FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Towards an Understanding of the Practical Use of UML

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Alexander Bohn
Matrikelnummer 0505808

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: O.Univ.Prof. Dipl.-Ing. Mag. Dr.techn. Gerti Kappel
Mitwirkung: Kolleg. Dipl.-Ing. BSc Tanja Mayerhofer

Wien, 28.11.2013

(Unterschrift Alexander Bohn)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Towards an Understanding of the Practical Use of UML

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Alexander Bohn
Registration Number 0505808

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: O.Univ.Prof. Dipl.-Ing. Mag. Dr.techn. Gerti Kappel
Assistance: Kolleg. Dipl.-Ing. BSc Tanja Mayerhofer

Vienna, 28.11.2013

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Alexander Bohn
Stolberggasse 1-3/31, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Alexander Bohn)

Abstract

UML is a standardized modeling language that is used in many application domains. Many companies use UML within their processes, and many UML models exist in different domains. With UML, a variety of systems can be modeled, for example software systems, business processes, and production processes. Since UML is a language that is used for many application domains, the question arises how UML is used in practice. Currently, there are only a few empirical studies about the practical use of UML. None of these studies has analyzed real world models. In most cases the usage of UML has been analyzed by surveys or investigating current UML literature and UML modeling tools.

Therefore, this work is devoted to the research question of how UML is used in practice and tries to answer this question by analyzing real world models.
In particular, the following questions are answered:
Which UML language units are used?
Which UML language concepts are used?
Which UML diagrams are used?
Which relationships are used between UML concepts of different UML language units?
Which UML profiles are used to provide additional information in UML models?

To answer these questions, 92 UML models, which are publicly accessible on the Web and created with the Enterprise Architect (EA) modeling tool, were quantitatively analyzed. EA was chosen as modeling tool because it provides its own API which enables to access the content of a model with script languages such as JavaScript or Visual Basic.

The results gave an insight into the usage of UML. Particularly, it could be determined which of the considered UML concepts and UML diagrams were often used and which were rarely used. Furthermore, the results showed between which UML language units more or less relationships were modeled. UML profiles were used quite commonly, but only a few different UML profiles have been used.

Finally, the master thesis revealed that further evaluations concerning the usage of UML are needed to obtain more reliable data about how UML is used in practice.

Kurzfassung

UML ist eine standardisierte Modellierungssprache, die in vielen Bereichen Anwendung findet. Viele Unternehmen benutzen UML für ihre Modellierungstätigkeiten und man darf annehmen, dass zahlreiche UML Modelle in unterschiedlichsten Domänen existieren. Mit UML können verschiedenste Systeme modelliert werden, seien es beispielsweise Softwaresysteme, Geschäftsprozesse oder Produktionsprozesse. Da UML eine Sprache ist, die für zahlreiche Einsatzgebiete verwendet wird, stellt sich die Frage, wie UML in der Praxis Verwendung findet. Derzeit gibt es nur wenige empirische Studien über die praktische Verwendung von UML. Keine dieser Studien hat sich dabei mit echten Modellen aus der Praxis beschäftigt. Zum Großteil wurden Daten über die Verwendung von UML mittels Umfragen oder aus der gängigen UML Literatur und den verwendeten Modellierungswerkzeugen ermittelt.

Diese Arbeit widmet sich daher der Forschungsfrage, wie UML in der Praxis Verwendung findet, und versucht diese durch die Analyse von Modellen aus der Praxis zu beantworten.
Für die Beantwortung dieser Forschungsfrage wurden folgende Detailfragen definiert:
Welche UML Spracheinheiten werden verwendet?
Welche UML Sprachkonzepte werden verwendet?
Welche UML Diagramme werden verwendet?
Welche Beziehungen werden zwischen UML Sprachkonzepten verschiedener UML Spracheinheiten verwendet?
Welche UML Profile werden verwendet um zusätzliche Informationen in UML Modellen zu erfassen?

Um diese Fragen zu beantworten, wurden 92 UML Modelle, die öffentlich im Web verfügbar sind und mit dem UML Modellierungstool Enterprise Architect (EA) erstellt wurden, automatisiert quantitativ analysiert. EA wurde gewählt, da dieses Modellierungswerkzeug eine eigene API zur Verfügung stellt, mit der Modellelemente mit Scriptsprachen wie JavaScript oder Visual Basic ausgelesen werden können. Die Ergebnisse gaben einen Einblick in die Verwendung von UML. Insbesondere konnte eruiert werden, welche der in der Analyse beachteten UML Sprachkonzepte sowie UML Diagramme häufig bzw. weniger häufig verwendet wurden. Des Weiteren konnte aus den Resultaten festgestellt werden, zwischen welchen UML Spracheinheiten mehr oder weniger starke Beziehungen modelliert wurden. UML Profile wurden zwar durchgehend und relativ häufig verwendet, jedoch wurden nur wenige verschiedene UML Profile verwendet. Weitere Evaluierungen betreffend des UML Sprachgebrauchs sind sicher nötig, um fundiertere Kenntnisse über die Verwendung von UML in der Praxis zu erhalten.

Contents

CHAPTER 1

Introduction

1.1 Motivation

According to the specification of the Unified Modeling Language (UML) [22] by the Object Management Group (OMG)[1] *"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system."* UML combines best practices in software modeling and has become de facto standard for software modeling. UML was originally developed for the specification of object-oriented programs, however, it is now used for a wide variety of purposes such as for instance systems modeling and process modeling. UML is independent from any programming languages, platforms and development tools. It is usable for different domains. Further UML does not support one specific development process, rather it is suitable for any process model chosen. With UML static and behavioral parts of a system can be modeled. Furthermore, with UML Profiles it is possible to extend or restrict parts of UML without changing the UML metamodel [15].

According to Fowler [9], people use UML in three modes: to model sketches, blueprints, or as programming language. Sketches show some aspects of a system in a less detailed way. Their aim is to give an understanding of some parts of a system. In project discussions and project meetings sketches are used in a supportive way to give a general overview about a system or process. Tools for sketches are usually whiteboards or light-weight modeling tools. On the other hand blueprints describe a system as complete as possible. For a programmer it should be possible to construct the whole software according to a blueprint. All needed parts of the final software are highly detailed modeled. For blueprints more sophisticated tools are used than for sketches. Sketches and blueprints can be used in a reverse engineering and forward engineering direction. In case of UML reverse engineering means generating UML models out of code and forward engineering means generating code out of UML models. In reverse engineering the UML models represent existing code and in forward engineering the UML models define

[1] OMG. http://www.omg.org. Accessed: 2013-28-02

exactly how the software is structured and how it behaves so that the source code can be automatically generated. In the third mode UML is used as a programming language. UML models can be compiled directly to running software, so that the UML model is the source code. Model driven architecture (MDA) is an approach using UML as programming language. The standards of MDA are defined by the OMG as well [20]. Highly superior tools are used for MDA. Further, standardized semantics for an executable subset of UML exists, called Foundational UML (fUML) defined by the OMG [19]. fUML is another approach of using UML as programming language. [2]

The main criticisms of UML regards mostly the high complexity of the language and its inconsistent metamodel [16], [8], [13]. Further, the semantics of numerous UML constructs are vague formulated and not fully specified in the standard. This leads to different ways in their usage and interpretation in models [9]. Moreover the different UML modeling tools consider different subsets of UML constructs [24]. The high complexity and unclear semantics of UML as well as the fact that UML tools are highly diverse regarding their support of UML leads to diverse ways in which UML is used in practice. UML is used for a wide variety of applications (e.g., for modeling software systems, industrial plants, business processes) and depending on the purpose for which UML is used, different subsets of UML are used.

In summary UML has a wide variety of applications and is differently used depending on the purpose of its usage. How UML is used in practice is a research question which is not well elaborated yet (cf. Chapter 5). However, practical understanding of the usage of UML can be valuable for several purposes. It can make the development of UML tools easier. Further, an understanding of how UML is used in practice can support further investigations of the weaknesses and potentials of the language. In addition knowledge about which parts of the language are commonly used and which not can lead to the development of a simplified version of UML by reducing its current complexity by focusing on the most used concepts of the language. Such a version of UML could make life easier for users of UML and for UML tool vendors.

1.2 Problem Statement

UML is a language which is used for numerous purposes. Depending on the purpose different parts of the UML are used. Currently no meaningful studies about the usage of UML in real world models exist (cf. Chapter 5). An understanding of how UML is used in practice can help in reducing the weaknesses of the language and minimizing the criticisms. This work tries to find out how UML, in particular UML 2.x, is used. In the following, potential findings, which can be deducted from such a research, are listed:

- **UML tools:** A better understanding of the practical usage of UML can help in developing UML tools. It can provide insight into which parts of UML are important and hence have

[2] http://de.slideshare.net/seidewitz/programming-in-uml-an-introduction-to-fuml-and-alf. Accessed: 2013-02-06

to be supported by the tool and which parts can be neglected. This leads automatically to a significant cost and time reducing in implementing such tools. Moreover the tools might be easier to use.

- **UML literature:** Knowledge about which parts of UML are used in practice would make life for authors of UML guides easier. They would know on which UML constructs they should focus their writing.

- **Education:** An improved understanding about the practical use of UML would enhance lectures about UML. The lecturers would be able to adapt their teaching on the most used UML concepts. Also a simplified version of UML for teaching and practising could be used in universities.

- **Simplified UML metamodel:** A simplified version of the UML metamodel, besides the current one, could be another result of studying which parts of UML are used in practice. This could help OMG in creating such a simplified version by removing unused or less popular concepts.

- **Modifications of the UML metamodel:** Studying which parts of UML are used in practice can also show OMG which UML constructs need further modifications, where the semantic definitions are not clear enough and need further explanations or where constructs need to be extended or even can be combined (for example notes and constraints could be combined if notes are usually used for expressing constraints).

1.3 Aim of this Work

The aim of this work is to analyze real world UML models in order to answer the following overall research question driving this thesis: Which parts of UML are used in practice? This question is broken down into the following more detailed research questions which are categorized into five groups.

Usage of UML Language Units

1. Which language units of UML are used?

2. Which language units of UML are frequently used in combination?

Usage of UML Concepts

1. Which modeling concepts of UML are used?

2. Which concepts of UML 1.4, 1.5 and 2.x are used?

Usage of UML Diagrams

1. Which diagram types of UML are used?
2. Which UML diagram types are frequently used in combination?
3. Which modeling concepts of UML are visualized in the distinct diagram types?

Relationships Between Different UML Language Units

1. How are the different parts of a model related to each other?

Stereotype Usage

1. Which UML concepts are frequently extended by stereotypes?
2. Which UML profiles are used?

1.4 Methodological Approach

The methodological approach for conducting the research presented in this thesis consists of 4 steps:

1. **Elaboration of metrics.** First, UML usage metrics were elaborated for answering the research questions of this thesis listed beforehand.
2. **Implementation and testing of the metrics.** In the second step, a program for analyzing UML models and calculating the elaborated metrics was implemented and tested.
3. **Collection of analyzable UML models.** Models were collected to be analyzed.
4. **Analysis of UML models.** The collected models were analyzed by the implemented program which calculated the elaborated metrics. Further, the calculated metrics were interpreted.

The aim of this thesis was to analyze real world models concerning their usage of UML. We chose to analyze UML models created with the UML modeling tool Enterprise Architect (EA). It is currently one of the most widely used UML modeling tools [3]. Furthermore a close contact to Sparx Systems[4] exists through the Business Informatics Group of the Faculty of Informatics of the Vienna University of Technology[5] which advised this thesis.

The EA models were collected via Google by searching for accessible Enterprise Architect Project (EAP) files, which could be analyzed.

[3] http://model-based-systems-engineering.com/2013/01/07/most-popular-uml-modeling-tools. Accessed: 2013-02-05.

[4] Sparx Systems. http://www.sparxsystems.com. Accessed: 2013-28-01

[5] BIG. http://www.big.tuwien.ac.at. Accessed: 2013-28-01

The results of the analysis of the collected UML models were evaluated using Microsoft Excel. Visual Basic for Applications (VBA) in Excel was used for aggregating the data collected in the analysis of the models. For calculating data mining metrics the tool WEKA[6] was used. Also the software development environment Eclipse[7] with the programming language JAVA was used for collecting additional data about the analyzed models.

1.5 Structure of this Work

In Chapter 2 the methodology used for carrying out the study about the practical usage of UML is explained in detail.

Chapter 3 describes how the models were analyzed for answering the research questions of this work. In the first part the tool Enterprise Architect and how it stores model data is explained. In the second part the challenges in obtaining model data from EA are discussed. In the third and last part an example model is illustratively analyzed step by step to explain how the models were analyzed in order to answer the research questions.

The results of the empirical study are presented and interpreted in Chapter 4. Each sub chapter deals with one category of research questions as presented in Chapter 1.3. Also limitations and possible future work for each evaluated research question are discussed.

In Chapter 5 related work is discussed and compared to the carried out study.

Chapter 6 sums up all findings and interpretations from Chapter 4 and also gives an overview of possible future work.

[6] WEKA - Data Mining Software. http://www.cs.waikato.ac.nz/ml/weka/index.html. Accessed: 2013-01-02

[7] Eclipse. http://www.eclipse.org. Accessed: 2013-20-02.

CHAPTER 2

Research Methodology

In this Chapter the steps carried out for answering the research questions of this thesis are described. All methods and technologies which were used in each step are explained in detail. The methodological approach consists of 4 steps. Each of the next 4 sections describes one step.

2.1 Elaboration of Metrics

Firstly, an in-depth research about the language concepts of UML was made. For that current UML literature [15, 25, 26] and the UML specification of OMG [22] were observed. Subsequently the research questions of this study were elaborated. As a last step metrics were developed for answering the research questions. For an improved understanding how metrics should look like and which metrics might be accurate, a research on existing object oriented metrics and UML metrics was made in advance (see Chapter 5).

The first draft of metrics only consisted of counting how often which UML concept is used in a UML model. For example the number of interfaces, abstract classes, states, etc. used in the analyzed UML models. In a second iteration aggregated metrics were developed based on the simple metrics, suitable statistical measurements for data sets were chosen [2,12], and data mining metrics were implemented [14]. In a further step these metrics were categorized according to the research questions they try to answer.

2.2 Implementation and Testing of Metrics

In this phase it was investigated, how the data necessary to answer the research questions can be obtained from EA models. The built in scripting environment of EA[1] was chosen for accessing and analyzing EA project files. The EA scripting technology is an easy way of accessing all

[1] EA Scripter Window. http://www.sparxsystems.com/uml_tool_guide/modeling_tool_features/the_scripter_window.htm. Accessed: 2013-29-01

model data. Furthermore, with the Model Driven Generation (MDG) Technology[2] such scripts can be transferred between different EA environments. EA supports several script languages. The script language Microsoft JScript was chosen for this study. Further details about how models are stored in EA and how they can be accessed can be found in Chapter 3.

After a review of all metrics the script for analyzing EA models was implemented. The testing was done during the implementation. For this own test models were created.

The output of the script was persisted in XML. The big advantage of this file type was that the results of the script could be displayed in Microsoft Excel in any possible way[3]. Therefore the output was formatted in a way, that it suited best for further analyses in Excel. The implementation of the script for analyzing EA models is presented in Chapter 3.2 in more detail.

2.3 Collection of UML Models

To get a good overview how UML is used in practice many UML models from the practice are needed. The initial plan was to spread over the script to companies. Therefore, it was carefully paid attention that the script reads out the model data as anonymously as possible. A 1:1 recreation of the analyzed models had to be impossible. The script was exported into a EA readable MDG format. Hence the possibility was given to import the script easily to other EA projects. A website was established[4], where companies could easily download the script and participate on a survey. The survey was created to gain additional meta data about the analyzed models and the companies. With the help of Sparx Systems it was tried to attract some of their customers to participate in the survey and to download the script, run it on their models, and send the output back for further analysis. The response from the established website, where the script could be downloaded, was zero. Neither one model was analyzed by the script nor a survey was filled in by someone. The reasons for such a low response are not clear. One might be that the companies did not see any benefit and usage of this research and therefore did no participate. Another reason could be that the companies were still too afraid of providing too much knowledge through the analysis of their UML models. Also the way how the script was distributed could be one of the causes why it failed.

Therefore, another source for EA models had to be found. We decided to use Google for getting analyzable EA models. With Google it is possible to search for special types of files. The script for analyzing UML models was only implemented for Enterprise Architect Project files, hence Google was scanned for this file type. The exact search query was “filetype:eap“. The search request delivered 320 results. Every result was examined for EAP files which are not corrupt

[2] MDG Technology.
http://www.sparxsystems.com/enterprise_architect_user_guide/9.2/standard_uml_models/mdgtechnologies.html. Accessed: 2013-28-03

[3] Excel-XML.
http://office.microsoft.com/en-us/excel-help/overview-of-xml-in-excel-HA010206396.aspx?CTT=1. Accessed: 2013-25-03

[4] Website of this thesis. http://www.modelexecution.org/?page_id=116. Accessed: 2013-20-03

and analyzable by the script. Overall 92 such EAP files were found, mostly from some open source software projects where EA was used for documenting their systems in a model-based way. Most of the models are from real world projects (for detailed information about the models see Chapter 4.1). Table A.14 in the Appendix A provides a list of the website urls where the analyzed EA models were found.

2.4 Analysis of UML Models

In the final step all the found models were analyzed by the implemented script in EA. Excel and Excel VBA [27] were used for further analyzing the data obtained by the script. As already mentioned the XML output of the script was formatted in a way Excel could easily open them. Additional with the tool WEKA data mining metrics were calculated.

While the results were investigated new interesting research questions arose, which could not be answered with the data obtained by the implemented script. Therefore, JAVA and the Automation Interface (AI) of EA[5] were used to obtain further data about the analyzed models, for answering the additional research questions. EA provides own JAVA libraries and uses ActiveX Com clients[6] for connecting easily to development environments. The additional metrics were implemented using JAVA with the development environment Eclipse.

After all data necessary for answering the research questions of this thesis were obtained, the data was investigated and interpreted.

[5] Automation Interface. http://www.sparxsystems.com/uml_tool_guide/sdk_for_enterprise_architect/setup.htm. Accessed: 2013-29-03

[6] ActiveX Clients. http://msdn.microsoft.com/en-us/library/windows/desktop/ms221336%28v=vs.85%29.aspx. Accessed: 2013-29-03

CHAPTER 3

Technical Implementation

In this Chapter it is described how the EA models have been analyzed by implementing a Microsoft JScript program. First the model handling of the EA tool is described, second it is explained how the script analyzes EA models, third the output generated by the script is explained using an example.

3.1 Input: Enterprise Architect Models

Enterprise Architect - Object Model

Every EA model is stored as an Enterprise Architect Project (EAP) file with the extension *.eap. For each EAP file an own database is created for storing the information of the contained model. EA supports several database management systems (DBMS). The supported DBMS are: Microsoft Access (all versions), SQL Server, MySQL, Oracle 9i and 10g, PostgreSQL, MSDE and Adaptive Server Anywhere[1].

The EA object model[2] defines how a model is stored in the database. For accessing an EA model the Windows OLE Automation[3] has to be used. This technology makes it possible to access objects from one application in another application.

A common way to access the model currently opend in an EA instance is through the Scripter Window in EA. It enables with simple script languages to read, write and manipulate data of the EA model. The following script engines are supported by the Scripter Window[4]:

[1] EA supported DBMS. http://www.sparxsystems.com/support/faq/database.html. Accessed: 2013-11-03.

[2] EA object model. http://www.sparxsystems.com/uml_tool_guide/sdk_for_enterprise_architect/theautomationinterface.htm. Accessed: 2013-17-03.

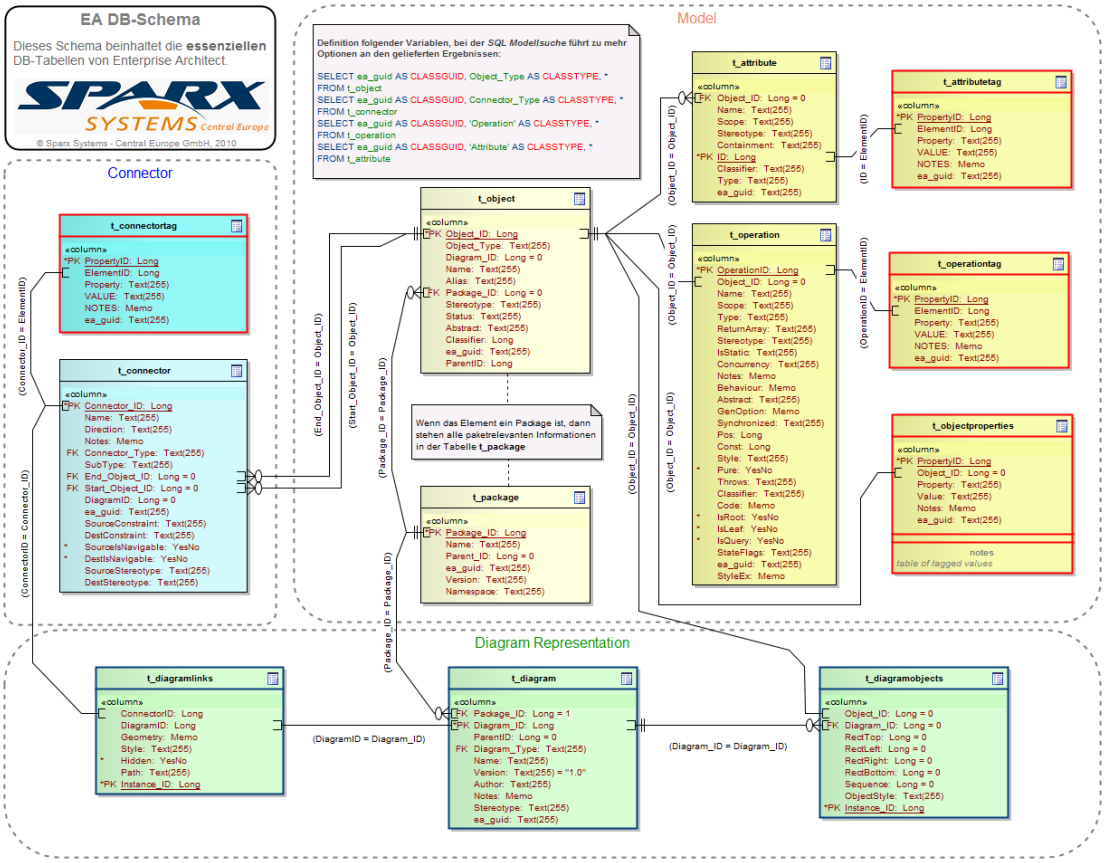
[3] Microsoft OLE Automation. http://msdn.microsoft.com/en-us/library/dt80be78.aspx. Accessed: 2013-29-01.

[4] EA Scripter Window. http://www.sparxsystems.com/uml_tool_guide/modeling_tool_features/the_scripter_window.htm. Accessed: 2013-29-01

- JavaScript
- Microsoft JScript
- Microsoft VBScript

The Automation Interface (AI) of EA provides an API for accessing the model. The most important interface of this API is the `Repository`. The `Repository` is the root object of a model and makes it possible to access all elements, connectors, diagrams and packages from the model. In EA an element is a modeling unit of a model and connectors are the relationships between the elements. For a clear understanding how EA stores a model, Figure 3.1 shows an excerpt of the database schema of EA[5].

Figure 3.1: EA DB-Schema



In the database an element is saved in the `t_object` table. An element can be an UML element like a class, a component or a state. Connectors are saved in the `t_connector` table.

