp. 114-130).*, November 1999.

[13] A. H. Seltveit. An approach to information systems modelling based on systematic complexity reduction. *Proceedings of the 29th Annual Hawaii International Conference on System Science*, January 1996.

[14] Ronald E. Walpole, Raymond H. Myers, and Sharon L. Myers. *Probability and Statistics for Engineers and Scientists.* Prentice Hall International Inc., Upper Saddle River, New Jersey, sixth edition, 1998.

[15] Yair Wand and Ron Weber. An ontological model of an information system. *IEEE Transactions on Software Engineering, v.16 n.11, p.1282-1292*, November 1990.

[16] Yair Wand and Ron Weber. A unified model of software and data decomposition. *Proceedings of the twelfth international conference on Information systems, p.101-110, New York, New York, United States*, January 1991.

[17] Claes Wohlin, Per Runeson, Martin Høst, Magnus c. Ohlsson, Bjørn Regnell, and Anders Wesslen. *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers, AH Dordecht, The NETHERLANDS, second edition, 2002.