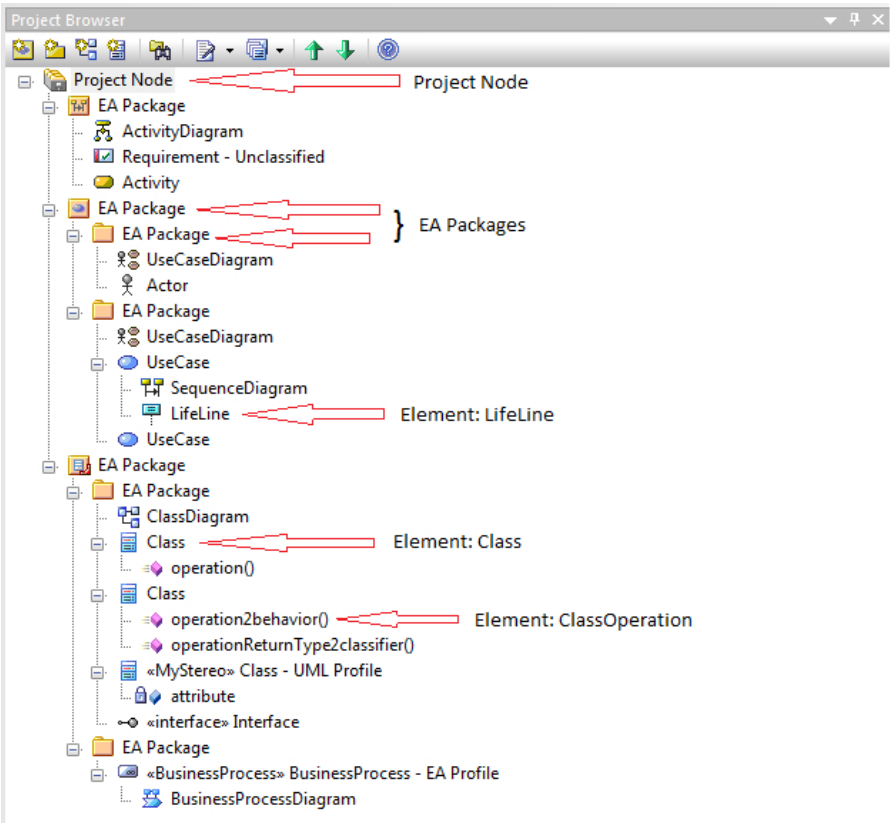
[5]EA DB-Schema. http://blog.sparxsystems.de/2010/10/enterprise-architect-db-schema/. Accessed: 2013-29-01

A connector is for an example an association, a note link or a control flow. Attributes and operations are related to an element. Typical attributes and operations are class attributes or class operations but also a state behavior (entry, do, exit) is handled as an operation in EA. Attributes and operations of an EA project are saved in the tables `t_attribute` and `t_operation`. In diagrams elements, connectors, attributes and operations can be visualized. Diagrams are saved in the table `t_diagram`. In this work the terms *element*, *package*, *diagram*, and *connector* are used as defined in EA, i.e., the contents of the tables `t_object`, `t_attribute`, and `t_operation` for *element*, the contents of the table `t_package` for *package*, the contents of the table `t_connector` for *connector* and the contents of the table `t_diagram` for *diagram*. Further, *model element* will be used as a general term for all these objects from now on.

Enterprise Architect - Project View

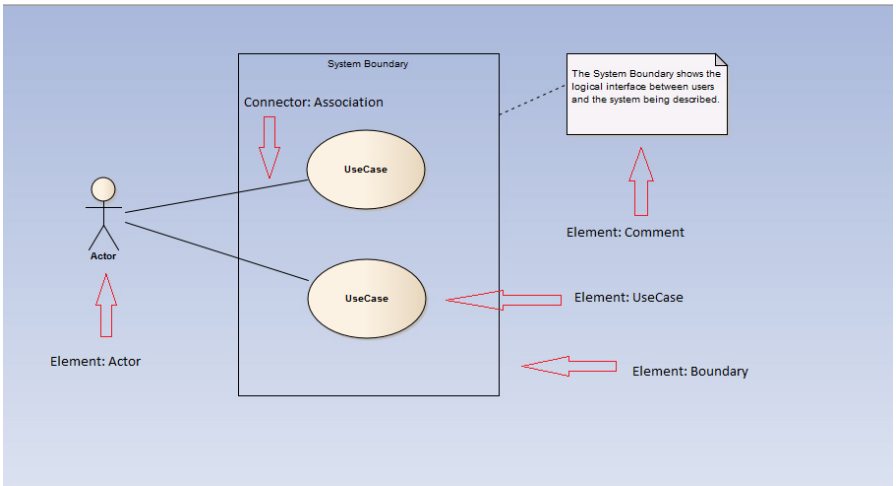
Figure 3.2 shows how a model is displayed in EA. Connectors are not shown in this view. They are only visible in the diagram view. Model elements are always contained in packages. Packages are used as a structuring method for models in the EA modeling tool. It is obligatory to create a package to be able to create model elements.

Figure 3.2: EA Project View



It is important to clarify that a project node (as shown in Figure 3.2) does not represent a model in this evaluation. The whole EA project file was considered as one model, regardless how many projects it had. From the analyzed 92 EA projects only 2 had more than one project node. In these 2 EA projects the content of the project nodes was seen as content of one single model.

Figure 3.3: EA Diagram View of an Use Case Diagram



Besides connectors also other model elements are not shown in the EA project browser. Notes and use case boundaries are examples. These elements can only be seen in the diagram where they are visualized (see Figure 3.3 for a diagram view in EA). How EA exactly decides which elements are displayed in the project browser and which not is unclear.

3.2 Implementation: Script for Analyzing Enterprise Architect Models

For this research the Scripter Window with the script language Microsoft JScript was used to access the model data and to obtain some simple statistics about the EA models. The script uses mainly the AI API for accessing the model data. In cases the provided AI calls were insufficient for obtaining the required model data, SQL queries were used to access the model data directly from the database. The script uses no real object orientation programming style. All queries and calculations are implemented in functions and data is stored in multiple arrays only at runtime.

The script consist of over 18,000 lines of code. Therefore, it needs considerable time for analyzing a model. For a model with 3,300 elements, 2,000 connectors, 400 diagrams and 300 packages the script needed over 25 minutes. As the evaluated models never exceeded the amount of 4,000 model elements, this was never the case in this research. Anyway it also depends on the used hardware and the nesting of elements within an EA project how long the script needs for analyzing the model.

The main task of the script is to identify the UML language concepts in the analyzed models and to read out all the information about the model elements, which were needed for calculating the defined metrics for answering the research questions. One of the big challenges in implementing the script was to determine the type of a model element, i.e., to identify if a model element is for instance a UML class, state, or activity. In EA all elements, diagrams, and connectors are saved as objects in the corresponding tables in the model database (see Chapter 3.1). There exists no explicit declaration of the type of a model element in the model database. Therefore several AI requests were needed for each model element to determine if it is an instance of a UML modeling concept and in a further step which concept exactly.

The determination if a specific model element is of the UML type `Interface` looks like depicted in Listing 3.1.

Listing 3.1: Microsoft JScript Code Example 1 - Determination if an Element is a UML Interface

```
1 if(currentElement.Type == "Interface" && currentElement.MetaType == "
      Interface" && currentElement.StereotypeEx == "interface" && (
      currentElement.Subtype == 0 || currentElement.Subtype == 8))
2   {
3     conceptType = "Interface:UML";
4   }
```

Usually that was the way to determine the type of each model element. The type of a model element in EA is defined by the values of four properties: `Type`, `MetaType`, `Stereotype`, and `Subtype`. Only if all values match certain constants, the model element can be categorized precisely as specific UML concept. Just one difference can lead to a complete other model element type or even a non UML element type.

However, in some cases those requests were not enough to determine the type of a model element. For example the query for a state element with regions looks like depicted in Listing 3.2.

Listing 3.2: Microsoft JScript Code Example 2 - Determination if an Element is a UML State containing Regions

```
1  if(currentElement.Type == "State" && currentElement.MetaType == "State" && (
       currentElement.Subtype == 8 || currentElement.Subtype == 0) &&
       currentElement.Stereotype == "")
2    {
3      conceptType = "StateElement:UML";
4
5      //more complex request to check if state has regions
6      //in this case the result hast just one column and one row!
7      var SQLresult = DBGetFieldValueArrayString("Description", "t_xref",
           undefined, "Client = '"+currentElement.ElementGUID+"' AND Type = '
           element property' AND Name = 'Partitions'");
8
9        if(SQLresult != "" && SQLresult[0][0].indexOf("@PAR;Name=") > -1)
10       {
11         conceptType += ";State with Regions:UML";
12       }
13   }
```

First the element type is read out (Line 1) and then a SQL query determines (Line 7) if the state has regions or not. The function `DBGetFieldVaueArrayString` processes the SQL requests. Parts of the SQL syntax are passed within the parameters of the function. In this way all needed SQL requests were made.

In programming steps the script has several phases for obtaining the needed model data and calculating the output. In the following just a very abstract description of these steps is given which should only give a raw overview how the structure of the script looks like.

1. In a first step all elements, connectors, diagrams, and packages are read out recursively starting from the root package. Elements, diagrams, and connectors are saved into multiple arrays according to the package they reside in.

2. The script goes iteratively through all the previously created arrays. All the necessary metrics about elements, connectors and diagrams are calculated from these data sets. The results are saved then in other multiple arrays, which can be easily read out for the final output.

3. All the generated arrays from the second step are read out and an XML file is created with the calculated values.

3.3 Output: Obtained Model Data

The script generates an XML file which can be opened with Microsoft Excel or Apache OpenOffice Calc. In this work Excel was used to analyse the obtained model data. If the XML file is

opened with Excel, the output generated by the script consists of 3 worksheets. Each sheet contains aggregated and calculated data about the analyzed model. The worksheets are called "Model - Content", "Diagram Metrics" and "Correlations and Relationships". In the "Model - Content" worksheet all UML concepts used in the model are listed with the number of their occurrence. The "Diagram Metrics" worksheet deals with data about the used diagrams and the diagram contents. All used relationships between elements in the model are captured in the worksheet "Correlations and Relationships". In the following each worksheet will be described in detail on the basis of a sample model (which is depicted in Figure 3.4).

Worksheet: Model - Content

All considered UML concepts, which were found in the analyzed EA models, are listed in this worksheet with the total amount of their occurrence. Connectors like associations or generalizations are not considered in this worksheet. Connectors are considered in the worksheet "Correlations and Relationships", which is explained in one of the next sub sections. The content of the worksheet "model - content" is explained with the help of the example model depicted in Figure 3.4. Table 3.1 and Table 3.2 contain parts of the output generated by the script for this example model. For an easier comparison the model elements of the model are named according to their type. As we see in Table 3.1 this worksheet contains the columns `uml language unit`, `concept`, `language`, `total` and `in diagrams`. The values in the `language unit` column give information about to which UML language unit the model element belongs to. The `concept` column holds the type of a model element. All the UML concepts, which the script considers are listed in the Appendix A.1. Another important information is contained by the column `language`. If the type of a model element is a UML modeling concept and it is not extended by a stereotype the value of the `language` column is "UML". But if the model element is extended by stereotypes the `language` value is "UML Profile". The value "EA Profile" means, that the model element is an element of one of the other languages supported by EA such as BPMN or SysML. The value "unclassified" means that the type of the model element can not be declared as "UML", "UML Profile" or "EA Profile". Every additional modeling language supported by EA is implemented as UML Profile. Therefore it cannot be easily distinguished between UML Profiles and other modeling languages. Further explanations about the used categorization method of the model elements into languages can be found in Chapter 4.1. If a model element is declared as EA Profile or unclassified the value in the column `concept` gives additional information about the type of the model element. For model elements declared as "EA Profile" the used EA stereotype and EA profile are showed. For "unclassified" model elements the `MetaType` as saved in the EA model database for this object is provided. The `total` column provides the information how many model elements of a specific type occurred in the analyzed model. The `in diagrams` column provides information about the visualization rate of the model element. The visualization rate says how many of the model elements from this type are visualized in diagrams.

As can be seen in Table 3.1 (5 and 8) the sample model contains 2 classes (cf. Figure 3.4/1) without stereotypes and 1 class extended by a stereotype (cf. Figure 3.4/1.1). Further, 1 class attribute (6; cf. Figure 3.4/4) and 3 class operations (9; cf. Figure 3.4/3) are contained in the

model. 1 of the class operations had as return type a classifier (11), 2 a primitive type (10). 1 activity was modeled (2; cf. Figure 3.4) and 7 packages were contained (1; in Figure 3.4/EA Package). They are named “EA Package“ in the output as they are slightly different used in EA than in UML (see Chapter 3.1 above). The visualization rate of packages is not measured. Furthermore, we can see a business process element from the language BPMN2.0 (13) and an unclassified element of the type “Requirement“ (3). The business process is declared as “EA Profile“ and the requirement as “unclassified“. More specific data about these two non UML concepts can be found in the `concept` column. In addition we see elements from the language unit use case (15, 16, 17), 1 life line (18; cf. Figure 3.4/2), 3 text elements (14) and 1 note (4). A closer look to Figure 3.4 indicates that some listed concepts (comment, boundary cf. Figure 3.5) from Table 3.1 are missing in the project browser view of EA. The reason is that the EA project browser does not visualize all modeled elements, as mentioned in Chapter 3.1.

Figure 3.4: EA Project Browser View of the Sample Model

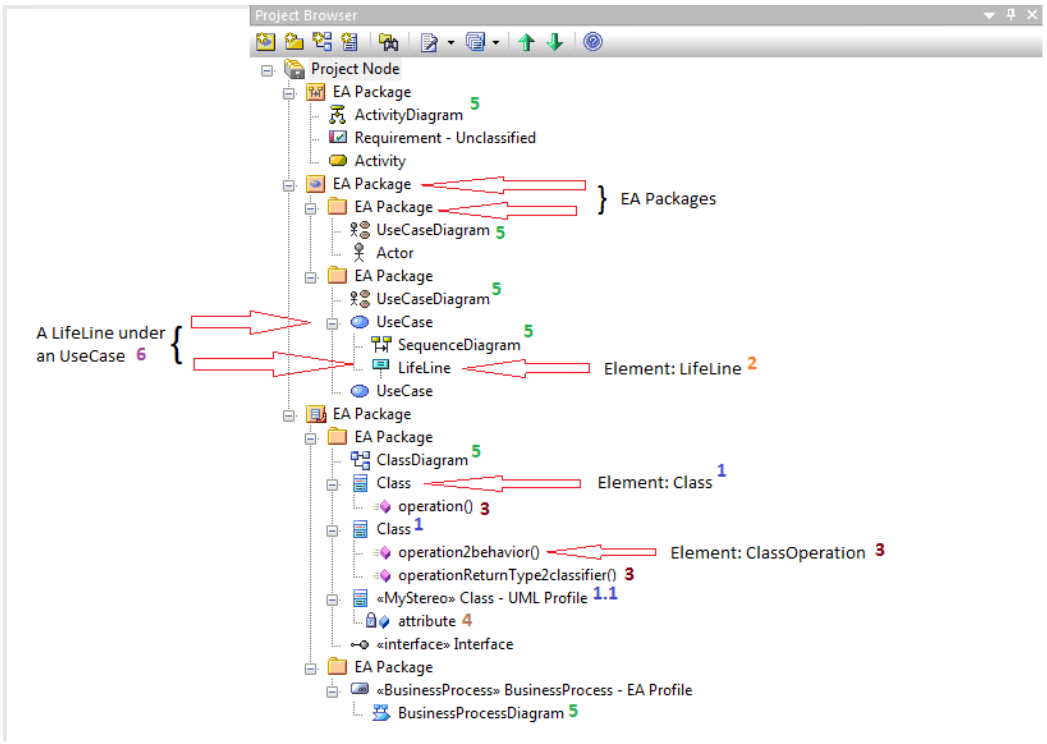


Figure 3.5: EA Diagram View of an Use Case Diagram of the Model in Figure 3.4

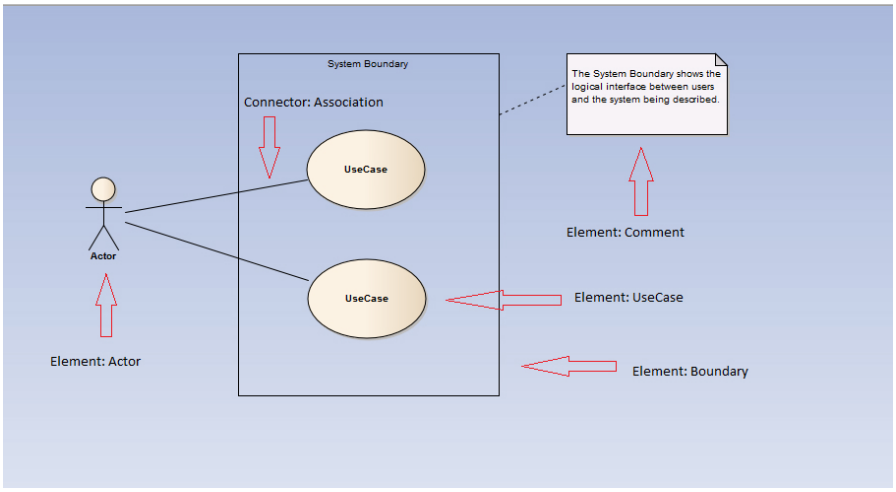


Table 3.1: Excel Worksheet: Model - Content (1/4). An Illustrative Example of the Output.

	Uml Language Unit	Concept	Language	Total	In Diagrams
1	EA Package	Package	UML	7	
2	Activity	Activity	UML	1	1
3	Unclassified	Requirement	Unclassified	1	0
4	Auxiliary Construct	Note	UML	1	1
5	Class	Class	UML Profile	1	1
6	Class	Class Attribute	UML	1	1
7	Class	Class Attribute - Primitive Type	UML	1	1
8	Class	Class	UML	2	2
9	Class	Class Operation	UML	3	3
10	Class	Class Operation - Primitive Type	UML	2	2
11	Class	Class Operation - Classifier Type	UML	1	1
12	Class	Interface	UML	1	1
13	Unclassified	BusinessProcess, Stereotype=BusinessProcess, Profile=BPMN2.0	EA Profile	1	0
14	Unclassified	Text Element	EA Note	3	3
15	Use Case	Actor	UML	1	1
16	Use Case	Boundary	UML	1	1
17	Use Case	Use Case	UML	2	2
18	Interaction	Life Line	UML	1	1

The worksheet consist of further 128 columns. These columns contain information about the occurrence of a special type of relationship between model elements. In EA it is possible to create a model element under another model element in the project browser, i.e., an arbitrary model element can be contained by another arbitrary model element (cf. Figure 3.4/6). In this work this construct was seen as a relationship between model elements. For example if life lines are modeled under a use case in the project browser we can assume that the life lines are somehow related to this use case. Due to the huge amount of possible containment relationships between model elements the script only considers if model elements are contained by model elements of the following types: class, activity, action, state machine, state, use case, interaction and component. An example is shown in Table 3.2, where we can see the distribution of life lines in use cases.

Table 3.2: Worksheet: Model - Content (2/4). An Illustrative Example of the Output.

Concept	In Use Cases	Max	Min	Avg	Stdv	Med	Range	Mode
Life Line	1	1	0	0,5	0,5	0,5	1	0,1

In the example model one life line is contained by a use case. (see Table 3.2, compare with Figure 3.4/6). The maximum (max) amount of life lines contained by use cases is 1, minimum (min) is 0, arithmetic mean (avg) is 0,5 with a standard deviation (stdv) of 0,5. The median (med) is 0,5, range is 1 and mode values are 0 and 1.

In the sample model a life line is contained by an use case, which would be counted in the evaluation part as relationship between a life line and an use case and therefore as a relation between the UML language units interaction and use case.

Worksheet: Diagram Metrics - Model Diagram Metrics

The "Diagram Metrics" worksheet consist of the parts, "Model Diagram Metrics" and "Element Occurrences".

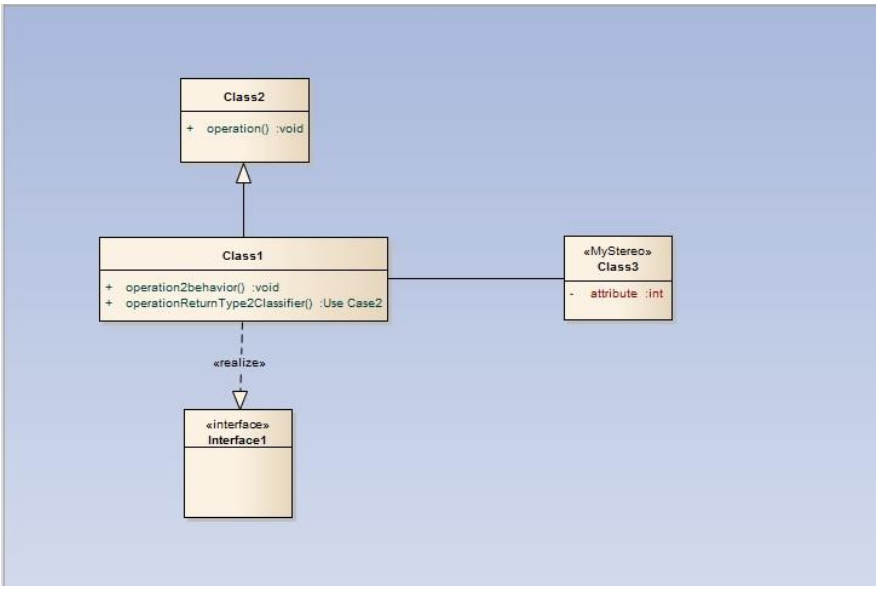
In Table 3.3 we see the "Model Diagram Metrics" part. The columns `diagram type` and `total` provide the information how many diagrams of a specific UML diagram type are used in the analyzed model. All the diagram types, which the script considers are listed in the Appendix A.2. If a diagram is found in the model, which can not be classified, then the `diagram type` column contains detailed EA specific information about this diagram. The next columns provides information about the content of the diagrams. Not all modeled elements in a diagram are considered. Connectors are skipped completely. The considered elements are some of the main UML language concepts, namely class, activiy, action, interaction, lifeline, state, state machine, use case, actors, component, and object. Thereby sufficient data can be obtained to determine in which diagrams which main UML language concepts and consequently which UML language units are visualized. Further, there is a distinction made between UML main concepts with stereotypes and UML main concepts without stereotypes.

Table 3.3: Worksheet: Diagram Metrics - Diagram Content. An Illustrative Example of the Output.

Diagram Type	Total	Total Classes	Max	Min	Avg	Med	Mod	Stdev	Range
ActivityDiagram	1	0	0	0	0	0	0	0	0
UseCaseDiagram	2	0	0	0	0	0	0	0	0
SequenceDiagram	1	0	0	0	0	0	0	0	0
ClassDiagram	1	2	2	2	2	2	2	0	0
Analysis:BPMN2.0 ::Business Process	1	0	0	0	0	0	0	0	0

In Table 3.3 the data obtained for the sample model in the Model Diagram Metrics part of the Diagram Metrics worksheet is shown. We see that 4 UML diagram types and 1 BPMN 2.0 diagram (compare with Figure 3.4/5) are contained in the sample model. Additionally we see

Figure 3.6: EA Diagram View of an Class Diagram of the Model Depicted in Figure 3.4



the occurrence of classes in these diagrams. The same data is obtained for the other considered UML concepts but skipped in Table 3.3. By comparing Table 3.3 with Figure 3.6 someone might claim that there should be 3 classes visualized in class diagrams but only 2 classes are listed in the worksheet output. The reasons is that the third class is extended by a stereotype and listed under different columns ("classes with profile") in the worksheet. The interface element and all connections from Figure 3.6 are not considered in the analysis of the diagram content as they are not one of the considered UML concepts.

Worksheet: Diagram Metrics - Element Occurrences

In Table 3.4 the element occurrences in diagrams can be seen. In EA an element can be visualized in several diagrams. For example if an class is created in the project browser it is possible to add this specific element to every diagram in the model. In this part of the worksheet such elements, which appear in several diagrams in the model, are listed. Only elements which appear more than once in diagrams are considered.

Table 3.4: Worksheet: Diagram Metrics - Element Occurrences in Diagrams. An Illustrative Example of the Output.

Uml Language Unit	Element Type	Exist in Diagrams(>1)
UseCase	ActorElement#1	3

As we can see in Table 3.4 an actor in the sample model was visualized in 3 different diagrams.

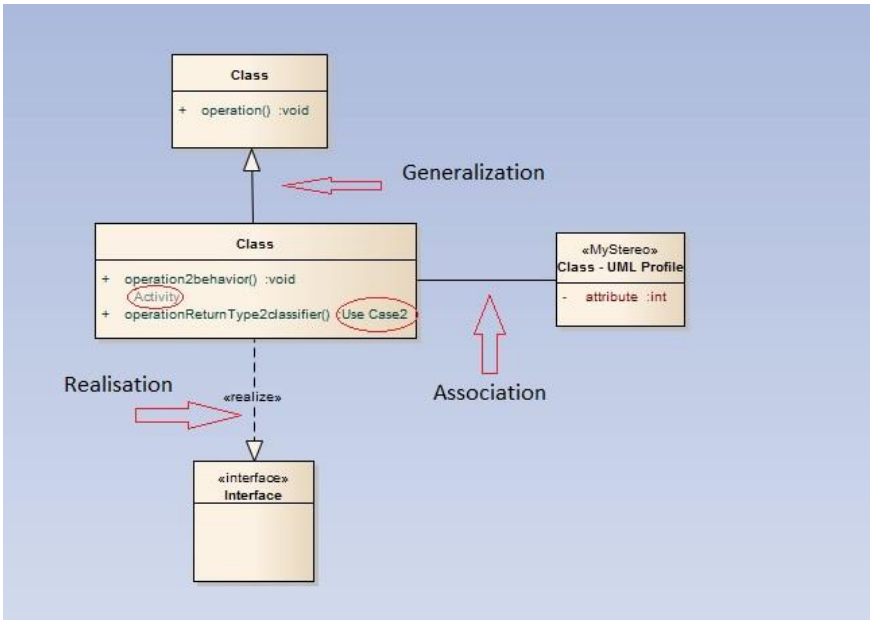
Worksheet: Correlations and Relationships

The last worksheet in the XML output file holds data about all relationships between the elements in the EA model. In Table 3.5 we can see the relationships obtained by the script for the excerpt of the sample model depicted in Figure 3.7. The target and source element, the relationships type and the total occurrences of the relationships are calculated by the script.

Table 3.5: Worksheet: Correlation and Relationships. An Illustrative Example of the Output.

Source Element	Target Element	Relation	Total
Class	Class	Association	1
Class	Interface	Realisation	1
Class	Class	Generalization	1
ClassOperation	Activity	ClassOperation2Behavior	1
ClassOperation	Use Case	ClassOperationReturnType2Classifier	1

Figure 3.7: EA Diagram View - Relations in an Class Diagram of the Model Depicted in Figure 3.4



In the sample model 1 association relationshp between classes, 1 realization relationship between classes and interfaces and 1 generalization relationship between classes are modeled. Moreover the behavior of 1 class operation is defined by an activity (see “ClassOperation2Behavior“ in Table 3.5) and another class operation has an use case as return type (see “ClassOperationReturnType2Classifier“ in Table 3.5). Table A.12 and Table A.13 in the appendix list all UML connectors and relationships the script considers.

CHAPTER 4

Results

In this Chapter the results of the empirical research of the UML Models will be presented and evaluated. After listing some general data about the observed models, each research question of this work will be answered based on the evaluated results about the UML usage in the models.

4.1 Basic Data of the Observed Models

As already mentioned in EA not just UML can be modeled also other modeling languages like BPMN, ICONIX or UML Profiles like SysML are available. The first important figures about the models are how much UML is used in the models, evolution of the models, what was the designated use of the models in which domains and how big are the models.

What is the Model Composition?

EA is a modeling tool supporting several model languages beside UML. Therefore it is important to distinguish between elements from UML and from other languages in the analysis of the models. The EA elements and connectors were categorized into 4 language types: `UML`, `UML Profile`, `EA Profile` and `Unclassified`. These categorization was chosen because it can show best how much UML was used in the 92 models on basis of the EA modeling schema. Further, it differentiates between extended and not extended UML elements.

All elements under the category `UML` are pure UML elements. No stereotypes were used for these elements and they can be found in the list of considered concepts in the appendix A.

Elements under the category `UML Profile` are only elements which were recognized as UML elements by the script and were extended by one or more stereotypes. But only stereotypes which had not a qualified name in their description in the EA database were considered, since this indicates an own modeling language.

`EA Profile` elements are EA specific modeling elements representing notations from all modeling languages EA supports. EA uses the concept of stereotypes with an annotation of a qualified name to distinct between the different modeling languages. Some of these elements have the same basic types as UML elements, like classes, actions and so on. For an exact determination it was therefore important to read out every stereotype and look up for a qualified name. The qualified names declare to which modeling language the elements belong to. Further if a UML element with a stereotype had an qualified name it was not seen anymore as an element from the UML.

`Unclassified` elements are all elements which could not be declared to `UML` or `EA Profile` elements. Where for `EA Profile` elements we can at least figure out to which modeling language the elements belong to, this is impossible for `Unclassified` elements.

In Table 4.1 figures about the compositions of the models are listed. The numbers in Table 4.1 differentiate between `UML`, `UML Profile`, `EA Profile` and `Unclassified Elements`. The *"Total Elements"* figure in Table 4.1 considers elements, attributes, operations and element parameters but no connectors, diagrams and packages from the 92 models. The same counts for *"Total Connectors"*, *"Total Diagrams"* and *"Total EA Packages"*, where all connectors, diagrams and packages of the models were taken separately into account. *"Total Model Elements"* represents the number of all found modeled elements in the 92 sample models. *"Total not visualized Elements"* tells us how many elements from the *"Total Elements"* could not be found in any diagrams in the models. So for these elements no visualization exists. The Figures 4.1, 4.2, 4.3 and 4.4 visualize the data in Table 4.1. All UML elements, connectors and diagrams which were taken into account for the following data are listed in the appendix A.

Table 4.1: Models Composition

Total UML Elements	18 558	
Total UML Profile Elements	7 157	
Total EA Profile Elements	259	
Total Unclassified Elements	804	
Total Elements	26 778	26 778
Total UML Connectors	7 341	
Total UML Profile Connectors	1 009	
Total EA Profile Connectors	50	
Total Unclassified Connectors	27	
Total Connectors	8 427	8 427
Total UML Diagrams	977	
Total Non UML Diagrams	201	
Total Diagrams	1 178	1 178
Total EA Packages	1 060	1 060
Total Model Elements		37 443
Total not visualized Model Elements		2 042

Interpretation. We can see that mostly pure UML elements, UML connectors and UML diagrams were used. Therefore the models can give a quite good representation about how UML is used in practice. Figure 4.1 tells us that 96% (69% UML elements + 27% UML profile elements) of the elements are without doubt from the language UML. The rest of the elements are a mixture of UML, UML Profile and non UML Elements. In Figure 4.2 we can see that over 99% (87,1% UML connectors + 12% UML profile connectors) of the modeled Connectors are from the language UML. It seems for connectors no other modeling languages were needed. If we focus on the ratio between UML and UML Profile Connectors we see that for pure UML connectors covered most of the designers needs. The same fact can be seen on Figure 4.3. 83% of the diagrams are original UML diagrams. The rest are diagrams form another modeling language. The diagram category data also underline the data about the element and connector categories data, that through all the 92 models UML concepts were strongly used. All the elements, connectors and diagrams found in the models are mostly from the language UML. This result supports the reliability of this evaluation about UML.

Another interesting fact can be seen on Figure 4.4. About 92 % of all elements are visualized in diagrams. Just 8% of all created elements can not be found in any diagrams. We can see

Figure 4.1: Element Types

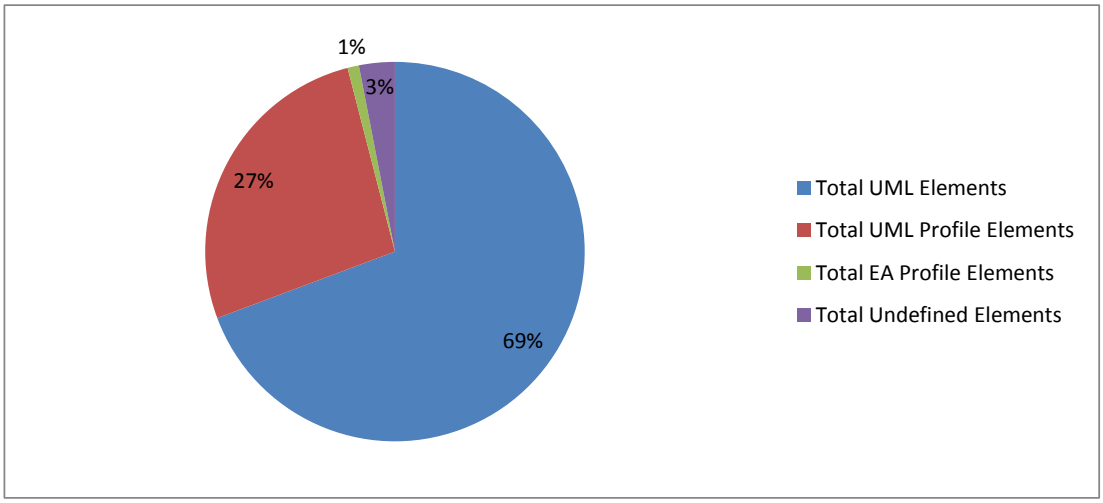


Figure 4.2: Connectors

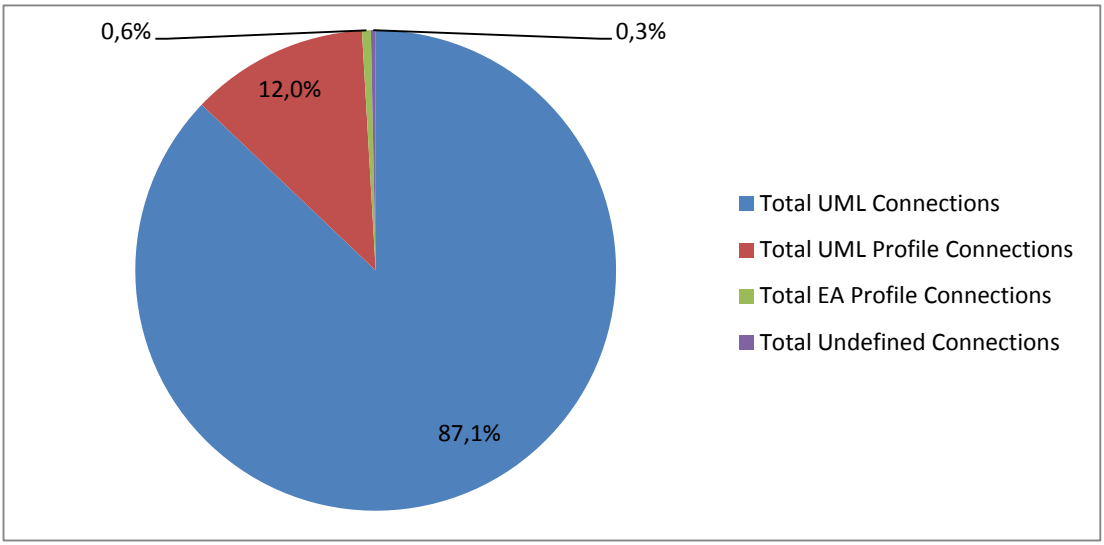


Figure 4.3: Diagrams

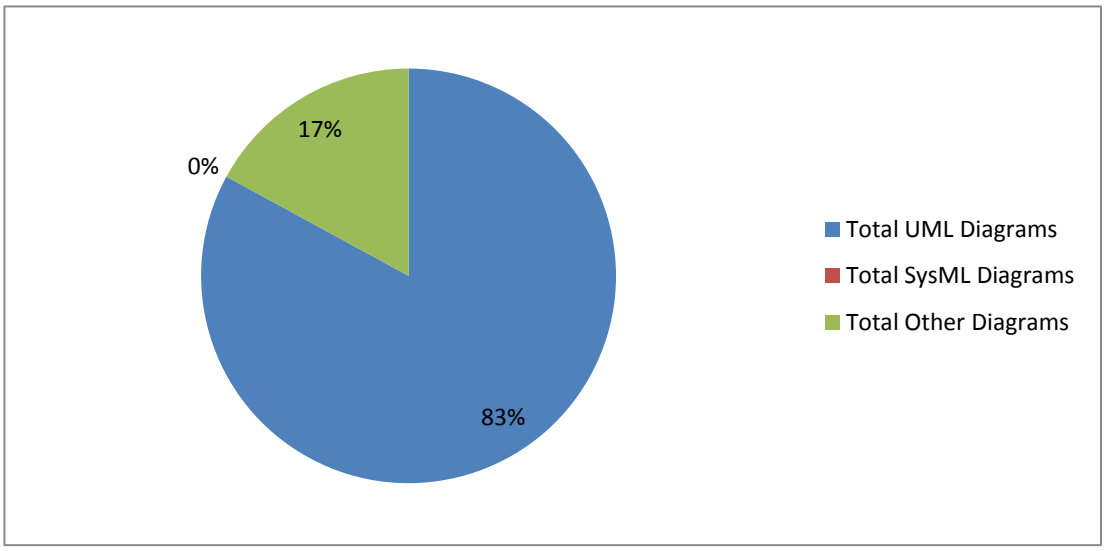
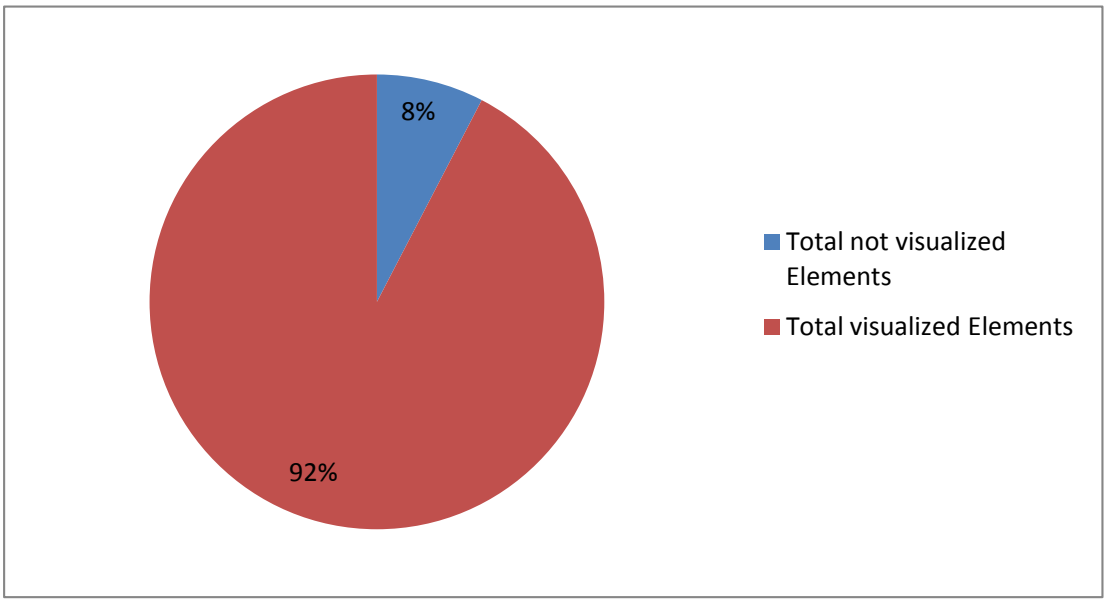


Figure 4.4: Visualization Ratio



that elements are mostly modeled in diagrams. It has also to be mentioned that in EA deleting an element from a diagram with the delete key from the keyboard does not mean that this element is also deleted from the model. You always have either right click on the element and delete it or delete it directly from the project browser. Therefore we can assume that even less elements are not visualized by purpose. This might indicate that UML models are strongly related to diagrams and that they are usually visualized with the available graphical UML notations.

Summary of the Findings

- Mostly non extended UML concepts were used in the 92 models.
- Model elements were usually visualized in diagrams.

Limitations/Future Work. In future work the script analysis methods could be improved to reduce number of unclassified model elements in the results.

Which EA Profiles and Unclassified Elements were used?

Table 4.2 and Table 4.3 give a closer look to the used EA Profile and Unclassified elements/-connectors used in the models. The values in the `EA Profile` column are the "FQName" values of the stereotypes in the EA database.

Table 4.2: EA Profile Elements in Total and in How many Models

EA Type	EA Profile	Total	In Models
Element	EAUI	192	3
Element, Connector	EAUML	37	3
Element	BPMN	16	2
Element	C#	3	2
Element	Delphi	2	2
Connector	Archimate	30	1
Element	Java	29	1
Total		309	9

Table 4.3: Model Elements declared as Unclassified in Total and in How many Models

EA Type	Unclassified Model Element	Total	In Models
Element, Connector	Unclassified	429	31
Element	Text Element	402	49
Total		831	59

Interpretation. The most used profile was *"EAUI"*, which could be found in 3 models with

a total of 192 elements (see Table 4.2). A closer look to the models with *"EAUI"* profile concepts showed that *"EAUI"* profile elements represented graphical user interfaces in the models. Most of the *"EAUI"* elements visualized buttons, text, forms or check boxes. *"EAUML"* were found in 3 models as well, *"BPMN"*, *"C#"* and *"Delphi"* in 2 models and *"Java"* and *"Archimate"* in one model. *"EAUML"* is a specific EA generated stereotype.

An text element representing text in EA models is the most used `Unclassified` element (see Table 4.3). Almost 50% of `Unclassified` elements are text elements. The other unclassified model elements were not observed in detail.

These figures underline how less non UML concepts were used through all the models and that there exist no EA Profile type or Unclassified element which could be found more regularly in the models.

Summary of the Findings

- Mostly elements had stereotypes with "FQName" attributes.
- Only 429 out of 35205 (i.e. 98,78 % of all elements and connectors) elements and connectors were marked as unclassified by the script. Therefore the script had a high rate of correct classification.
- Simple text in the EA models took the biggest part of unclassified model elements.

What is the Model's Age?

Other interesting figures about the models are their ages. In EA for every element the creation and the last modification date is saved, which can be read out from the corresponding database for each EA project file. The earliest created and latest modified element was taken from each model to calculate the model's ages measured in days. Bellow Table 4.4 shows the statistical data regarding the ages of the analyzed models. The statistics include maximum (max), minimum (min), arithmetic mean, median, mode, quartile 25% and 75% and the standard deviation of the model age distribution. Furthermore, the amount of models, which are older than a year and the amount of models, which are older than a month but younger than a year were measured.

Interpretation. On average (arithmetic mean) each model was 1,047.5 days old, almost 3 years. The youngest model was a day old and the oldest 2,955 days, which is a time span of over 8 years. The standard deviation shows that the ages of the models were highly unequally distributed. Over 50 % of the models were more than 1139 days old, but 25% were not more than 6.5 days old. So we can notice an increase of the model ages between the 25% quartile and the median. The value which appeared most often was 1, which means that most models were only a day old. When considering the median, the arithmetic mean and the additional data in the last two rows of Table 4.4, most of the models had an age measurable in years. Only 9 models had a life age which might fit to the average lengths of IT projects life spans.

To summarize the statistics mentioned above, we can deduce that the models had either a short life cycle of a few days, or very long cycle of over a year.

Table 4.4: Statistics About the Model Age in Days

Measured Values	Age in Days
Min	1
Max	2,955
Arithmetic Mean	1,047.55
Standard Deviation	966.77
Median	1,139.5
Quartile 25%	6.5
Quartile 75%	2,006.25
Mode	1

Measured Values	In Models
Models older than 1 Year	53
Models older than 1 Month, younger than 1 Year	9

Summary of the Findings

- 53 Models were older than 1 year
- 9 Models were older than 1 month, younger than 1 year
- Either models had a short life cycle of a few days or a very long life cycle over a year.

Limitations/Future Work. Additional from each model element creation and modification date can be measured and evaluated. With such analysis the evolution of models could be figured out. For example which parts are created first, which last, are more elements created at the beginning or at the end, which elements are more modified than others.

For which Domains the Models were used?

The models were also examined in what was modeled and for which domain. This was done by manual review over all model sources. Table 4.5 lists all found domains with the number of models. Figure 4.5 visualizes the data from Table 4.5. As models in the education domain all models were counted which were used in university lectures or in master thesis. In Table 4.6 these models were split into their usage with the number of models. Models found in master thesis were not counted as *"Example Model - Lecture Excercise"*, they were split to their designated use. Figure 4.6 visualizes the findings in Table 4.6.

Interpretation. The most used propose was describing a software system as Table 4.6 and Figure 4.6 show. Business processes and embedded systems were only described in 2 models.

In 3 models some kind of structure was modeled, for example in one model the parts of a standardized document for a specific domain were described. Models could be found in 9 different domains. The leading domain the models were build for was the IT sector (see Table 4.5 and Figure 4.5). As second ranked was the industrial sector, followed by the education sector. In the education sector mostly small example or exercise models were found, except some few models for master thesis.

Table 4.5: Model Domains

Domain	Models
IT	45
Industrial sector	17
Education (incl. Master thesis)	13
Medical care	5
E-Commerce	4
Hospitality	2
Housing	2
Ecology	2
Public sector	2

Table 4.6: Models - Designated Use

Designated Use	Models
Software System	76
Example Model - Lecture Excercise	9
Structure	3
Embedded System	2
Business Process	2

Figure 4.5: Model Domains

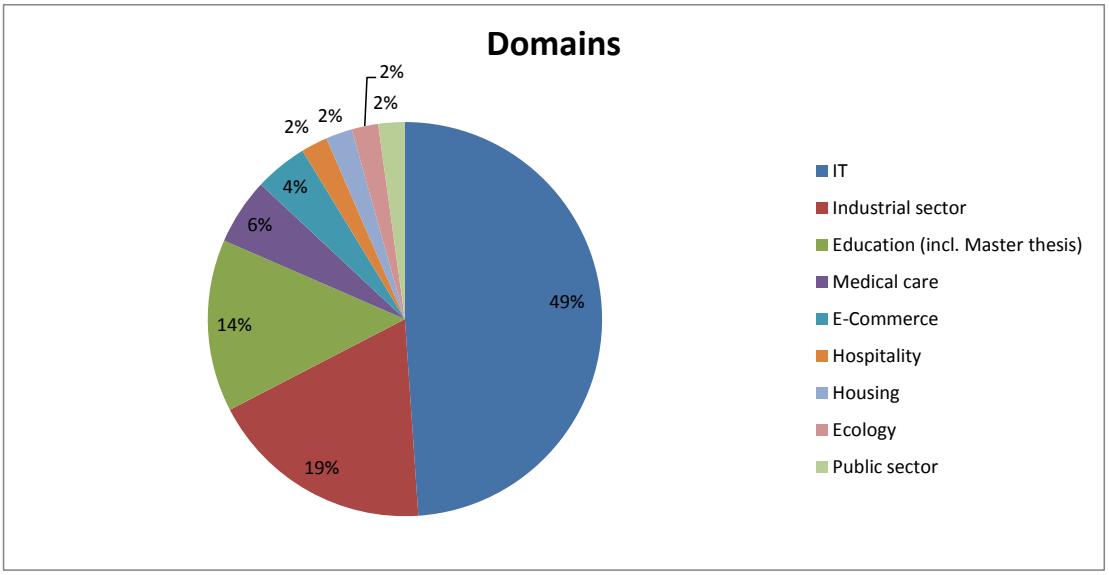
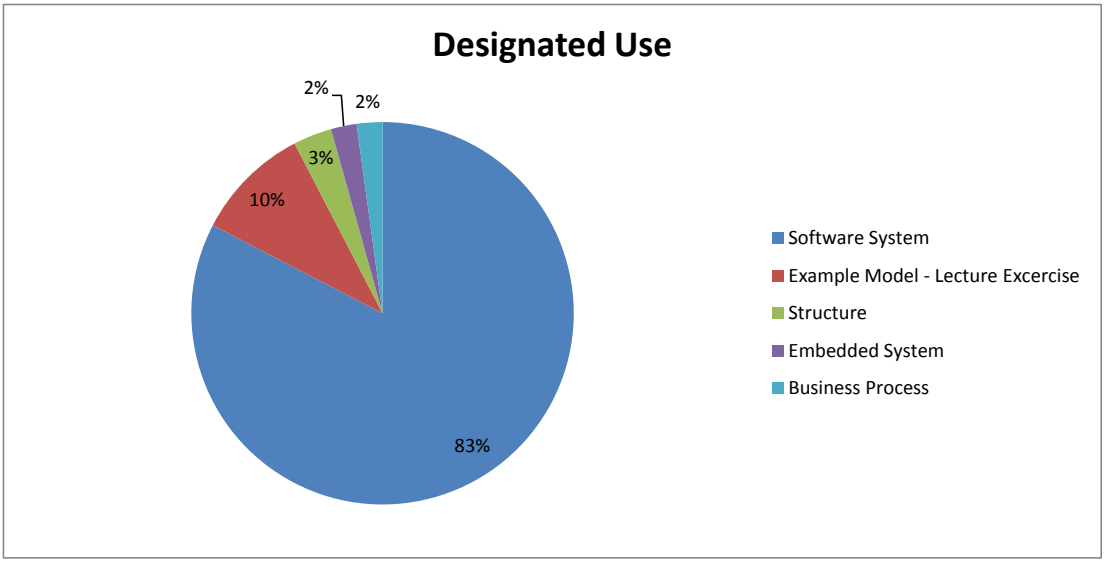


Figure 4.6: Models - Designated Use



The strong use of UML concepts in the models might cause the high usage for describing software systems. In the 92 models the UML language was mainly used for what it was intended for, describing software systems.

Summary of the Findings.

- Most models defined software systems.
- Half of the models were used in the IT sector, the other half were split over several sectors.

What was the Model's Size?

For the measurement of the model's size the amount of model elements a model consisted were chosen (elements, packages, diagrams, connectors). In Table 4.7 we can see statistical data about the model's size. The arithmetic mean, the median, the standard deviation, the minimum and the maximum were calculated over the amount of model elements per model. Figure 4.7 completes the results about the model sizes. The models were grouped there into 4 different model size categories, small (S) 9-100 model elements, medium (M) 101-500 model elements, large (L) 501-1000 model elements and x-large (XL) 1001-3918 model elements. For each model size group the amount of models in this group is denoted. Additional statistics (arithmetic mean, median, max, min and standard deviation) for each model size category are listed in the Tables 4.8, 4.9, 4.10, 4.11.

Interpretation. The average (arithmetic mean) amount of model elements per model is 407, the corresponding standard deviation is 656 and the median is 165. Because of the high standard deviation the distribution of elements over the models seems to be quite unequal. Further, this results in considering the median as more reliable average measurement of the dataset. There is also a huge gap between maximum and minimum. According to the median at least 50 % of the models have less then 165 model elements. 76 models (i.e., 83% of the models) have less than 501 model elements (see Figure 4.7).

Table 4.7: Statistics about the Model Sizes

Measured Values	Number of Model Elements
Arithmetic Mean	407
Standard Deviation	656
Median	165
Maximum	3918
Minimum	9

For the smallest model size group (9-100, Table 4.8) the median and the arithmetic mean is almost the same and positioned in the middle of the model size range. This indicates a normal

Figure 4.7: Model Dimensions

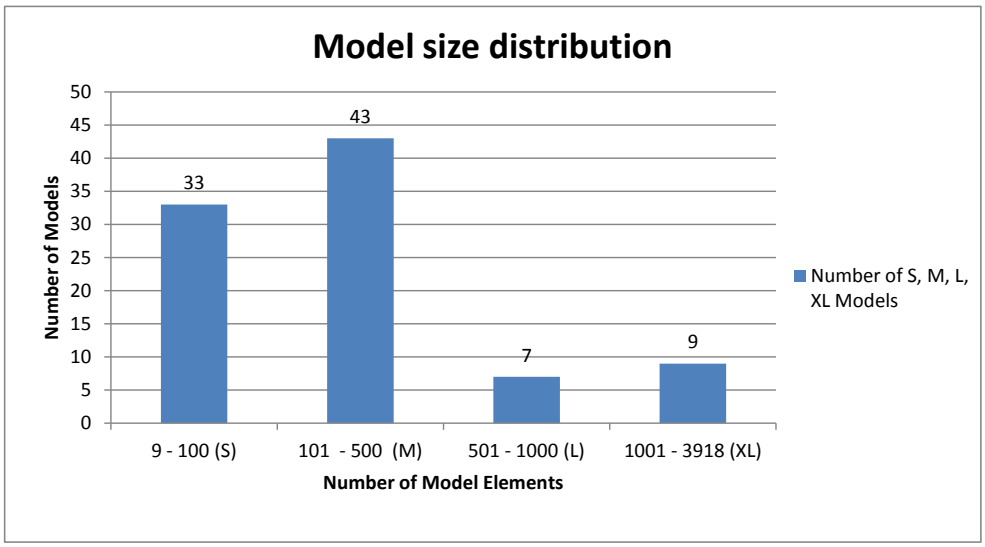


Table 4.8: Statistics about the Model Sizes within the Model Size Group Small (S)

Measured Values	Number of Model Elements
Arithmetic Mean	54
Standard Deviation	23
Median	52
Maximum	92
Minimum	9

distribution of the dataset. The empirical rule[1] states that approximately over 68% of the models in this group have a size between 31 and 77 model elements.

Table 4.9: Statistics about the Model Sizes within the Model Size Group Medium (M)

Measured Values	Number of Model Elements
Arithmetic Mean	252
Standard Deviation	109
Median	243
Maximum	473
Minimum	104

For the group M (Table 4.9), 50% of the models have 104 to 243 model elements. The median and arithmetic mean are close together and situated in the middle of this range of size group. The standard deviation is smaller than the arithmetic mean. For the overall statistics of the model sizes (Table 4.7) the standard deviation was even bigger than the arithmetic mean.

Table 4.10: Statistics about the Model Sizes within the Model Size Group Large (L)

Measured values	Number of Model Elements
Arithmetic Mean	742
Standard Deviation	112
Median	758
Maximum	901
Minimum	594

The L group (see Table 4.10) has the same statistical propositions as the other groups above. Arithmetic mean and median are close together and located in the middle of the group. The standard deviation has the same ratio to the arithmetic mean as in the statistics of model group sizes before.

In group L the models (Table 4.11) have a maximum of 3918 model elements and a minimum of 1456 model elements. The arithmetic mean and median are close together.

We can notice that the general statistics about the model sizes (Table 4.7) are positively skewed [2], this means more small models than large models exist. In contrast the arithmetic mean and the median in the model size groups are close together. This supports the choice of the group sizes

[1] *"For a distribution that is symmetrical and bell-shaped (in particular, for a normal distribution) approximately 68% of the data values will lie within 1 standard deviation on each side of the mean"* ([2], page 252).

[2] *"In a skewed distribution, the scores lend to pile up toward one end of the scale and taper off gradually at the other end. A skewed distribution with the tail on the right-hand side is said to be positively skewed."* ([12], page 50).

Table 4.11: Statistics about the Model Sizes within the Model Size Group X-Large (XL)

Measured Values	Number of Model Elements
Arithmetic Mean	2178
Standard Deviation	731
Median	2100
Maximum	3918
Minimum	1456

and their reliability. A closer look to Figure 4.7 indicates that many small models, several medium sized models and only few big models exist.

To summarize the evaluation about the model sizes, the observed models were kept small to medium. Real huge models (over 10 000 model element) were not found via Google. Of course models from real big players are missing and therefore the data is only relevant for open EA project files, reachable via the Google search engine. If in practice models are kept in this sizes can not be found out with this sample of models.

Summary of the Findings

- Not very big models were evaluated.
- In general the models were small to medium sized with in average 165 model elements.

4.2 Usage of UML Language Units

With this section the answering of the research questions start. In this section figures about the language unit distribution over all models will be presented. The elements were categorized into 10 different language units (see appendix A) to see which general UML concepts were used as most and which not. The general definition of language units, used in this research, is described, followed by the presentation of the results of the evaluations.

A Side Note to the Language Unit Conception

In the UML standard by the OMG [22] all UML elements belong to one of the specified language units. A language unit is a way to categorize similar UML concepts. One language unit is for example the language unit class. Elements like classes, attributes, operations, interfaces, associations, etc. belong to this language unit. This work also groups the concepts of UML into language units similar to the OMG specification of UML. There are some slightly differences between the OMG definitions of the different language units and how similar concepts were categorized in this work. Following the modifications to the OMG UML language unit are described.

The Language Unit Concepts. All considered UML concepts for each language unit are listed

in Appendix A. The choice which concepts the script should have to handle was strongly related to the supported modeling possibilities EA offered, the UML literature [26], [15], [25], and the OMG specification of UML [22].

Combined Language Units (Activity and Action). In the UML specifications there are the language unit action and activity. In this research paper concepts from the UML language unit activity and the UML language unit action were grouped into one language unit, called activity.

Additional Language Unit (Object). The UML concept object was analyzed separately. Hence an own language unit called object was introduced.

Extended Language Unit (Auxiliary Construct). The language unit "Auxiliary Construct" defined in the UML OMG specifications is also used but with additional concepts. Notes and constraints are units of the language unit "Auxiliary Construct" in this work.

Special treatment: Connectors. Connectors were treated in a special way. As in Chapter 2 explained connectors describe relationships between elements in EA projects. Connectors can be UML concepts like associations, generalizations or compositions or other concepts from other modeling languages. In the most cases it is difficult to classify UML connector concepts to a language unit. For example generalizations can be used between any classifiers. Consequently connectors in the analyzed models were not considered as part of any language unit. In the evaluation part connectors measure the way how and which elements were connected to each other and how often. A complete list of all considered UML relationships can be found in the Appendix A.

Special treatment: Packages. Packages are saved in an own table in the repository of EA (see Chapter 3, Figure 3.1). Consequently in this work packages are not part of any language unit and are rarely relevant in the evaluations of the models. Packages in this work are seen as EA packages and not as UML packages in the evaluation. Therefore the UML concept package is not listed in one of the considered concepts in Appendix A.1. They are treated separately. In the analysis of the model data only the number of EA packages per model is considered.

The adapted UML language units the script uses for categorizing UML concepts are listed in Table 4.12 bellow with the total amount of different UML concepts the script considers. Diagrams are not included in Table 4.12 as all UML diagrams are considered anyway, only elements, attributes an operations are counted. A complete list of all for this work relevant UML concepts can be find in Appendix A.1, grouped by language units and with the appropriate Automation Interface calls.

Table 4.12: The Language Units for the Evaluation

Language Units	Amount of Different UML Concepts Considered
Class	14
Interaction	23
State Machine	34
Use Case	3
Activity	87
Object	1
Component	4
Deployment	5
Composite	4
Auxiliary Construct	3
Total Amount of Different UML Concepts Considered in this Work (excl. Connectors and Diagrams)	199

In How Many Models are the Distinct Language Units Used?

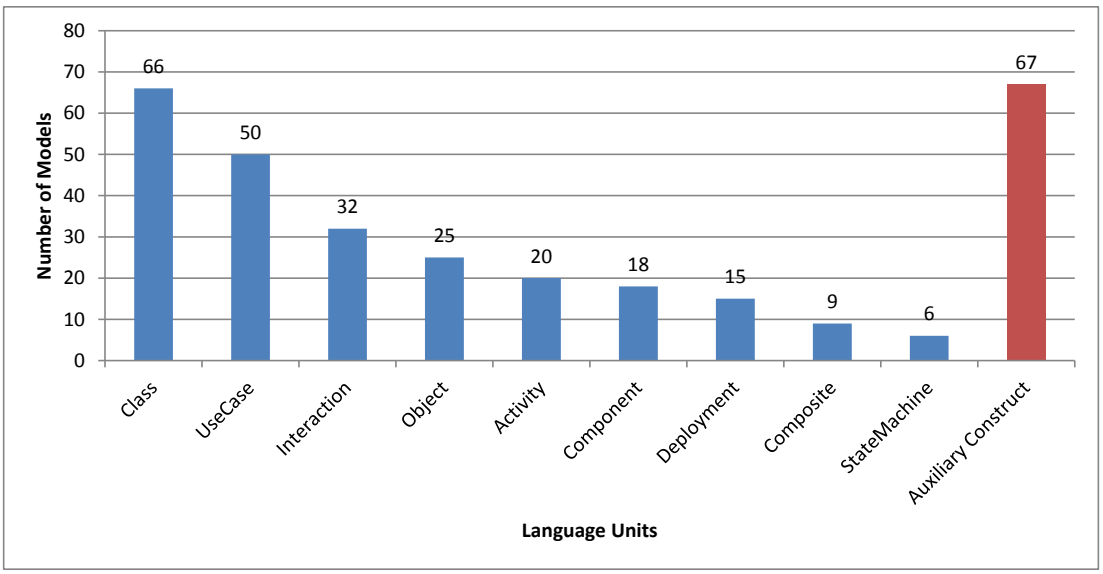
Figure 4.8 lists all considered language units with the number of models in which modeling elements, belonging to the respective language unit, were used. Only elements were regarded for calculating the distinct language unit usages in the models. Like for the model sizes analysis, connectors and diagrams were skipped. The reason for this is that some connectors (for example generalizations) can not be clearly classified to a language unit (see "A side note to the language unit conception" above). Thus only considering the elements of the language units seems to be the best and accurate approach in getting reliable data about the popularity of the language units. Moreover no differentiation between UML elements with and UML elements without stereotypes was made. For example a class extended by a stereotype still counted as member from the language unit class.

Interpretation By far the most used language unit over all models is class, which is used in 66 models out of 92. The second most used language unit is use case. In 50 models (i.e., 54% of the models), concepts from this language units were used. On the third rank we have the language unit interaction, which was used in 32 models. Object and activity constructs are fourth and fifth in the rank. Slightly more models with elements from the language unit object can be found than with activity concepts. On the next three ranks we can find the language units component, deployment and composite. Least used was the language unit state machine. Only 6 models out of 92 used state machine concepts.

A special case is the language unit auxiliary construct. Actually it is the most used language unit over all 92 sample models, but because of the usage of the elements (e.g., comment, ...), which can be used in every part of a model, it is treated separately.

Summary of the Findings

Figure 4.8: Language Unit Usage



- Class and use case UML concepts were most used.
- State machine UML concepts were least used.

How many distinct Language Units are used in Models?

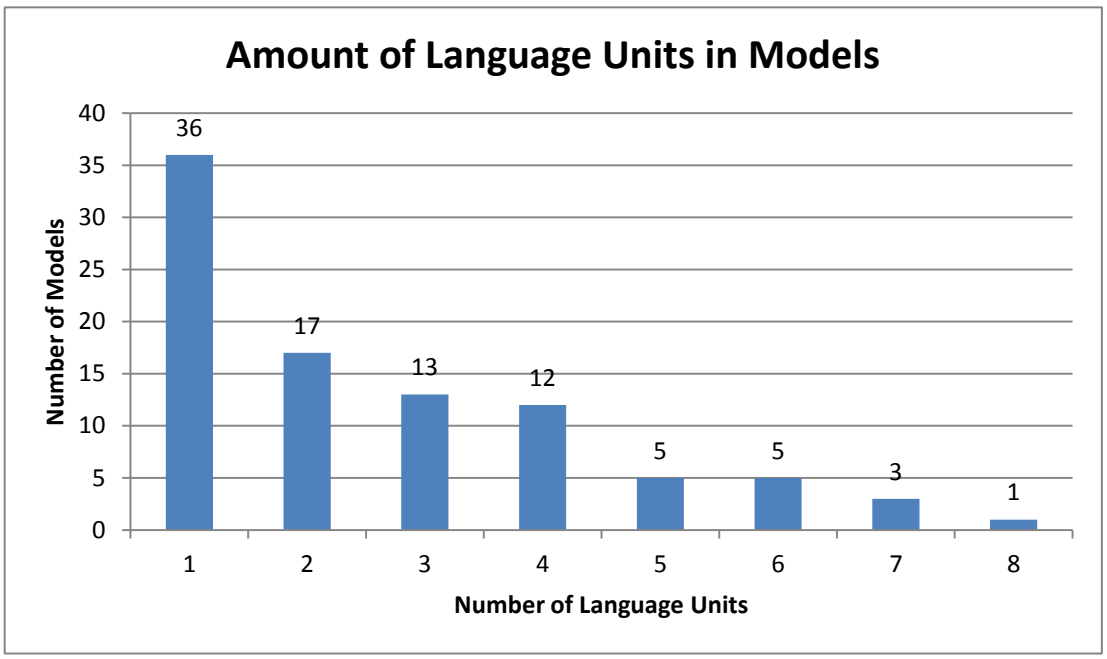
To answer this research question the amount of distinct language units used in models was analyzed. Table 4.13 lists the measured statistical data about the number of language units in models. Minimum, maximum, arithmetic mean, standard deviation, median and mode are the calculated numbers. Furthermore Figure 4.9 illustrates the amount of language units the sample models have. The language unit auxiliary construct was excluded for this evaluation.

Table 4.13: Statistics about the Number of Language Units used in Models

Measured Values	Language Units
Minimum	1
Maximum	8
Arithmetic mean	2,61956522
Standard Deviation	1,79880631
Median	2
Mode	1

Interpretation. We see on Figure 4.9 that 36 models (i.e., 39% of the models) consist of elements which only belong to one UML language unit. 53 models (i.e., 58% of the models)

Figure 4.9: Amount of Language Units used in Models



have elements which belong to 1 to 2 language units. According to this data we can assume that in 39% of the models one language unit was enough to represent and describe a system functionality. Nevertheless for the other 56 models only elements of one language unit were insufficient to describe the model. Further, we see that the distribution has a positive skew. So more models with little amount of language units exist than models with several language units.

The maximum amount of different language units in a model was 8 (see Table 4.13), which occurred ones. The mode value tells us that the most frequent amount of language units in models was 1. The standard deviation of the arithmetic mean is high. Consequently for measuring the central tendency of this distribution the median is more reliable than the arithmetic mean. Hence, in average (median) a model consist of 2 language units.

Summary of the Findings

- In average a model consisted of 2 language units.

Which Language Unit Pairs are frequently used?

To answer the research question which language unit pairs are frequently used, a matrix (see Table 4.14), illustrating the language units in rows and columns, was created. The total appearance of each language unit pair in models can be found in the cells. The numbers in the diagonal from top left to bottom right is equal to the number of models using the language unit (compare with Figure 4.8). The tool WEKA was used for calculating the occurrence of each language

unit pair. For this a list with the used language units in each model served as input for WEKA. The list was generated from the results of the models. Out of this list, with the in WEKA implemented Apriori Algorithm [14], the occurrences of all pairs could be generated. Furthermore, the language unit pairs were weighted in Table 4.15. The exact same matrix structure was used as for Table 4.14. The numbers are calculated by dividing the *amount of models with the language unit pair combination* through the *amount of models with the focused language unit*. The rows present the focused language unit and the columns the partner language unit. This gives a statement about the strength of the relation for a distinct language unit. For example at column *"UC"* on row *"Cl"* we have the value 0,50. This means that in 50% of model with elements from the language unit class also elements from the language unit use case were modeled. As closer the number to 1 as more unlikely the focused language unit will be modeled without the partner language unit and consequently as stronger the relation is.

The language unit auxiliary construct was not considered for these metrics and as above diagrams and connectors were skipped too as parts of the observed language units. For the diagram evaluations see Section 4.3.

Table 4.14: Language Unit Pairs with their occurrence in the 92 Models

Language Unit	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*66*	33	26	21	13	14	14	8	5
UC	33	*50*	25	21	12	13	14	7	4
Int	26	25	*32*	14	6	8	10	8	2
Obj	21	21	14	*25*	11	5	8	1	4
Act	13	12	6	11	*20*	4	5	2	3
Comp	14	13	8	5	4	*18*	8	8	3
Deploy	14	14	10	8	5	8	*15*	4	2
Composite	8	7	8	1	2	8	4	*9*	1
SM	5	4	2	4	3	3	2	1	*6*

Cl = Class, UC = Use Case, Int = Interaction, Obj = Object, Act = Activity
Comp = Component, Deploy = Deployment, Composite = Composite, SM = State Machine

Interpretation. Observing Table 4.14, 5 mostly used language unit pairs could be detected:

- Class/use case - in 33 models (i.e., 36 % of the models)
- Class/interaction - in 26 models (i.e., 28 % of the models)
- Use case/interaction - in 25 models (i.e., 27 % of the models)
- Class/object - in 21 models (i.e., 23 % of the models)

Table 4.15: Relation Strengths of the Language Unit Pairs

Language Unit	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*1,00*	0,50	0,39	0,32	0,20	0,21	0,21	0,12	0,08
UC	0,66	*1,00*	0,50	0,42	0,24	0,26	0,28	0,14	0,08
Int	0,81	0,78	*1,00*	0,44	0,19	0,25	0,31	0,25	0,06
Obj	0,84	0,84	0,56	*1,00*	0,44	0,20	0,32	0,04	0,16
Act	0,65	0,60	0,30	0,55	*1,00*	0,20	0,25	0,10	0,15
Comp	0,78	0,72	0,44	0,28	0,22	*1,00*	0,44	0,44	0,17
Deploy	0,93	0,93	0,67	0,53	0,33	0,53	*1,00*	0,27	0,13
Composite	0,89	0,78	0,89	0,11	0,22	0,89	0,44	*1,00*	0,11
SM	0,83	0,67	0,33	0,67	0,50	0,50	0,33	0,17	*1,00*

Cl = Class, UC = Use Case, Int = Interaction, Obj = Object, Act = Activity
Comp = Component, Deploy = Deployment, Composite = Composite, SM = State Machine

- Use case/object - in 21 models (i.e., 23 % of the models)

The other language unit pairs have much less relevance. The high numbers are all located on the top left area of the matrix, involving the language units class, use case, interaction and object. As more it goes to the bottom right, as lower are the numbers. This clearly shows again which language units are used as most among the sample models.

On table 4.15 we can see that the language unit class had not a strong relation to any of the other language units. The strongest relation was with use case concepts; 50% of models with class concepts also had use case concepts. Similar is the result for the language unit use case. Although in 66% of the models using the use case language unit also class language unit concepts were used. In contrast the language unit interaction was strongly modeled together with the language units class and use case. Class concepts were found in 81% of models with interactions and use case concepts in 78% of models with interactions. Also the language unit object was strongly modeled with class and use case concepts. For the language unit activity similarities as for class and use case exist. It seems that for activities other language units had not a big relevance. Component concepts could be mostly found with class, second with use case concepts. In nearly all models with the language unit deployment, class and use case concepts could be found. These 2 pairs have the strongest relation (both 0,93) through all language unit pairs. Composite concepts appeared in the most models together with parts of the language units class, use case, interaction or component. The rarely modeled language unit state machine was found with class concepts as most.

Frequently used over the 92 sample models were combinations between class, use case, object and interaction concepts in a model. Few relevant were the other combinations, including the language units activity, component, composite, deployment and state machines. This also reflects the usage rate of the distinct language units. Furthermore some language units seem to be more dependent than others. Class, use case and activity concepts had no strong relations

to other language units. This tells us that other language units did not influence much the occurrence of these 3 language units in models. On the other hand the language units interaction, object, component, composite and deployment were strongly related to classes and use cases. State machine concepts are excluded from this conclusion as there was not enough data available about this language unit.

Summary of the Findings

- The frequently used language unit pairs consist mainly of the language units class, use case, object and interaction.
- The language units class, use case and activity had no high relation strengths to other language units.
- The language units interaction, object, component, composite and deployment were strongly related to class and use case.

Which Language Units are Frequently used in Combination?

Cluster analysis was used to find out which language unit combination patterns could be found in the 92 models. With clustering a set of data patterns can be obtained telling which data is correlated and which not. For generating clusters out of the language unit sets the tool WEKA was used with the SimpleKMeans Algorithm[3]. As distance metric the Euclidean distance was selected. Items within a cluster have small distances, items between clusters have long distances. In terms of language units the distances are as closer as more often the language units were in combination together in the models. Detailed information about clustering and clustering methods can be found in [14]. Figure 4.10 shows the cluster output of WEKA. The *"Attribute"* column lists the different language units. *"Full Data"* represents in percentage in how many of the 92 sample models the respective *"Attribute"* (language unit) could be found, in sum all values in *"Full Data"* are 100. The columns with the headline *"0"* to *"4"* are the calculated clusters, the number in brackets show the amount of models a cluster have. The numbers in these cells tell us how many models in a cluster (in percentage) have the distinct language unit. 1 means that the language unit is in every model and 0 means that the language unit is in no model in the cluster. The clusters are disjunct so a model can only appear in one cluster.

Interpretation: 5 reliable clusters were found. Followed the characteristics of each cluster will be explained.

- **Cluster 0.** This model group has the focus on 6 language units: Class, use case, interaction, component, deployment and composite. Nearly not relevant are the language units activity, object and state machine.

[3]Data mining with WEKA. http://www.ibm.com/developerworks/opensource/library/os-weka2/. Accessed: 2013-16-01

Figure 4.10: WEKA Cluster Output

Attribute	Full Data (92)	Cluster# 0 (9)	1 (10)	2 (24)	3 (35)	4 (14)
Class	0.7174	1	0.9	0	1	0.9286
UseCase	0.5435	1	1	0.625	0.1429	0.7857
Interaction	0.3478	0.8889	0.2	0.2083	0.0857	1
Object	0.2717	0.2222	1	0.0833	0.0286	0.7143
Activity	0.2174	0.2222	0.9	0.25	0.0571	0.0714
Component	0.1957	0.7778	0.2	0.1667	0.1143	0.0714
Deployment	0.163	0.7778	0.4	0.0417	0.0286	0.1429
Composite	0.0978	0.6667	0	0.0417	0.0571	0
StateMachine	0.0652	0.1111	0.3	0.0417	0.0286	0

Clustered Instances

0	9 (10%)
1	10 (11%)
2	24 (26%)
3	35 (38%)
4	14 (15%)

- **Cluster 1.** In this group use case and objects are modeled in every model and activity and class in 90% of the models. Therefore 4 language units are characteristic for this group. The other language units have no big relevance.
- **Cluster 2.** This cluster can be also called the “use case“ group. Use case concepts were modeled as most. Classes can not be found in any model. The other language units have a low representing.
- **Cluster 3.** In this group the language unit class is the center. We can call it the “class“ group. Other language units are rarely modeled.
- **Cluster 4.** The language unit quartet “class - use case - interaction - object“ gives the group his name. These language units appear in the most models of this cluster. All the other language units are not relevant. State machine and composite concepts even appear in no model

The clusters support some of the results from the previous research question. Many models with focuses on class (Cluster 3) or use cases (Cluster 2) exist. Close relations between class, use case, object and interaction could be found as well. In Cluster 4 all 4 language units appeared in Cluster 1 class, use case and interaction and in Cluster 2 class, use case and object had the main weight in the models. These all reflects the language unit pair metrics of the question above. Additional the combination between the language units class, use case, activity and objects could

be obtained.

In summary the following 3 additional common language unit combinations were found with the clustering method:

- class - use case - interaction - component - deployment - composite
- class - use case - object - activity
- class - use case - interaction - object

With the common language unit pairs from the section above, this completes the list of the common language unit combinations.

Summary of the Findings

- class - use case - interaction - component - deployment - composite is a common language unit combination
- class - use case - object - activity is a common language unit combination
- class - use case - interaction - object is a common language unit combination

Limitations/Future Work. The clustering analysis in this work are only relevant for the 92 sample models. Much more reliable would be real big sets of UML models.

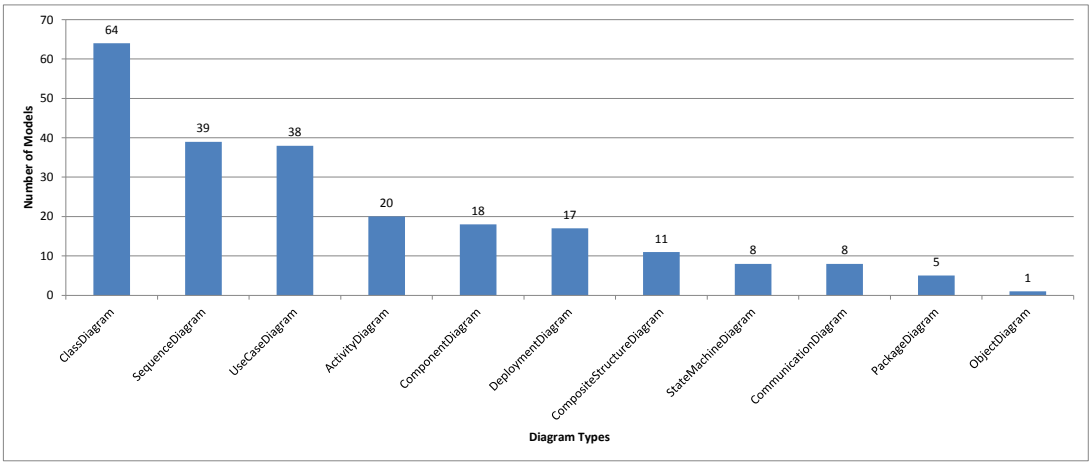
4.3 Usage of UML Diagrams

In this section the UML diagram type distribution over the 92 sample models will be presented as well as which diagram types appear mostly together in models. Further the content of each diagram type is analyzed over all models.

In how many Models are the distinct Diagram Types used?

For each model the total amount of different used diagram types were counted to answer this research question. Figure 4.11 presents in a bar chart the amount of models having a distinct diagram type.

Figure 4.11: UML Diagram Distribution - in Models



Interpretation. The class diagram is the most used diagram type, followed by the sequence diagram and the use case diagram. These can be considered as the top 3 used UML diagram types. In the mid rank of diagram usage over the models are the activity, component, deployment and composite structure diagrams. Not many diagrams are from type state machine, communication, package and object. In no single model profil, timing and interaction overview diagrams could be found.

Comparing these results with Figure 4.8 on page 41 we can see a few differences. Obviously objects were not modeled in object diagrams and EA packages not in package diagrams. EA packages are essential for every EA project, without a package it is not possible to model elements or diagrams. For the modelers of the 92 models creating additional package diagrams to visualize the package structure of the models were obviously seen as unnecessary. Objects took part in different diagram types (cf. 4.20 on page 58). Presenting only interactions between objects in own object diagrams were seen as completely unnecessary in the sample models as objects were used in various interactions with other language unit concepts. Further, more use

cases than use case diagrams exist. In contrast more diagram types (sequence and communication diagrams) corresponded to the language unit interaction exist than interaction concepts. The reason for this was that in interaction diagrams also concepts from other language units than from interactions were found. Some interaction diagrams only consisted of classes, actors or use cases (cf. 4.20) The other diagram types and language units appear in similar amount over the models. In one of the further results the content of the diagrams is represented. This will help to clarify in which diagrams the distinct language units were modeled.

Summary of the Findings

- The class diagram are the most used UML diagram type, followed by sequence and use case diagrams.
- Only 1 object diagram exist. Therefore in at least 24 out of 25 models with object concepts, objects were not visualized in object diagrams.
- In 39 models sequence and in 8 models communication diagrams exist. Both are diagrams for visualizing interaction concepts. Hence models exist with interaction diagrams but no element from the language unit interaction.
- The other language units appear with their corresponding diagram types in almost the same amount of models.

How many distinct UML Diagram Types are used in Models?

As for the language units also for diagrams the amount of different diagram types in models is measured. Table 4.16 lists the measured statistical data about the number of diagram types in models. Minimum, maximum, arithmetic mean, standard deviation, median and mode are the calculated numbers. Furthermore Figure 4.12 illustrates the amount of distinct UML diagram types in models.

Interpretation. We see on Figure 4.12 that 41 models (i.e., 45% of the models) consist of one UML diagram type. 15 models had 2 different diagram types and 14 models had 3 different diagram types. According to this data we can assume that in 45% of the models one diagram type was enough to visualize a systems functionality. Nevertheless for the other 51 models only one diagram type was insufficient to picture the different parts of a system. Furthermore, we see that the distribution has a positive skew. The density of data is situated on the left side. So more models with little amount of diagram types exist than models with several diagram types.

The maximum amount of different diagram types in a model was 8 and the minimum 0 (see Table 4.16), both occurred ones. In one model no UML diagram was modeled. The mode value tells us that the most frequent amount of diagram types in models was 1. The standard deviation of the arithmetic mean is high. Consequently for measuring the central tendency of this distribution the median is more reliable than the arithmetic mean. Hence, in average (median) a model consist of diagrams of 2 distinct types.

Figure 4.12 has almost the same shape as Figure 4.9 on page 42. Only a few exceptions exist like the different minimum of the distribution and that more models with only one distinct diagram type exist than with only one language unit.

Figure 4.12: Amount of distinct UML Diagram Types in Models

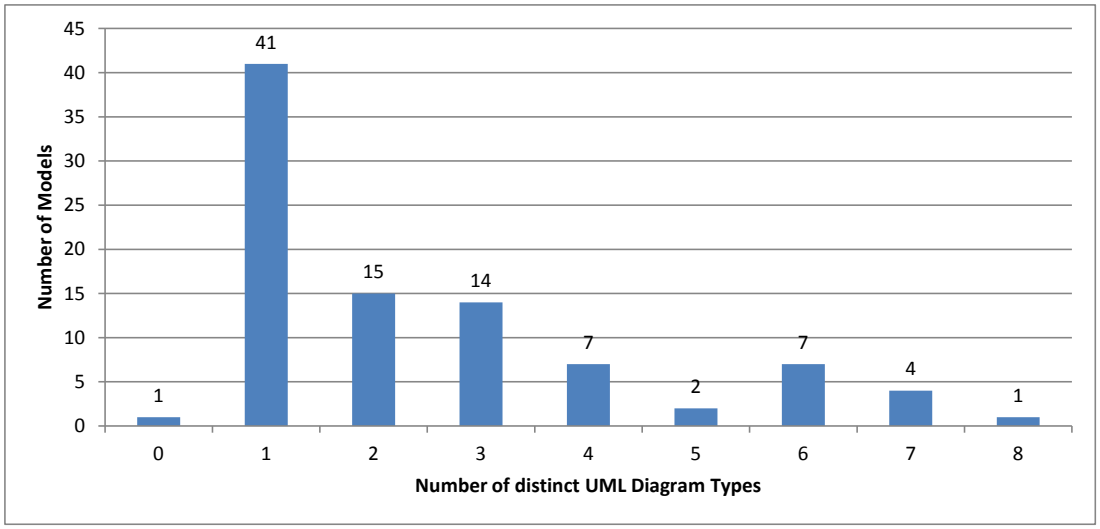


Table 4.16: Statistics about the Number of distinct Diagram Types in Models

Measured Value	Diagram Types
Maximum	8
Minimum	0
Arithemtic Mean	2,49
Standard Deviation	1,90
Median	2
Mode	1

Summary of the Findings

- In average 2 types of diagrams were found in the models.

What is the Usage of Diagrams over the Models?

In Table 4.17 we see the distribution of the diagram types in the models. Each distribution only considers models with the respective diagram type. For example the average amount of class diagrams in models with class diagrams. The statistics measuring the distribution are the maximum, the arithmetic mean, the median, the standard deviation and the mode.

Table 4.17: Diagram Distribution over Models

	Maximum	Arithmetic Mean	Median	Standard Deviation	Mode
Class Diagram	79	6,17	3	10,69	1
Sequence Diagram	15	2,64	1	2,65	1
Use Case Diagram	26	3,5	3	4,02	3
Activity Diagram	25	4,6	2,5	5,78	1
Component Diagram	14	3,22	1,5	3,55	1
Deployment Diagram	11	4,71	7	3,51	1
Composite Structure Diagram	3	1,27	1	0,62	1
State Machine Diagram	6	2,38	1	2,12	1
Communication Diagram	25	8,38	1	9,86	1
Package Diagram	10	3	1	3,52	1
Object Diagram	1	1	1	0	1

Interpretation. In the most cases the standard deviation is relatively high. Therefore the median is more reliable for presenting the average than the arithmetic mean. Models with class and use case diagrams have a high median. In average 3 class or use case diagrams are in models with class or use case diagrams. It seems that for the 92 sample models more visualizations views were needed to describe class structure and use cases. The biggest median was calculated for deployment diagrams. In 50% of the models with deployment diagrams 7 or more deployment diagrams could be found. Though the mode value is only 1, which means the most frequent amount of deployment diagrams in models is 1. So there were some outstanding exception within the sample models with a high number of deployment diagrams. Models with activity diagrams had an average of 2,5 activity diagrams per model. Therefore for activity models in average more than one visualization view was needed. 1,5 component diagrams could be found in average in the models. The median values of all the other diagrams occurrences in models is 1.

Summary of the Findings

- In average deployment, class, use case, component and activity diagrams were modeled more than one time in a model.
- All the other UML diagram types were in average modeled 1 time in a model.

Which UML Diagram Type Pairs are frequently used?

The exact same approach was done as for the calculation of the frequent language unit pairs in Chapter 4.2. Table 4.18 and Table 4.19 have the same structure and meaning than Table 4.14 and Table 4.15 from Chapter 4.2. The total amount of models in which the respective diagram type pair combination can be found is noted in the cells (Table 4.18). Table 4.19 lists the relation strengths of the diagram pairs.

Table 4.18: Diagram Pairs with their occurrence in the 92 Models

Diagram Type	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*64*	27	34	0	15	13	16	10	7
UC	27	*38*	26	0	11	11	15	5	6
Int	34	26	*43*	0	11	13	15	7	6
Obj	0	0	0	*1*	0	0	0	0	0
Act	15	11	11	0	*20*	6	7	3	4
Comp	13	11	13	0	6	*18*	9	9	4
Deploy	16	15	15	0	7	9	*17*	5	4
Composite	10	5	7	0	3	9	5	*11*	1
SM	7	6	6	0	4	4	4	1	*8*

Cl = Class Diagram, UC = Use Case Diagram, Int = Interaction Diagrams (Sequence Diagram + Communication Diagram)
Obj = Object Diagram, Act = Activity Diagram, Comp = Component Diagram
Deploy = Deployment Diagram, SM = State Machine Diagram

Table 4.19: Relation Strengths of the Diagram Pairs

Diagram Type	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*1,00*	0,42	0,53	0,00	0,23	0,20	0,25	0,16	0,11
UC	0,71	*1,00*	0,68	0,00	0,29	0,29	0,39	0,13	0,16
Int	0,79	0,60	*1,00*	0,00	0,26	0,30	0,35	0,16	0,14
Obj	0,00	0,00	0,00	*1,00*	0,00	0,00	0,00	0,00	0,00
Act	0,75	0,55	0,55	0,00	*1,00*	0,30	0,35	0,15	0,20
Comp	0,72	0,61	0,72	0,00	0,33	*1,00*	0,50	0,50	0,22
Deploy	0,94	0,88	0,88	0,00	0,41	0,53	*1,00*	0,29	0,24
Composite	0,91	0,45	0,64	0,00	0,27	0,82	0,45	*1,00*	0,09
SM	0,88	0,75	0,75	0,00	0,50	0,50	0,50	0,13	*1,00*

Cl = Class Diagram, UC = Use Case Diagram, Int = Interaction Diagrams (Sequence Diagram + Communication Diagram)
Obj = Object Diagram, Act = Activity Diagram, Comp = Component Diagram
Deploy = Deployment Diagram, SM = State Machine Diagram

Interpretation. The following top 3 diagram type pairs could be detected from Table 4.18:

- Class/interaction - in 34 models (i.e., 37 % of the models)
- Use case/interaction - in 26 models (i.e., 28 % of the models)
- Class/use case - in 27 models (i.e., 29 % of the models)

The findings here corresponds to the results from Chapter 4.2 in many ways. Class, interaction and use case were considered as the most relevant diagram and language unit pairs. The only exception is that object diagrams had no relevance at all in the 92 models and therefore no diagram pair with object diagrams could be found. The reason for this is because objects were mainly modeled in interaction diagrams (cf. Table 4.3). The other diagram type pairs have much less relevance like the related language unit pairs. This is also shows illustrative the matrix (Table 4.18). As for the matrix (Table 4.14) the high numbers are all located on the top left area of the matrix (Table 4.18). As more it goes to the bottom right, as lower are the numbers. This clearly shows again the ranking of the diagram types among the sample models.

Similar relation weights between the UML diagram type pairs exist as for the language unit pairs (compare Table 4.15 with Table 4.19). What can be noticed is that on one hand interaction diagrams had stronger dependencies to the other UML diagram types as the language unit interaction to other language units. On the other hand object diagrams were not relevant at all for any UML diagram type.

Summary of the Findings

- Diagram pairs appeared in similar amount together in the models as the language unit pairs.
- The most diagrams were more dependent to interaction diagrams than the language units to the interaction language unit. Exeptions are composite/interaction and object/interaction pairs.
- Object diagrams had no relevance.

Which UML Diagram Type are frequently used in Combinations?

As for the language unit combinations also for finding UML diagram type combination patterns clustering was used. The exactly same methods as in Chapter 4.2 were applied for calculating the clusters. The WEKA output is depicted in Figure 4.13, which has the same structure as Figure 4.10, with the exception that instead of language units diagram types were observed.

Interpretation. 5 reliable clusters were found. Followed the characteristics of each cluster will be explained.

- **Cluster 0.** This cluster can be also called the “use case group“. The main focus lays definitely on use case diagrams, which is modeled as most. The other diagram types have a low representing.

Figure 4.13: WEKA Cluster Output

Attribute	Full Data (90)	Cluster# 0 (24)	1 (11)	2 (14)	3 (12)	4 (29)
ClassDiagram	0.7111	0.5833	0.5455	0.5714	1	0.8276
UseCaseDiagram	0.4222	1	0.2727	0	0.9167	0
InteractionDiagrams	0.4778	0.625	0.1818	1	1	0
ObjectDiagram	0.0111	0	0	0	0	0.0345
ActivityDiagram	0.2222	0.0833	1	0	0.5833	0
ComponentDiagram	0.2	0.125	0.0909	0.1429	0.75	0.1034
DeploymentDiagram	0.1889	0.1667	0	0	1	0.0345
CompositeDiagram	0.1222	0	0.0909	0.1429	0.4167	0.1034
StateMachineDiagram	0.0889	0.0833	0	0	0.3333	0.069

Clustered Instances

0	24 (27%)
1	11 (12%)
2	14 (16%)
3	12 (13%)
4	29 (32%)

- **Cluster 1.** In this group the main focus lays on activity diagrams. In every model of this cluster activity diagrams appeared. The other diagrams had no serious presence in the models. We can call this cluster the “activity group“.
- **Cluster 2.** Interaction diagrams characterize this group. A low role had class diagrams which appear in 50% of the models. The other diagram types are not important in this cluster. The name “interaction group“ fits to this cluster.
- **Cluster 3.** The diagram types “class - use case - interaction - component - deployment“ gives the group his name. These diagram types appear in the most models of this cluster. The other diagram types are not relevant. State machine and composite concepts even appear in no model
- **Cluster 4.** In this group the diagram type class is the center. We can call it the “class group“. Other diagram types are rarely modeled.

The calculated diagram clusters are a bit different to the language unit clusters. There are 2 more clusters only focusing on one diagram type (cluster 1 and cluster 2), and only 1 cluster (cluster 3) where several diagram types are used.

In summary the following additional common diagram type combination were found with the clustering method: class - use case - interaction - component - deployment

Summary of the Findings

- 2 more clusters (interaction and activity clusters) focusing on one “type“ were found than for the language unit clusters
- 4 clusters were characterized by only one diagram type, which were class, interaction, activity and use case diagrams.
- Class - use case - interaction - component - deployment is an common diagram type combination among the models.

What is the Content of UML Diagrams?

In Table 4.20 we can find a statistical evaluation about the diagram content. The diagrams were only searched for elements of a certain UML type (cf. 4.20). The observed UML element types are *“class“*, *“action“*, *“activity“*, *“interaction“*, *“life line“*, *“use case“*, *“actor“* *“state“*, *“state machine“*, *“object“* and *“component“*. These chosen UML element types were seen as sufficient to get data about which language units were usually visualized in which diagram types over the sample models. The numbers in Table 4.20 declare the total amount of an element of the respective type which were visualized by the given diagram type. The numbers in bracket stands for the number of models this combination of diagram type and UML element type could be found.

Table 4.20: Diagram Content

Diagram/Concept	Cl	Act	Action	Int	LL	UC	Actor	SM	State	Obj	Comp
ClassDiagram	**2723 (63)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**2 (1)**	0 (0)	**5 (1)**	**32 (1)**	0 (0)
PackageDiagram	**5 (3)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
ComponentDiagram	**2 (1)**	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**5 (3)**	0 (0)	0 (0)	0 (0)	**261 (13)**
DeploymentDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**44 (1)**	**20 (4)**
SequenceDiagram	**21 (3)**	0 (0)	0 (0)	0 (0)	**574 (29)**	0 (0)	**68 (24)**	0 (0)	0 (0)	**41 (6)**	**43 (4)**
ActivityDiagram	0 (0)	**353 (16)**	**56 (7)**	0 (0)	0 (0)	0 (0)	**18 (2)**	0 (0)	0 (0)	**11 (2)**	**11 (1)**
UseCaseDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**475 (34)**	**185 (34)**	0 (0)	0 (0)	**3 (1)**	0 (0)
StateMachineDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**74 (6)**	0 (0)	0 (0)
CommunicationDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**58 (6)**	0 (0)	0 (0)	**375 (7)**	0 (0)
CompositeStructureDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**15 (1)**	**7 (7)**
ObjectDiagram	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**1 (1)**	0 (0)	0 (0)	**12 (1)**	0 (0)

Cl = Class, UC = Use Case, Int = Interaction, LL = LifeLine

Comp = Component, Act = Activity, SM = State Machine, Obj = Object

Interpretation. Followed the content of each diagram type is analyzed:

- **Class diagram.** The results in Table 4.20 show that in class diagrams mainly classes were visualized. From the 64 models with class diagrams (cf. 4.3), in 63 models classes were found in class diagrams. So only one class diagram exist without any modeled class. In all 2723 classes could be found in the 63 models with classes in class diagrams. Few actors, states and objects could be found as well in class diagrams but only in one model each.
- **Package diagram.** Only classes could be found in package diagrams
- **Component diagram.** In component diagrams mostly components were visualized.
- **Deployment diagram.** As deployment concepts were not considered, components were modeled as most (4 models with components in deployment diagrams exist). In one model 44 objects were modeled in deployment diagrams.
- **Sequence diagram.** life lines and actors were usually found in sequence diagrams.
- **Activity diagram.** Activities and actions are mostly modeled in activity diagrams. Many activities and only few actions were modeled in activity diagrams.
- **Use case diagram.** Use case diagrams consist mainly from concepts of the language unit use case.
- **State machine diagram.** In state machine diagrams only states could be found. None of the other considered element types were modeled in state machine diagrams. Even none of the state machine elements could be found in state machine diagrams.
- **Communication diagram.** The most found concept were objects (in 7 models) followed by actors (in 6 models).
- **Composite structure diagram.** Composite concepts were not considered. In the most models with composite structure diagrams component elements could be found.
- **Object diagram.** The only found object diagram in the 92 sample models had 12 object elements and 1 actor.

In summary concepts from the language unit object and use case had an important role in interaction diagram types. In the other UML diagram types mainly the concepts from the associated language unit were found.

Summary of the Findings

- Lifelines and actors were significantly modeled in sequence diagrams. In the other diagram types mainly the UML concepts which are associated to these diagrams were found.
- Communication diagrams only existed of actors and objects. This is exactly for what communication diagrams should be used for.

Limitations/Future Work. The considered element types should have given a good indicator which language units are visualized with which diagram type. Studies about all model elements in diagrams could lead to further and more accurate evaluations. Furthermore, data were only obtained for all diagrams of a type but not for the individual diagrams. This restricts the evaluation in analyzing individual diagrams about their content. By observing the content of a distinct diagram, they can be categorized into multi-view/unit-view-diagrams. Where a multi view diagram means that elements of types of different language units are modeled and a unit view diagram only consist of concepts from one language unit. This might be interesting for future work. Furthermore the diagram sizes can be interesting for further work as well. Exist there many diagrams with only a few elements or do diagrams have usually many elements modeled? Subsequently does the diagram type and used language unit influence the size of a diagram? Are there diagram type/language unit combinations which results in bigger diagrams than others? How are specific element types distributed among certain diagram types? There are many opportunities for possible future work related tho this research question.

4.4 Usage of UML Concepts

In this Chapter we investigate the use of the considered UML concepts of each language unit. The list of all concepts the script is able to read out can be found in the appendix A. It will be cleared now which concepts from the language units were used, which not, which one quite often and how much of a language unit was not used. For each language unit the used elements and connectors between those elements are listed in the next tables in this Chapter. The next research questions about the used language unit concepts use all the same statistics and table structure. Therefore, only for the following research question the statistical measurements and the table properties will be explained in detail.

Which Class Concepts are used?

Table 4.21 and Table 4.22 holds data about the usage of the concepts of the language unit class. In Table 4.21 all elements which represent an UML concept are listed with their distribution over the 92 models. The column `Concept` stands for the UML concept. `In Models` stands for the number of models having this UML concept. `Total` represents the total amount of the respective concept over all models. The other columns indicate how the elements of the UML concept are distributed over the models. The measured statistics are the maximum (max), minimum (min), arithmetic mean (am), median (med) and the standard deviation of the arithmetic mean (sd). In the last row of the table we can find the total amount of models with concepts from the language unit class and the total amount of elements which were detected as language unit class concepts. Table A.1 in the appendix lists all UML class concepts the script considers in the 92 models.

In Table 4.22 all connectors which are modeled as relationship between concepts from the language unit class are depicted. Only relationships between elements of the language unit class were considered. The column `Relationship` holds the relationship concept. `In Models`

gives information about the amount of models the connector was modeled between concepts from the language unit class. The total occurrence of the connector in all models stands in column `Total`. The same statistical data is measured for connectors as for elements. In the last row the number of models having relationships between concepts from the language unit class is depicted as well as the total amount of relationships between class concepts over all models.

Table 4.21: Class Concepts Distribution over the Models

Concept	In Models (66)	Total	Max	Min	AM	Med	SD
Class	65	2 124	214	1	32,68	18	41,74
- (Class) Abstract Class	(14)	(48)	10	2	3,43	2,5	2,13
- (Class) Association Class	(3)	(7)	4	1	2,33	2	1,25
- (Class) Active Class	(0)	(0)	0	0	0	0	0
- (Class) Parameterized Class	(0)	(0)	0	0	0	0	0
Class Attribute	51	7 812	1 457	1	153,18	44	297,75
Class Operation	43	6 851	861	1	159,33	61	242,50
Class Operation Parameter	32	4 262	861	1	133,19	25	213,04
Interface	28	180	65	1	6,43	1,5	13,12
Enumeration	10	75	31	1	7,50	2	10,16
DataType	1	1	1	1	1,00	1	0,00
Signal	0	0	0	0	0	0	0
N-ary Association	0	0	0	0	0	0	0
Total	66	21 305					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. By far the most used UML class concept was the class by itself. 65 out of 66 models which uses the language unit class had classes. This means that over 98% of the models, which have members of the language unit class implemented, had class elements. By manually observing the one model without classes it came out that this model only had one concept from the language unit class modeled which were interfaces. Not surprisingly followed by class attributes and class operations. Operations appear in less models than attributes. If we look at the proportion between class to attributes (2124/7812) and class to operations (2124/6851) we can assume that in general a class consisted of multiple attributes and operations in the sample models. Parameters in operations also seems to be quite popular because in 32 of the 43 models with class operations also operation parameters were modeled. The proportion between operation to operation parameter (6851/4262) concludes that operation parameter where widely used in operations. Also still widely used are interfaces. 28 out of 66 class models have interfaces which means over 42% of all class models contain interfaces. Little relevant are abstract classes and enumerations in the 66 class models. Rarely important seems to be the other concepts. Only in 3 models association classes were modeled and in no models we could find any signal, active class, parameterized class or N-ary association. These UML class concepts had no relevance in

Table 4.22: Class Relationship Distribution over the Models

Relationship	In Models (62)	Total	Max	Min	AM	Med	SD
Association	54	2141	714	1	39,65	14	103,63
Generalization	40	389	46	1	9,73	4	11,90
Realisation	27	177	51	1	6,56	2	11,43
Dependency	20	247	96	1	12,35	8	19,92
Aggregation	19	637	546	1	33,53	5	120,84
Composition	12	51	17	1	4,25	2	4,49
Message Synchron	3	31	20	2	10,33	9	7,41
Unclassified Connector	2	6	3	3	3,00	3	0,00
Use Dependency	2	3	2	1	1,50	1,5	0,50
Message Asynchron	1	5	5	5	5,00	5	0,00
Instantiate	1	9	9	9	9,00	9	0,00
Total	62	3 696					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

the sample models.

If we take a look at the distribution of each UML class concept several conclusions can be drawn. The distributions are in general highly asymmetric. Except for the data type all other concept distributions have a positive skew. Their median is always lower than the arithmetic mean. Further, the standard deviation is very high, sometimes even higher than the arithmetic mean. Subsequently the median is more reliable for measuring the average as the arithmetic mean in all cases. The highest average could be measured for class operations. In average 61 operations were modeled in models with class operations. Class attributes were modeled 44 times in average per model, followed by class operation parameters with an average of 25 and classes with an average of 18 per model. On the other hand in models with interfaces only 1,5 interfaces in average could be found. In models with abstract classes 2,5 were modeled in average and in models with enumerations 2 enumerations were modeled in average.

In case of used relationships between UML class concepts the most popular one was the association (in 54 class models) followed by the generalization (in 40 class models). 177 Realisations could be found in 27 models. Dependencies were modeled in 20 class models, aggregations in 19 class models and compositions in 12 class models. The other 5 connectors ranked at the last places can be seen as the EA freedom of modeling. For example messages belong to the language unit interaction but in EA it is possible to model messages between classes as well. Anyway these connectors had no relevance in the class models as they are rarely modeled. The figures in Table 4.21 further say that associations between class concepts could be found in 54 models but in 65 models classes were found. It seems that in some models (exactly in 11) only the other types of relationships like generalization, aggregation and so on were used between

classes or relationships between classes and UML concepts from other language units lead to this difference. In section 4.5 relationships between different language units are evaluated.

The distributions of the distinct relationships have the same shape as the distributions of the elements. The distributions are highly asymmetric and have mostly a positive skew. Also here the median is best for measuring the average. Associations have the highest average with 14 associations per models. Compositions and realisations have the lowest average.

The most used modeling concepts in class models were classes with attributes and operations connected with associations and generalizations. The two considered UML class concepts n-ary association and signal were not used in the models. Moreover, according to these figures class operations were usually modeled with parameters together. Operation parameters were used in 32 out of 43 models with class operations (i.e., 75% of models with class operations also had operation parameters). Furthermore, Interfaces and realisations were found in an nearly congruent amount of models with almost the same occurrence and similar statistics about their distributions over the models. As we know realisations are used to realize interfaces. So the data closely assume that interfaces were usually realized by classes.

Summary of the Findings

- Class, attribute, operation, association and generalization concepts are the most used modeling concepts of the language unit class.
- The proportion between class to attributes and operations assumes that a class consisted in general of multiple attributes and operations. Further operation parameters were widely used for operations. It seems that in the 92 sample models the concepts of attributes, operations and operation parameters in classes were strongly used.
- Only 2 considered UML class concepts were not used.
- 75% of models with class operations also had operation parameters.
- It seems interfaces were usually realized, as interfaces and realisations appeared in similiar amount of models with nearly the same total amount and distribution over all models.

Limitations/Future Work. In this work only the distribution of the used relationships were observed. Mored detailed investigations could look at which concepts are connected. Further also the amount of attributes and operations a class have could be analyzed in future work as well as how many parameters operations have.

Which Use Case Concepts are used?

A list of all UML use case concepts the script checked the models for can be found in Table A.5. Table 4.23 and Table 4.24 represent the results about the usage of the language unit use case over all models.

Table 4.23: Use Case Concepts Distribution over the Models

Concept	In Models (50)	Total	Max	Min	AM	Med	SD
Actor	44	257	32	1	5,84	3	7,11
Boundary	37	86	17	1	2,32	1	2,78
Use Case	35	489	52	1	13,97	10	12,93
Total	50	832					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Table 4.24: Use Case Relationships Distribution over the Models

Relationship	in Models (35)	Total	Max	Min	AM	Med	SD
UseCaseLink	23	152	27	1	6,61	3	6,62
Association	18	224	40	1	12,44	9,5	11,11
Generalization	13	45	14	1	3,46	2	3,67
Include	9	76	20	1	8,44	9	5,93
Extend	8	73	41	1	9,13	3	12,69
Dependency	4	11	6	1	2,75	2	2,05
Message Synchron	2	6	5	1	3,00	3	2,00
Realisation	1	8	8	8	8,00	8	0,00
Unclassified Connector	1	2	2	2	2,00	2	0,00
Aggregation	1	1	1	1	1,00	1	0,00
Total	50	598					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. More models with actors exist than with use cases (see Table 4.23). Even more models with Boundaries exist as with use cases. The difference in the amount of models actors and use cases were used is a result of the usage of actors in interaction diagrams to represent communication partners (see Chapter 4.3). Further it seems that in at least 2 models boundaries were used as system border for other UML concepts than use case elements. The language unit use case does not have many concepts and we see that the concepts appeared in an similar amount of models.

In total the most used concept is the use case element, followed by actor and as last the boundary element. The distributions are highly asymmetric and they have all a positive skew. The standard deviation is always higher than the arithmetic mean. The maximum of each distribution also indicate the existent of some outliers. Therefore, the median measures the average best. Use cases can be found 10 times in models with use cases in average. Actors are only modeled 3 times and boundaries 1 time in average.

In 23 out of 50 models with use case elements, the connector type “UseCaseLink“ acted as relationship between use case concepts. The “UseCaseLink“ is an EA connector type which only can be modeled between use cases and actors. It can be seen as use case connection with blank behavior, nor extend neither include. So this connector type does not say much about the relationships between the use case concepts. Second most used relation type between use case concepts was the association. Generalizations were third popular relation types, which appear in at least 26% of all models with use case concepts. The specialized relationships extend and include took a minor role in the use case models. Only 9 out of 50 use case models had include relationships and 8 out of 50 had extend relationships modeled. Maybe the modelers did not know how to use include and extend relationships in use case diagrams as “UseCaseLink“ were much more often used. Rarely relevant were dependencies between use case concepts. Again also connectors (messages, realisations, aggregations) which do not belong to the language unit use case were found, but they had no relevance at all in the use case models.

We have much more models with one of the observed language unit concepts (50), than with one of the possible relation types the language unit supports (35). One of the reasons is the strong correlation between the language units use case, interaction and object (cf. Table 4.45, Table 4.50 and Table 4.47). Relations between concepts from different language units are considered in Chapter 4.5.

In the 92 sample models in over 40% boundary concepts were modeled. This clearly shows that this concept had a high relevance for the examined models. A possible conclusion could be that boundaries were also used within other language unit constructs to represent system borders or separate or highlight parts. In general use cases were more frequently modeled than actors or boundaries. The more simpler use case connection type was preferred by most of the designers within the 92 sample models. Extend and include relationships had not that necessity for use cases in the sample models. Maybe it might be a good idea to replace these UML concepts by a simple use case link like EA offers it. Further research in this area has to be done by OMG or other parties.

Summary of the Findings

- Boundary concept seems to be useful for other language units as well, to separate or highlight parts.
- Use case elements are most frequently modeled followed by actors and boundaries.
- The EA use case link type was the most used relationship between use case concepts. Surprisingly include and extend relations were no often used. Maybe the modelers had not enough knowledge about the include and extend relationships.
- 15 models with use case concepts but no relationship between use case concepts were found. An possible indicator for the high cohesion with other language units (cf. Table 4.45).

- use cases, actors, boundaries, use case links, and associations are the heart of use case models.

Which Activity Concepts are used?

As next we observe the language unit activity. In Table 4.25 and Table 4.27 data about the usage of the language unit activity is depicted. In Table 4.26 all non modeled UML concepts are listed. In Table 4.26 the concept "any other kind of Action" includes all the different action types the script is able to read out, except the "WriteVariableAction" and the "CallOperationAction" which occur in the models as seen on Table 4.25. All considered activity concepts can be found in Table A.2 of the appendix.

Table 4.25: Activity - Concepts

Concept	In Models (20)	Total	Max	Min	AM	Med	SD
ActivityElement	18	392	80	2	21,78	10	24,16
ActivityInitialNode	17	69	18	1	4,06	3	4,14
ActivityFinalNode	15	65	16	1	4,33	3	4,06
ActivityDecisionNode	15	90	19	1	6,00	4	4,76
Action	11	85	30	1	7,73	4	9,35
- (Action) Atomic-Action	(10)	(81)	29	1	8,10	4	9,44
- (Action) CallOperation-Action	(3)	(3)	1	1	1,00	1	0,00
- (Action) WriteVariable-Action	(1)	(1)	1	1	1,00	1	0,00
ActivityDecisionMergeNode	7	15	4	1	2,14	2	1,12
ForkNode	7	8	2	1	1,14	1	0,35
JoinNode	6	17	7	1	2,83	2	1,95
FlowFinalNode	6	10	3	1	1,67	1,5	0,75
ActivityPartition	5	18	8	1	3,60	3	2,42
ActivityMergeNode	5	11	3	1	2,20	2	0,75
JoinForkNode	3	5	2	1	1,67	2	0,47
DataStore	3	10	5	2	3,33	3	1,25
ObjectNode	2	5	4	1	2,50	2,5	1,50
InterruptibleActivityRegion	2	4	3	1	2,00	2	1,00
LoopNode	1	1	1	1	1,00	1	0,00
ExpansionRegion	1	3	3	3	3,00	3	0,00
- (ExpansionRegion) in Iterative Mode	1	3	3	3	3,00	3	0,00
Total	20	821					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. As for class and use case concept distributions the median is again the most reliable number for measuring the average. If we focus on the used elements from the language

Table 4.26: Activity - Not used Concepts

Concept	In Models
Any Other kind of Action (36)	0
ActionPin	0
ActivityParameter	0
CentralBufferNode	0
ConditionalNode	0
ConditionalNode with Expansion Node(s)	0
ConditionalNode with Pin(s)	0
ExceptionHandler	0
ExceptionHandler with incoming InterruptFlow(s)	0
ExpansionNode	0
ExpansionRegion in parallel mode	0
ExpansionRegion in stream mode	0
ExpansionRegion with Expansion Node(s)	0
InterruptibleActivityRegion with InterruptFlow(s)	0
LoopNode with Expansion Node(s)	0
LoopNode with Pin(s)	0
SequenceNode	0
SequenceNode with Activity Parameter(s)	0
SequenceNode with Expansion Node(s)	0
SequenceNode with Pin(s)	0
StructuredActivityNode	0
StructuredActivityNode with Expansion Node(s)	0
StructuredActivityNode with Pin(s)	0

Table 4.27: Activity Relationships Distribution over the Models

Relationship	In Models (18)	Total	Max	Min	AM	Med	SD
ControlFlow	18	812	145	4	45,11	32,5	39,16
Dependency	4	15	6	2	3,75	3,5	1,79
InformationFlow	3	32	16	6	10,67	10	4,11
ObjectFlow	1	2	2	2	2,00	2	0,00
Total	18	861					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean

Med = Median, SD = Standard Deviation

unit activity on Table 4.25 first thing we can notice is that there exist far more activities than actions. Almost 5 times more activities were modeled than actions. In average a model had 10 activities in models with activities but only 4 actions in models with actions. This is a quite surprising fact as in the UML literature actions are seen as the main component for describing activity diagrams. It seems activities were used like actions in many of the 92 models. Initial, final an decisions nodes could be found in the most activity models beside activities. In average 3 to 4 nodes were found in activity models with nodes.

In the most cases the default action, the atomic action, was implemented in the models. Only 2 other kind of actions were used out of the catalog of further 38 possible action types. The call operation action appears in 3 models and the write variable action in one model. Of course the questions occurs if the write variable action might be modeled by accident and not by purpose. It seems also that activity models were kept simple. Not many different activity elements could be found. The most models consist of activities, nodes for end, final and decision making and some actions. In 5 out of 20 models activity partitions and in 3 data store elements were found. Only in 2 models interruptible activity regions were modeled, and only in one model loop node and expansion regions.

Far longer is the list of not used concepts of the language unit activity (Table 4.26), if we consider all the special action types. There exist no activity with an activity parameter and also no single action with an action pin. Many concepts which are emphasized in the most UML guides had no importance for the 92 evaluated EA models. This indicates that the activity diagrams were not that detailed.

If we focus on the use of relationships between activity concepts we will detect two interesting facts. First, by far more control flows were used then object flows (see Table 4.27). On Table 4.26 we see that no single action pin was modeled in all 92 models. Modeled object flows between activity concepts were not detected in the 92 sample models. Second, the number of total control flows is almost double as high as the number of activity and action elements together in the 92 sample models. In average control flows were found over 32 times in the 18 models with control flows. Compared to the average of activity elements, nodes and actions this is rather high. This leads to the conclusion that in activity models very often different possible ways of activity/action flows were modeled. This might indicate that even though designer used mostly simple activity concepts for activity models, they had a certain complexity still.

Summary of the Findings

- activities were used like actions.
- the heart of activity models consist of activities, initial, final and decisions nodes and control flows.

- In 6 out of 20 activity models (i.e., 30% of the activity models) useless decision and merge nodes were found.
- The list of the not used UML concepts is large. Actions pins, activity parameters and many other UML concepts from the language unit activity (see Table 4.26) were never used.
- The action semantics introduced in UML 1.5 had no relevance for the models.
- Models mainly consisted of control flows rather than object flows.
- Activity models had a high number of possible paths.

Which Interaction Concepts are used?

Table 4.28 and Table 4.29 contain data about the usage of the interaction concepts.

Interpretation. The element distributions of the interaction concepts over the models have the same shape as the distributions previously. Therefore, the median is again chosen as the better average for interpreting the datasets. The top used UML concept was life line, followed by interaction fragment and interaction (see Table 4.28). In average 3 life lines were modeled in interaction models. In case of interaction fragments the most popular one is from kind loop, followed by alt, seq and last opt which occurs just in one model. All the other kinds of interaction fragments were not used. In general 8 out of 32 models (i.e., 25% of the interaction models) had interaction fragments. Two message endpoint constructs and one gate were found in one model. Without any relevance are the other interaction concept elements in Table 4.28. Never used were interaction states, interaction parameters, interaction occurrences, message labels, state lifelines, value lifelines and the other types of interaction fragments.

The most common message type was synchron, which appeared 452 times in 15 models (see Table 4.29). The asynchron message type were only modeled in two models. The reason for this is clear. In EA when a message is drawn between two life lines, this message is from type synchron by default. So it seems that the modelers did not care much about the message type and used the EA default settings. Rather relevant are the other types of messages in the interaction models. In average 16 sychnron messages between interaction concepts were modeled in models. Constructing a model with the average amount of life lines and sychnron messages would result in 16 communication paths between 3 life lines. If we compare the total amount of models with messages with the total amount of models with life lines we notice that there exist much more models with life lines than with messages. This has the simple reason that many life lines interact with concepts from other language units, mainly use cases and objects. Messages between communication partners from different language units are not counted here. The inter language relations with interaction participation is analyzed in Chapter 4.5.

An interesting fact, which appeared out of the results, is that just in 4 models the concept interaction element by itself appears. So for the sample models the common way of modeling

Table 4.28: Interaction - Concepts

Concept	In Models (32)	Total	Max	Min	AM	Med	SD
LifeLine	31	246	46	1	7,94	3	9,76
CombinedFragment	8	33	11	1	4,13	3	3,41
- (CombinedFragment) - Loop	(5)	(14)	7	1	2,80	2	2,23
- (CombinedFragment) - Alt	(4)	(11)	4	1	2,75	3	1,30
- (CombinedFragment) - Seq	(2)	(5)	4	1	2,50	2,5	1,50
- (CombinedFragment) - Opt	(1)	(3)	3	3	3,00	3	0,00
Interaction	4	39	20	1	9,75	9	7,98
MessageEndpoint	1	2	2	2	2,00	2	0,00
Gate	1	1	1	1	1,00	1	0,00
CombinedFragment - Break	0	0	0	0	0	0	0
CombinedFragment - Par	0	0	0	0	0	0	0
CombinedFragment - Critical	0	0	0	0	0	0	0
CombinedFragment - Neg	0	0	0	0	0	0	0
CombinedFragment - Assert	0	0	0	0	0	0	0
CombinedFragment - Strict	0	0	0	0	0	0	0
CombinedFragment - Ignore	0	0	0	0	0	0	0
CombinedFragment - Consider	0	0	0	0	0	0	0
InteractionState - Invariant	0	0	0	0	0	0	0
InteractionState - Continuation	0	0	0	0	0	0	0
InteractionParameter	0	0	0	0	0	0	0
InteractionOccurrence	0	0	0	0	0	0	0
MessageLabel	0	0	0	0	0	0	0
State Lifeline	0	0	0	0	0	0	0
Value Lifeline	0	0	0	0	0	0	0
Total	32	321					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Table 4.29: Interaction Relationships Distribution over the Models

Relationship	in Models (16)	Total	Max	Min	AM	Med	SD
Message Synchron	15	452	133	1	30,13	16	31,74
Message Asynchron	2	21	12	9	10,50	10,5	1,50
Message	1	20	20	20	20,00	20	0,00
Message Synchron - New	1	6	6	6	6,00	6	0,00
Total	16	499					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

interaction views of systems was in modeling the interaction concepts without the construct of an interaction element holding those concepts. The reason for this is probably the toolbox for interaction concepts, where users can pick the elements which should be drawn in a diagram, in EA. The toolbox does not list the interaction element. Furthermore, the averages of the interaction concept distributions indicate that the interaction models were kept small but this has to be enjoyed carefully. In the diagram statistics (see Chapter 4.3) the results show that in interaction diagrams many concepts from the language unit use case and object were found. Further, in numerous models with interaction concepts also use case concepts and object concepts were found as well (see Chapter 4.2). Hence, interaction concepts interacted in many cases with concepts from other language units and so the size of the interaction models is different then it can be derived from this statistics.

Summary of the Findings

- Only few different concept types from the language unit interaction could be found in the models.
- The preferred message type is synchron.
- The data indicates that the amount of messages per life line is rather high.
- The concept interaction by itself was not commonly used. Considering EA might be the reason why this is the case.
- The obtained data supports the strong cohesion of interaction concepts with use case and object concepts.

Which Object Concepts are used?

Table 4.30 and table 4.31 holds data about the usage of the object concepts.

Table 4.30: Object - Concepts

Concept	In Models (25)	Total	Max	Min	AM	Med	SD
Object	25	671	140	1	26,84	12	37,51
Total	25	671					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. Objects could be found in 25 models with a total amount of 671 (see Table 4.30). The standard deviation is higher than the arithmetic mean and the maximum indicates the existent of at least one outlier, therefore, the median measures the average best. In average 12 objects were modeled in models with objects.

Table 4.31: Object Relationships Distribution over the Models

Relationship	In Models (17)	Total	Max	Min	AM	Med	SD
Association (Link)	11	425	164	2	38,64	13	48,92
Message Synchron	5	27	11	1	5,40	3	4,63
Communication Message	5	580	457	6	116,00	51	171,57
Dependency	2	25	16	9	12,50	12,5	3,50
Message Synchron - New	2	4	2	2	2,00	2	0,00
Aggregation	1	16	16	16	16,00	16	0,00
Object Flow	1	69	69	69	69,00	69	0,00
Unclassified Connector	1	1	1	1	1,00	1	0,00
Total	17	1147					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

The most used relationship between objects was the object link, which is an instance of the association (see table 4.31). Also in this table the median can be seen as more reliable. Numerous interaction relationships were used between objects. Synchron messages, communication messages and synchron - new messages were modeled between objects. This is a strong indicator for the cohesion between objects and interactions. On the other hand only in one model the activity relationship object flow was used between objects. This also shows that the language units activity and objects hat not much in common in the 92 sample models.

In 25 models objects were modeled but only in 17 models relationships between objects could be found. Consequently in at least 8 models objects had relationships to other language unit concepts or nor relationships at all. Considering the strong cohesion between objects and interactions the first assumption seems to be more reliable.

Summary of the Findings

- Object links were the most used relationships between objects.
- In at least 8 models objects had relationships to other language unit concepts or none.

Which State Machine Concepts are used?

Table 4.32 lists all used state machine concepts. UML concepts which were not used in state machine models are depicted in Table 4.34. The used relationships between state machine concepts can be found in Table 4.33.

Interpretation. In every state machine model states were found (see Table 4.32). Also quite popular were initial states and final states, which occur in 4 models (i.e., in 66% of the state machine models). Still some relevance had the main concept of the language unit, the state ma-

Table 4.32: State Machine - Concepts

Concept	In Models (6)	Total	Max	Min	AM	Med	SD
State	6	72	24	4	12,00	10	7,44
InitialPseudoState	4	16	8	1	4,00	3,5	2,74
FinalState	4	18	9	1	4,50	4	2,96
Terminate	2	2	1	1	1,00	1	0,00
Choice	2	3	2	1	1,50	1,5	0,50
StateMachine	2	2	1	1	1,00	1	0,00
Junction	1	1	1	1	1,00	1	0,00
EntryPoint	1	1	1	1	1,00	1	0,00
Total	6	115					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Table 4.33: State Machine Relationships Distribution over the Models

Relationship	In Models (6)	Total	Max	Min	AM	Med	SD
Transition	6	127	48	6	21,17	18,5	15,31
Total	6	127					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

chine, which appeared in 2 models (i.e., in 33% of the state machine models). The few usage of the state machine element can be derived from the EA way of modeling, as state machine concepts can be created without state machine elements as container. This is different to the UML standard which says that every state machine construct should be only modeled in a state machine.

For the distributions of the UML concepts the median is again more reliable than the arithmetic mean as average number. In average 10 states were found in models with state machine concepts. Final states have the second highest average with 4, followed by initial states with 3,5 elements per model.

A significant number of state machine concepts were never used in the models (see Table 4.34). One of them are triggers and state behaviors which were never modeled by the modelers.

The only modeled relationship between state machine concepts was the transition, which could be found in all 6 state machine models (see Table 4.33). Almost double as many transitions were modeled as states. The average of transitions per model is 18,5, which is significant higher than for states. Creating a model according to the average values, for each state almost 2 transitions

Table 4.34: State Machine - Not Used Concepts

Concept	In Models
ProtocolStateMachine	0
State with Regions	0
StateBehavior	0
SubmachineState	0
State - isOrthogonal	0
Simple State	0
ExitPoint	0
ShallowHistoryNode	0
ShallowHistoryNode with Default Target	0
DeepHistoryNode	0
DeepHistoryNode with Default Target	0
SynchState	0
SignalTrigger	0
SignalTrigger with Specification	0
CallTrigger	0
CallTrigger with Specification	0
TimeTrigger	0
TimeTrigger with Specification	0
ChangeTrigger	0
ChangeTrigger with Specification	0
AnyTrigger	0
AnyTrigger with Specification	0

would be created.

We have to consider that not much data could be found about state machine concepts and therefore the findings about the used concepts have to be enjoyed carefully.

Summary of the Findings

- The majority of considered state machine concepts were never modeled (for example triggers and behaviors).
- States, transitions and initial and final states covered 233 of the 242 found state machine concepts over all models (i.e., over 96% of the total amount of found state machine concepts)

Limitation/Future Work. In the model samples state machine concepts were used rarely. A bigger set of models with state machines would be needed for more detailed and reliable researches in the usage of state machine concepts.

Which Component Concepts are used?

Data about the usage of component concepts are depicted in Table 4.35 and Table 4.36.

Table 4.35: Component Concepts

Concept	In Models (18)	Total	Max	Min	AM	Med	SD
Component	15	400	181	1	26,67	13	44,39
RequiredInterface	11	82	47	1	7,45	1	13,21
ProvidedInterface	10	109	56	1	10,90	2,5	17,22
Packaging Component	0	0	0	0	0	0	0
Total	18	591					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Table 4.36: Component relationships distribution over the models

Relationship	In Models (12)	Total	Max	Min	AM	Med	SD
Dependency	9	104	40	1	11,56	6	13,70
Assembly	5	51	19	1	10,20	10	6,37
Message Synchron	4	85	43	1	21,25	20,5	20,28
Message Asynchron	2	37	21	16	18,50	18,5	2,50
Association	2	23	14	9	11,50	11,5	2,50
Unclassified Connector	1	1	1	1	1,00	1	0,00
Generalization	1	1	1	1	1,00	1	0,00
Total	12	302					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. The main concept of the language unit, the component by itself, was used in 15 out of 18 models with elements from the language unit component (see Table 4.35). The 2 types of interfaces were the second most popular component concepts. Interesting is the fact that three models had no components but some of the other two concepts, provided interface or/and required interface. Therefore, in at least 3 models provided and required interfaces were used with no other component concepts together.

Without any importance for the evaluated models is the packaging component concept. A packaging component is similar to component with the difference that it can have inside other packages and elements as well. This construct seems to have no relevance for the functionality of any system described with the help of the analyzed models.

The most used relationship according to the *"In Models"* column between component concepts was dependency followed by assembly (see Table 4.36). The interaction relationships were modeled between component instances in some models. The synchron message was found in 4 and the asynchron message in 2 models between component instances. The other relationships are less relevant for the evaluation as they only occurred in 2 or 1 models.

Summary of the Findings

- Component, required and provided interface were the most used component concepts. In at least 3 models provided and required interfaces were used without any other component concepts together.
- Dependency was the most used relationship between component concepts.

Which Deployment Concepts are used?

The used deployment concepts and relationships between deployment concepts in the models can be found in Table 4.37 and Table 4.38.

Table 4.37: Deployment - Concepts

Concept	In Models (15)	Total	Max	Min	AM	Med	SD
NodeElement	11	44	10	1	4,00	3	3,28
Artifact	9	42	16	1	4,67	2	5,16
Device	7	15	4	1	2,14	2	1,12
ExecutionEnvironment	3	8	4	1	2,67	3	1,25
DeploymentSpecification	2	2	1	1	1,00	1	0,00
Total	15	111					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Table 4.38: Deployment Relationships Distribution over the Models

Relationship	In Models (7)	Total	Max	Min	AM	Med	SD
Association	5	21	7	3	4,20	3	1,60
Dependency	2	4	2	2	2,00	2	0,00
InformationFlow	1	2	2	2	2,00	2	0,00
Total	7	27					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. Top ranked in the usage is the standard node element, followed by the artifact (see Table 4.37). Still some relevance has the node type device, which was found in 7 deployment models (i.e., 50% of the deployment models). Less relevant but still used were the node type execution environment and the artifact type deployment specification. Therefore only special types of artifacts are important but the standard blank artifact type seems to be useless in the observed models.

The most used relationship was the association between deployment concepts (see Table 4.38) followed by dependency and information flow.

Summary of the Findings

- Nodes, artifact and devices in combination with associations as relationships were the most used deployment concepts.

Which Composite Concepts are used?

Table 4.39 gives an overview of the popularity of the individual composite concepts. No relationships between composite concepts were measured in all composite models. Subsequently no table for the used relationships exist. The UML relationship type "Connector" would belong to this language unit.

Table 4.39: Composite - Concepts

Concept	In Models (9)	Total	Max	Min	AM	Med	SD
Part	8	26	13	1	3,25	1	4,18
Port	2	4	3	1	2,00	2	1,00
Collaboration Element	0	0	0	0	0,00	0	0,00
Collaboration Use	0	0	0	0	0,00	0	0,00
Total	9	30					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. In 8 models with elements from the language unit composite (i.e., over 88% of the models with elements from the language unit composite), part elements were modeled. Far behind is the port concept, which just occurred in 2 models (i.e., 22,22% of the models with composite concepts). Never used was the collaboration element and the collaboration use element. In average (median) 1 part were modeled in models with part concepts. The modelers used parts very rarely.

Summary of the Findings

- No relationships exist between composite concepts. The UML concept “Connector“ could be modeled between composite concepts to express relationships. It seems that classifiers, which were modeled with parts and ports, were simple structured without relationships between parts and ports.
- Only parts and ports were modeled
- In average only 1 part was found in models with part concepts

Future Work/Limitation. More models with composite concepts would be needed for more meaningful studies about this language unit.

Which Auxiliary Construct Concepts are used?

Last the auxiliary construct use is observed in Table 4.40. No relationships exist between auxiliary construct concepts.

Table 4.40: Auxiliary Construct - Concepts

Concept	In Models (67)	Total	Max	Min	AM	Med	SD
Comment	67	902	97	1	13,46	5	19,50
Constraint	2	16	9	7	8,00	8	1,00
Information Item	0	0	0	0	0,00	0	0,00
Total	67	918					

Max = Maximum, Min = Minimum, AM = Arithmetic Mean
Med = Median, SD = Standard Deviation

Interpretation. In 67 models 902 comments could be found (see Table 4.40), only in 2 models constraints. The other auxiliary construct concepts were not important for the 92 observed models. In average (median) 5 notes were modeled in models with notes and 8 constraints in models with constraints.

The numerous models with notes and the median of the notes distribution indicate a high popularity of comments through the evaluated models. No other single UML concept could be found in that many models.

Summary of the Findings

- Comments were quite popular within the 92 sample models.

Overall Limitations/Future Work - UML Concepts

The specification of the latest UML 2.x version by OMG has much more concepts than considered here. In this work a limit was set by concentrating on the most important UML concepts

to keep the study within an acceptable border. Further studies could go more into detail by observing all possible concepts and features within the language units. For example the use of multiplicities for attributes and associations might be interesting for future work. Further, in this work only the relationships between UML concepts were analyzed but not between which UML concept exactly those relationships exist. The reason was that this work mainly concentrated on the popularity of the UML concepts by itself. Future work could do deeper and more detailed research in this case.

What was the common Comment Length?

Table 4.41 and Table 4.42 show statistics about the length of UML comments in the 92 sample models. In Table 4.41 comments were categorized into 3 groups of sizes - short, medium and large comments. Comments with more than 30 characters were categorized as short, comments which had between 30 and 100 characters were categorized as medium sized and all comments which had more than 100 characters were seen as large comments. To get more knowledge about the data sets of comments further statistics were calculated in Table 4.42. The maximum, the minimum, the arithmetic mean as well as the median over all comment lengths were calculated.

Table 4.41: Comment Length

Comment Size	Total
Short Comments (< 30 Characters)	71
Medium Sized Comments (30 to 100 Characters)	315
Large Comments (> 100 Characters)	505
Total Comments with Content	891

Table 4.42: Comment Length - Statistical Analysis

Biggest Comment	1669 Characters
Smallest Comment	4 Characters
Average Size (Arithmetic Mean)	186 Characters
Average Size (Median)	116 Characters

Interpretation. 71 comments in the 92 models had less than 30 characters and were hence counted as short comments. 315 comments had between 30 to 100 characters and are handled as so called medium sized comments in this work. The most comments were found in the biggest group. 505 comments had more than 100 characters. Comparing Table 4.41 with Table 4.40 from the previously research question it can be noticed that 11 out of 902 comments had no content at all.

We see the averages are 186 and 116 characters and the biggest comment has 1669 characters. This points out that comments where mostly from bigger sizes.

Summary of the Findings.

- Comments had big sizes, in average between 100 to 200 characters.

What was the Content of the Comments?

Another interesting data can be taken out from Table 4.43, where the content of the comments were analyzed. `Context` declares the type of comment contents and `Total` in how many comments this type of content could be found.

Table 4.43: Comment Context

Context	Total
Natural Language	837
HTML Tags (<html>,<b>, ,<body>,<p>,<font>, , , ,...)	28
Constraint Parameters (=, <, >, !=, <>, ==)	15
Calculations(*,+,-,/)	9
OCL Keywords(context, inv:,self., select, collect,implies)	1
Pseudo Code (IF, ELSE, AND, OR)	1
Total Comments	891

Interpretation. Mostly comments consisted of text written in natural language without any special constructs. HTML tags were the second most used comment type, but far behind text comments. Several comments had some calculations or constrain parameters inside. Only one comment was written in OCL language and only one comment was written in pseudo code.

Summary of the Findings

- Comment contents were mostly written in natural language.
- Considering the comments lengths, those text consisted therefore usually of several words.

Are the Language Concepts introduced with UML 2.x used?

To answer this question all considered UML concepts in this work were categorized to the UML version they were introduced. Table 4.44 lists the UML concepts according to the UML version they first appeared in. The column `UML Concept` shows the element type. `In Models` declare the amount of models this concept was found. The UML version the concept was introduced earliest stands in `UML Version`. Only elements were taken into account, connectors were skipped.

Table 4.44: List of UML Concepts Categorized by the UML Version they were Introduced

UML Concept	In Models	UML Version
Class	65	<= 1.4
ClassAttribute	51	<= 1.4
ActorElement	44	<= 1.4
ClassOperation	43	<= 1.4
Boundary	37	<= 1.4
UseCaseElement	35	<= 1.4
ClassOperationParameter	32	<= 1.4
LifeLine	31	<= 1.4
Interface	28	<= 1.4
Object	25	<= 1.4
Activity	18	<= 1.4
ActivityInitialNode	17	<= 1.4
ActivityFinalNode	15	<= 1.4
ActivityDecisionNode	15	<= 1.4
Component	15	<= 1.4
Abstract Class	14	<= 1.4
Action	11	<= 1.4
RequiredInterface	11	<= 1.4
Node	11	<= 1.4
Enumeration	10	<= 1.4
Atomic-Action	10	<= 1.4
ProvidedInterface	10	<= 1.4
Artifact	9	<= 1.4
CombinedFragment	8	2.x
PartElement	8	2.x
ActivityDecisionMergeNode	7	<= 1.4
ForkNode	7	<= 1.4
Device	7	2.x
State	6	<= 1.4
JoinNode	6	<= 1.4
Unused ActivityDecisionMergeNode	6	<= 1.4
FlowFinalNode	6	2.x
CombinedFragment - Loop	5	2.x
ActivityPartition	5	2.x
ActivityMergeNode	5	<= 1.4

Continued on Next Page

Table 4.44 – Continued From Previous Page

UML Concept	In Models	UML Version
Interaction	4	<= 1.4
CombinedFragment - Alt	4	2.x
InitialPseudoState	4	<= 1.4
FinalState	4	<= 1.4
AssociationClass	3	<= 1.4
CallOperationAction	3	1.5
JoinForkNode	3	<= 1.4
DataStore	3	2.x
ExecutionEnvironment	3	2.x
CombinedFragment - Seq	2	2.x
Terminate	2	2.x
Choice	2	<= 1.4
StateMachine	2	<= 1.4
ObjectNode	2	2.x
InterruptibleActivityRegion	2	2.x
Port	2	2.x
DeploymentSpecification	2	2.x
DataType	1	<= 1.4
MessageEndpoint	1	2.x
Gate	1	2.x
CombinedFragment - Opt	1	2.x
Junction	1	<= 1.4
EntryPoint	1	<= 1.4
LoopNode	1	2.x
ExpansionRegion	1	2.x
ExpansionRegion in iterative mode	1	2.x
Unused JoinForkNode	1	<= 1.4
WriteVariableAction	1	1.5
Active Class	0	<= 1.4
Parameterized Class	0	<= 1.4
Signal	0	<= 1.4
N-ary Association	0	<= 1.4
CombinedFragment - Break	0	2.x
CombinedFragment - Par	0	2.x
CombinedFragment - Critical	0	2.x
CombinedFragment - Neg	0	2.x
CombinedFragment - Assert	0	2.x

Continued on Next Page

Table 4.44 – Continued From Previous Page

UML Concept	In Models	UML Version
CombinedFragment - Strict	0	2.x
CombinedFragment - Ignore	0	2.x
CombinedFragment - Consider	0	2.x
InteractionState - Invariant	0	<= 1.4
InteractionState - Continuation	0	2.x
InteractionParameter	0	2.x
InteractionUse	0	2.x
State LifeLine	0	2.x
Value LifeLine	0	2.x
ProtocolStateMachine	0	2.x
State with Regions	0	<= 1.4
StateBehaviorElement	0	<= 1.4
StateBehavior - Entry	0	<= 1.4
StateBehavior - Do	0	<= 1.4
StateBehavior - Exit	0	<= 1.4
StateBehavior - Any	0	<= 1.4
SubmachineState	0	<= 1.4
State - isOrthogonal	0	<= 1.4
Simple State	0	<= 1.4
ExitPoint	0	<= 1.4
ShallowHistoryNode	0	<= 1.4
DeepHistoryNode	0	<= 1.4
SynchState	0	<= 1.4
SignalTrigger	0	<= 1.4
CallTrigger	0	<= 1.4
TimeTrigger	0	<= 1.4
ChangeTrigger	0	<= 1.4
AnyTrigger	0	<= 1.4
CallBehaviorAction	0	2.x
AcceptCallAction	0	2.x
AcceptEventAction	0	2.x
AcceptEventTimerAction	0	2.x
AddStructuralFeatureValueAction	0	2.x
AddVariableValueAction	0	1.5
BroadcastSignalAction	0	1.5
ClearAssociationAction	0	1.5
ClearStructuralFeatureAction	0	2.x

Continued on Next Page

Table 4.44 – Continued From Previous Page

UML Concept	In Models	UML Version
ClearVariableAction	0	1.5
CreateLinkAction	0	1.5
CreateLinkObjectAction	0	1.5
CreateObjectAction	0	1.5
DestroyLinkAction	0	1.5
DestroyObjectAction	0	1.5
HyperlinkAction	0	2.x
RaiseExceptionAction	0	2.x
ReadExtentAction	0	1.5
ReadIsClassifiedObjectAction	0	1.5
ReadLinkAction	0	1.5
ReadLinkObjectEndAction	0	1.5
ReadLinkObjectEndQualifierAction	0	1.5
ReadSelfAction	0	1.5
ReadStructuralFeatureAction	0	1.5
ReadVariableAction	0	1.5
ReclassifyObjectAction	0	1.5
RemoveStructuralFeatureValueAction	0	2.x
RemoveVariableValueAction	0	1.5
ReplyAction	0	2.x
SendObjectAction	0	2.x
SendSignalAction	0	1.5
StartClassifierBehaviorAction	0	2.x
TestIdentifyAction	0	2.x
ValueSpecificationAction	0	2.x
WriteLinkAction	0	1.5
WriteStructuralFeatureAction	0	2.x
SendSignalAction	0	1.5
ActionPin	0	1.5
ActivityParameter	0	2.x
CentralBufferNode	0	2.x
ConditionalNode	0	2.x
ExceptionHandler	0	2.x
ExpansionNode	0	2.x
ExpansionRegion	0	2.x
SequenceNode	0	2.x
StructuredActivityNode	0	2.x

Continued on Next Page

Table 4.44 – Continued From Previous Page

UML Concept	In Models	UML Version
PackagingComponent	0	2.x
CollaborationElement	0	2.x
CollaborationUse	0	2.x

Interpretation. We can see that the most used UML concepts among the models already exist at UML version 1.4 or earlier. Required and provided interfaces were the most used new UML concepts. All other newly introduced UML concepts in version 2.x were found in less than 10 out of 92 evaluated models. The new action semantics which were established in the UML version 1.5 had no relevance among the 92 models. Only 2 new action concepts could be found which appeared in 3 and in 1 model.

Summary of the Findings

- Used UML concepts mostly exist already in UML version 1.4 or earlier
- All new concepts from UML version 2.x had not a big acceptance over the models.
- The introduced actions semantics from version 1.5 were rarely used (not more than in 3 models).

4.5 Relationships between UML Language Units

In this chapter all modeled relationships between model elements from different UML language units were observed. All possible relationships which were considered in this work are listed in table A.12 and table A.13 of the appendix A. Relationships to or from concepts from the language unit auxiliary constructs were not considered.

Between which Language Units Relationships exist?

Table 4.45 shows between which language units relationships exist in how many models. The rows and columns stand for the language units and the cells contain the number of models having modeled relationships between the language units. Further, Table 4.46 supports the significance of the amount of models with relationships between two distinct language units. This is done by calculating the ratio between the amount of models having relationships between the language unit pairs (see Table 4.45) and the amount of models containing the considered language units (see Chapter 4.2, Table 4.14). Each cell in Table 4.46 is the result of *total models with relationships between language unit A and language unit B* divided by *total modals which contain*

language unit A and language unit B. A result of 1 means that in every model (i.e., in 100% of the models) where language unit A and B appeared also relationships were explicitly modeled between A and B. On the other hand 0 means that in all models containing A and B no single relationship between A and B were modeled.

Table 4.45: Number of Models which modeled Relationships between Language Units

Language Unit	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*64*	4	3	**7**	2	5	0	1	1
UC	4	*35*	**21**	**14**	4	3	0	0	0
Int	3	**21**	*16*	**6**	0	0	0	0	0
Obj	**7**	**14**	**6**	*17*	4	1	1	0	0
Act	2	4	0	4	*18*	0	0	0	1
Comp	5	3	0	1	0	*13*	2	**7**	0
Deploy	0	0	0	1	0	2	*7*	0	0
Composite	1	0	0	0	0	**7**	0	*0*	0
SM	1	0	0	0	1	0	0	0	*6*

Cl = Class, UC = Use Case, Int = Interaction, Obj = Object, Act = Activity
Comp = Component, Deploy = Deployment, SM = State Machine

Table 4.46: Explicit Strengths of the Language Unit Relationships

Language Unit	Cl	UC	Int	Obj	Act	Comp	Deploy	Composite	SM
Cl	*0,97*	**0,12**	**0,12**	**0,33**	**0,15**	**0,36**	0,00	**0,13**	**0,20**
UC	**0,12**	*0,70*	**0,84**	**0,67**	**0,33**	**0,23**	0,00	0,00	0,00
Int	**0,12**	**0,84**	*0,50*	**0,43**	0,00	0,00	0,00	0,00	0,00
Obj	**0,33**	**0,67**	**0,43**	*0,68*	**0,36**	**0,20**	**0,13**	0,00	0,00
Act	**0,15**	**0,33**	0,00	**0,36**	*0,90*	0,00	0,00	0,00	**0,33**
Comp	**0,36**	**0,23**	0,00	**0,20**	0,00	*0,72*	**0,25**	**0,88**	0,00
Deploy	0,00	0,00	0,00	**0,13**	0,00	**0,25**	*0,47*	0,00	0,00
Composite	**0,13**	0,00	0,00	0,00	0,00	**0,88**	0,00	*0,00*	0,00
SM	**0,20**	0,00	0,00	0,00	**0,33**	0,00	0,00	0,00	*1,00*

Cl = Class, UC = Use Case, Int = Interaction, Obj = Object, Act = Activity
Comp = Component, Deploy = Deployment, SM = State Machine

Interpretation. Relations between interaction and use case language units were modeled at most. They could be found in 21 models (see Table 4.45). The second most modeled relation were between use case and object concepts, which appeared in 14 models. In 7 models relationships between class and object and relationships between composite and component were found. Following, the language unit pairs which had the 5 most models with relationships between them are:

- Use case and interaction - in 21 models
- Use case and object - in 14 models
- Class and object - in 7 models
- Component and composite - in 7 models
- Interaction and object - in 6 models

Table 4.45 also shows that there are numerous language unit combinations which had no relationships. Deployment, composite and state machine concepts had only with 2 other language units modeled relationships. Interaction concepts only had relationships with class, use case and objects.

The language unit combination interaction/use case had a strength of 0,84 in Table 4.46. This means in 84% of the models with interaction and use case concepts also relations were modeled between these language units. Another big cohesion also exists between Component and Composite concepts, which had a strength of 0,88. So in 88% of the models with these language units also modeled relations between them exists. As we go further in Table 4.46 the language unit pair use case/object had a strength of 0,67. In 67% of the models with this language unit combination also Connectors between them were modeled. Still relevant is the language unit combination of interaction/object which had a value of 0,43. The combination class/object which is one of the 5 most used combinations had only a value of 0,33. The other language unit pairs had a little amount of modeled relationships between them in the models and low ratios between relationships and model occurrences.

Interaction/use case and component/composite have the strongest cohesion within the evaluated models. Further the results mirror the language unit combinations of Chapter 4.2. Use case, interaction and object are also there strongly modeled together in the models. The results in this section show that they also have modeled relationships to each other and not only modeled in models side by side without Connectors.

To sum it up the most language unit pairs had little modeled relationships to each other. Only few language unit combinations were strongly related to each other. This result is different compared to the results about the appearance of the language unit combinations in models, where many language units appeared together in many models. There exist probably several reasons why language units which appear in many models together have only a few modeled relations to each other. One could be that UML does not provide adequate language tools for expressing relations between these language concepts. Another one that EA simply does not support the required relationships between elements. It is also possible that the modelers do not know constructs for modeling relations between some language units. Finally it could be that there was no need for modeling relations between these language units because they should give separated views to the abstracted system. For example the language units class and interaction. According to the data from Table 4.14 of Chapter 4.2 and Table 4.46 many models exist with this language

unit combinations but just a few really have modeled relations between them. It seems that class constructs does not take a big part in interaction models and that these two language units model different aspects of a system. A bit different is the situation between interaction and object concepts. In almost half of the models with interaction and object participation modeled relations between them exists. This covers also the interpretation and use of interactions with objects in the UML literature, where objects can be found as instances of classes in several sequence diagrams as life lines. It is important to clarify that a life line in EA is an own element type. So in a sequence diagram a life line has to be distinguished from an object or an actor. In EA elements like actors, classes or objects can be dragged into a sequence diagram and would be displayed as life lines but in the database they are still saved as an element from the type actor, object or class. For the following detailed analyzes of the relationships between objects, use cases and interactions the EA element types were analyzed. A life line was seen as an interaction concept, an object was seen as an object concept and an actor was seen as an use case concept.

Summary of the Findings

- Few relationships between concepts of different language units exist in the 92 models.
- The strongest cohesion exist between the language unit combinations use case/interaction and component/composite.
- As Tables 4.14, 4.15, 4.18 and 4.19 in Chapters 4.2 and 4.3 already showed, interaction, objects and use case concepts were often modeled together. The results above showed now that these language units also had modeled relations to each other in the most models where they appeared together.

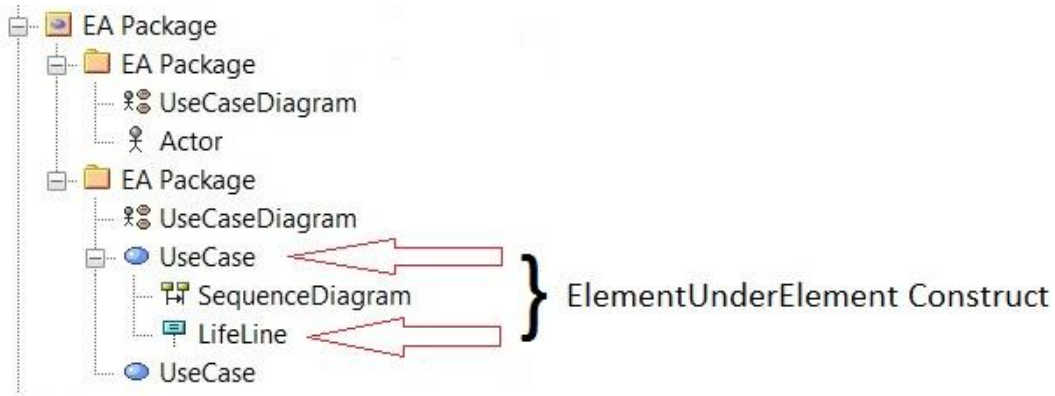
Limitations/Future Work. In this work only relationships listed in Table A.12 and Table A.13 are considered. All possible further relationships, which might be not considered in this work between UML language units, can be part of future work.

Which Relationships are used for the Top Related Language Units?

In this chapter from the top 5 language unit pairs, which were figured out in the answer of the research question above, the used relationships will be listed and interpreted. The relationships for each language unit pair are described with three tables. One table lists all used relationships between the language units with the total amount and the occurrence in models. This table can be seen as an overview, the other tables present more details about the data in the first overview table. One of the other tables describes in more detail the EA specific “ElementUnderElement“ relationships, if some exist between the language units. An “ElementUnderElement“ relationship means that under an element in the EA project browser another element is modeled. In this work this was seen as a modeled relationships between those two elements. In Figure 4.14 illustrates an “ElementUnderElement“ construct between a use case and a life line. The use case is the parent element and the life line the child element. Therefore in the “More Detail: ElementUnderElement“ table the parent and child element combination with the total amount and occurrence in models is presented. The last table lists the element types of the language units

between the found relationships. No difference between source and target was made, as it was not seen as relevant for this evaluation.

Figure 4.14: EA Specific “ElementUnderElemen“ Construct



Interaction - UseCase Pair

Table 4.47 lists the used relationships between interaction and use case concepts. Table 4.48 gives a closer look between which concepts of the language units use case and interaction “ElementUnderElement“ relationships were modeled. Finally Table 4.49 lists the exact concepts which were linked with the other relationships.

Table 4.47: Used Relationships

Relationship	Total	In Models
Message	138	18
ElementUnderElement	102	16
LifeLine2ClassifierInstance	5	3
Total	245	21

Table 4.48: More Detail: ElementUnderElement

Parent Element	Child Element	Total	In Models
UseCase	LifeLine	42	15
UseCase	Interaction	38	3
Interaction	Actor	22	3
Total		102	16

Table 4.49: More Detail: Other Relationships

UML Concept	UML Concept	Relation Type	Total	In Models
Actor	LifeLine	Message	138	18
LifeLine	Actor	LifeLine2ClassifierInstance	5	3
Total			143	19

Interpretation. In 21 out of 25 models with elements from the language units interaction and use case also relationships were modeled. The most popular way of expressing a relation between interaction and use case concepts is with the UML message concept (see Table 4.47). In 18 out of 21 models with relations between interaction and use case concepts this type of linking was used. Closely followed by modeling one element under another element ("ElementUnderElement") in the EA Project Browser with 16 out of 21 models. Far behind is the modeling construct of instantiating a life line object from a use case concept ("LifeLine2ClassifierInstance"), which only appears in 3 models.

As we can see top ranked in the "ElementUnderElement" constructs are life lines under use cases. In 15 models (i.e., 71% of models with use case and interaction relationships) this way of connection was modeled with a total of 42 times. Far behind are interactions under use cases and actors under interactions.

Message Connectors were found between life line instances of actors and life lines in 18 models. The relation type "LifeLine2ClassifierInstance" was used in 3 models between life lines and actors. The relationship "LifeLine2ClassifierInstance" in this context means that a life line is initiated from an actor.

The results lead to the conclusion that use cases were described by sequence diagrams, as life lines can only be modeled in sequence diagrams in EA, and so by whole interaction structures. Furthermore, in interaction models, mostly in sequence diagrams if we consider the diagram evaluations from Chapter 4.3, actors were modeled as communication partners together with life lines.

Summary of the Findings

- use cases were explained in detail by sequence diagrams.
- in sequence diagrams life lines and actors were communication partners.

UseCase - Object Pair

Table 4.50: Used Relationships

Relationship	Total	In Models
Association	74	11
ElementUnderElement	210	5
CommunicationMessage	103	4
Object2Classifier	14	3
Message	5	2
Aggregation	3	1
Total	409	14

Table 4.51: More Detail: ElementUnderElement

Parent Element	Child Element	Total	In Models
UseCase	Object	210	5
Total		210	5

Table 4.52: More Detail: Other Relationships

UML Concept	UML Concept	Relation Type	Total	In Models
Actor	Object	Association	66	10
Actor	Object	CommunicationMessage	103	4
Actor	Object	Message	5	2
Object	UseCase	Object2Classifier	4	2
Object	Actor	Object2Classifier	10	2
Object	Actor	Aggregation	3	1
Object	UseCase	Association	8	2
Total			199	14

Interpretation. The most popular relationship between the language units use case and object is association (see Table 4.31). In 11 out of 14 models with relationships between use case and object an association link was used to express this construct. An association is not the UML standard way of modeling Connectors between object and use case concepts. There exist an explanation why associations where used in this context. In EA linking two communication partners automatic generates an simple association between them. If you want to explicit model an communication message between two interaction partners in an communication diagram (for example between an object and an actor) you have to add extra the message you want to send or receive between the partners to the modeled association. The message counts as communication message and the relationship between the communication partners as association in EA. As the results show in many cases modelers simple linked two communication partners without extra adding an communication message to the link.

The second most used way of linking these two language units were by modeling one concept under another in the EA project browser. Table 4.51 gives more details about this. In only 3 models an object represented an instance of an use case classifier and in 2 models messages could be found between use case and object concepts. One model had even an aggregation relation between those two language units.

Only objects were modeled under use cases but not the other way around as we can see on Table 4.51.

In Table 4.52 we can see that associations, communication messages and sequence messages were usually found between actor and object elements.

If we compare these results with the results from the diagram contents of Chapter 4.3, Table 4.20 and the results of the relationships between use case and interactions a conclusion can be made. In sequence as well as in communication diagrams objects and use case concepts could be found. The evaluations here show that actor and objects were in communication to each other with associations and messages. Further, objects were found under use cases. Under use cases also life lines were found and therefore sequence diagrams. This leads to the conclusion that use cases were described by interaction constructs in form of communication and sequence diagrams with mostly objects, life lines and life line instances of actors as communicators.

Summary of the Findings

- Use cases were also described by communication diagrams.
- In communication diagrams actors and objects were communication partners.

Object - Class Pair

For the object/class pair only 2 tables describe the used relationships (Table 4.53 and Table 4.54). There were no "ElementUnderElement" constructs found between these 2 language units.

Table 4.53: Used Relationships

Relationship	Total	In Models
Object2Classifier	57	7

Interpretation. As Table 4.53 lists, for all modeled relations between object and class concepts only one type was used, "Object2Classifier". For a more detailed view Table 4.54 shows between which exact UML concepts from both language units the relationship was used. In the

Table 4.54: More Detail: Other Relationships

UML Concept	UML Concept	Relation Type	Total	In Models
Object	Class	Object2Classifier	50	6
Object	Interface	Object2Classifier	7	2
Total			57	7

most cases an object represented an instance of a class. In only 2 models it was different and a object was an instance of an interface.

Summary of the Findings

- "Object2Classifer" was the only modeled relationship between objects and class concepts.

Component - Composite Pair

Table 4.55: Used Relationships

Relationship	Total	In Models
ElementUnderElement	28	7
Delegate	6	6
Dependency	2	1
Total	36	7

Table 4.56: More Detail: ElementUnderElement

Parent Element	Child Element	Total	In Models
Component	Part	25	7
Component	Port	3	1
Total		28	7

Interpretation. On top we see the "ElementUnderElement" construct in Table 4.55. In every model with a modeled relationship between component and composite elements this type of relation was used. In 6 out of 7 models the delegate connector could be found and in only one model 2 dependency connectors were used. To see between which concepts the different relation types were modeled we take a look into Table 4.56 and Table 4.57. There we see a component element was always the parent element within an "ElementUnderElement" relation.

Table 4.57: More Detail: Other Relationships

UML Concept	UML Concept	Relation Type	Total	In Models
ProvidedInterface	Part	Delegate	6	6
Component	Port	Dependency	2	1
Total			8	7

So in the project browser of EA composite concepts were modeled under component elements. In 6 out of 6 models with a delegate connection between component and composite the relation was between a provided interface and a part element. Only one model had a delegate connection between a component and a port element.

Summary of the Findings

- Relationships were mostly modeled between parts and components.

Object - Interaction Pair

Table 4.58: Used Relationships

Relationship	Total	In Models
Message	17	4
ElementUnderElement	99	2
Total	116	6

Table 4.59: More Detail: ElementUnderElement

Parent Element	Child Element	Total	In Models
Interaction	Object	99	2

Table 4.60: More Detail: Other Relationships

UML Concept	UML Concept	Relation Type	Total	In Models
LifeLine	Object	Message	17	4

Interpretation. In 4 models message Connectors between object and interaction concepts could

be found and in 2 models an element under element construct was used (Table 4.58). If we observe the data from Table 4.59 and Table 4.60 it shows us a clear picture how the language unit object and interaction were mainly used together. The messages were only between object and life lines and according to Table 4.59 in 2 models objects were modeled under interaction elements. Objects were less in relation with life lines than with actors. It seems objects in interaction diagrams are more in communication with use case concepts, specially actors.

Only in a few models relationships between objects and interactions could be found. Use case concepts were the more common interacting partner for objects in the models. Objects were mostly modeled within interaction diagrams (see Chapter 4.3, Table 4.20) but acted more with actors than with life lines.

Summary of the Findings

- Objects had mainly relationships with actors in interaction diagrams rather than with life lines.

Overall Findings of Relationships between UML Language Units

The results of the evaluations showed that the language units use case, interaction, object and class had a strong correlation together. Following the findings supporting the strengths of the combination of these four language units are listed:

- The appearances of the language unit pairs as found in Table 4.14 in Chapter 4.2 are: class/use case (in 33 models), class/interaction (in 26 models), use case/interaction (in 25 models), class/object (in 21 models), use case/object (in 21 models) and object/interaction (in 14 models).
- The relevant diagram contents listed in Table 4.20 in Chapter 4.3 are: In 24 models actors could be found in sequence diagrams, in 6 models objects were found in sequence diagrams, in 6 models actors could be found in communication diagrams, in 7 models objects were found in communication diagrams.
- In 15 models life lines were modeled under use cases
- In 18 models actors were linked to life lines with messages
- In 11 models actors and objects were linked via associations and in 4 models via communication messages.
- In 7 models classes were instantiated by objects.

The results showed that the language units use case and interaction were strongly related. In EA life lines can only be modeled in sequence diagrams and in no other diagram type. This means that in 25 models with concepts from the language units use case and interaction in 15 models use cases were described by sequence diagrams because in 15 models life lines were modeled

under use cases in the EA project browser as “ElementUnderElement“ constructs. Further in 6 models objects were found in sequence diagrams and in 7 models objects were found in communication diagrams. In the other UML diagrams objects were rarely found (see Table 4.20, Chapter 4.3). In 7 models out of 21 models with class und object elements (i.e., 33% of models with class und object elements) objects represented instance of classes.

Summarized use case and interaction language unit concepts were strongly related and objects could be found in many relations to the language units use case, interaction and class, where for classes objects were mainly used as instances of classes.

4.6 Stereotype Usage

The last section of this chapter explores for which UML concepts stereotypes were used and how often in the 92 sample models. The aim of this specific research is to show which UML constructs are usually more extended and configured by modelers and which are not.

In EA exist multiple ways of defining stereotypes and assign them to model elements. As a consequence stereotypes are mapped in different ways in the EA model repository. Further, stereotypes are not only used to extend UML elements. In EA stereotypes also declare an model element as concept from BPMN, ArchiMate, SoaML or other from EA supported modeling languages. This means that even as UML concept declared model elements can be from another language than UML, only by using such stereotypes. Those stereotypes have always a full quality name (FQName) declaring the modeling language. A FQName is an attribute value pair in the description of the stereotypes in the database. If a user defines an own stereotype there is also the opportunity to set a FQName in some cases. Then it could be even an UML profile. Anyway there exists no clear way to distinguish if a model element with a stereotype which has a FQName is an UML element or not. As a consequence in this work all model elements which have a FQName are declared as `EA Profile` and not considered in the evaluations of the UML (see also page 25 in this work). If the stereotype has no FQName it is not an `EA Profile` and therefore considered as `UML Profile` in this work. Following only model elements which are declared as `UML Profile` are considered. So these model elements have stereotypes without a FQName.

Which UML Concepts had Stereotypes?

Table 4.61 lists all UML concepts which were extended by stereotypes, which were considered as `UML Profile`. The values in the `Concept` column represent the UML element type, which are sorted by their `Language Unit`. In the `Total` column the total amount of elements of the UML concept in the models can be found. In the `Total (St)` column the amount of model elements from the UML concept which are extended by stereotypes is depicted. The `in Models` column shows in how many models the UML element type occurs.

The numbers in the `in Models (St)` column indicate in how many models model elements from the UML concept are extended by stereotypes.

Table 4.61: UML Concepts with Stereotypes

Language Unit	Concept	Total	Total (St)	In Models	In Models (St)
Class	Class	2124	555 (26%)	65	27 (42%)
Class	ClassAttribute	7812	3839 (49%)	51	12 (24%)
Class	ClassOperation	6851	1844 (27%)	43	10 (23%)
Class	Abstract Class	48	14 (29%)	14	5 (36%)
Class	ClassOperationParameter	4262	11 (0%)	32	1 (3%)
Class	Interface	180	1 (1%)	28	1 (4%)
UseCase	UseCaseElement	489	53 (11%)	35	8 (23%)
UseCase	ActorElement	257	25 (10%)	44	2 (5%)
Interaction	LifeLine	246	101 (41%)	31	9 (29%)
Object	Object	671	555 (83%)	25	18 (72%)
Activity	ActivityElement	392	7 (2%)	18	3 (17%)
Activity	ActivityPartition	18	3 (17%)	5	1 (20%)
Activity	Action	85	2 (2%)	11	1 (9%)
Component	Component	400	78 (20%)	15	4 (27%)
Component	ProvidedInterface	109	32 (29%)	10	2 (20%)
Component	RequiredInterface	82	18 (22%)	11	2 (18%)
Deployment	Document Artifact	42	27 (64%)	9	4 (44%)
Deployment	NodeElement	44	3 (7%)	11	2 (18%)
Deployment	ExecutionEnvironment	8	1 (13%)	3	1 (33%)
Composite	Port	4	2 (50%)	2	1 (50%)
Unclassified	Dependency	489	178 (36%)	36	10 (28%)
Unclassified	Association	3047	171 (6%)	67	9 (13%)
Unclassified	Assembly	51	24 (47%)	5	2 (40%)
Unclassified	Message	921	88 (10%)	34	2 (6%)
Unclassified	Aggregation	729	546 (75%)	23	1 (4%)
Unclassified	Use	158	2 (1%)	23	1 (4%)
Total		29 519	8 166	92	56

St = Stereotype, Total (St) = Total Model Elements of a UML Concept with Stereotypes
In Models (St) = Total Models with Model Elements of a UML Concept with Stereotypes

Interpretation. 555 out of 2124 classes were extended by stereotypes (i.e., 1/4 of all classes) in 27 out of 65 models with classes (see Table 4.61. 555 objects in 18 models had stereotypes (i.e., over 80 % of modeled objects). Many class operations and attributes were extended by stereotypes. Almost half of all class attributes, these are 3839 attributes, were extended by stereotypes in 12 models (i.e., 1/5 of models with class attributes). 1844 class operations in 10 models (i.e., 1/4 of the models with class operations)) had stereotypes. Also for life lines the percentage of

used stereotypes is high. From 246 life lines found in 31 models 101 were extended by stereotypes in 9 models. Not many artifacts exist in the models but more than half of them are extended by stereotypes in 4 out of 9 models with this UML concept. The UML relationship with the most stereotypes in models was dependency. Associations were mostly used without stereotypes only 171 of 3047 associations were extended by stereotypes in 9 models. Nearly half of all assembly connectors had stereotypes. The amount of aggregations with stereotypes is high. 546 out of 729 aggregations were extended by stereotypes but they were all found in one model. In case of language units we can notice that no single UML concept from the language unit state machine were extended by stereotypes.

To sum it up the 6 most modified UML concepts by stereotypes, which appeared in at least 9 models, are:

- Object (i.e., in 72% of models with objects)
- Class (i.e., in 42% of models with classes)
- Life lines (i.e., in 29% of models with life lines)
- Dependency (i.e., in 28% of models with dependencies)
- Attributes (i.e., in 24% of models with attributes)
- Operations (i.e., in 23% of models with operations)

What are the most popular Stereotypes?

By observing the used stereotypes of the model elements in the models, a list with all the names of the used stereotypes could be generated. In Table 4.62 we see the names of the 5 most used stereotypes. `EA Type` stands for the model element(s) the stereotype was used for. `Stereotype` lists the name of the stereotype. `Total` the total amount this stereotype was used and `in Models` the number of models where this stereotype could be found on the model element.

Table 4.62: Top 5 used Stereotypes

EA Type	Stereotype	Total	In Models
Object, Life Line, Class	Entity	281	20
Object, Life Line	Control	138	14
Class	Table	332	13
Object, Life Line Class	Boundary	92	13
Class Attribute	Column	2199	8

Interpretation. The stereotype “entity“ was the most used stereotype within the sample models. In 20 models this stereotype could be found on objects, life lines and classes. The stereotypes

“control“ were found in 14 models on objects and life lines, followed by “table“ and “boundary“ in 13 models. The fifth most used stereotype was “column“ which could be found in 8 models on class attributes.

The stereotypes “entity“, “boundary“ and “control“ belong probably to the UML profile for Software Development Processes. This is an official UML Profile introduced with the UML version 1.3 [21].

Interesting is also the use of the stereotypes “table“ and “column“. It seems that classes were sometimes used as table entities with columns as attributes. This could indicate the use of “Entity Relationship“ (ER) [4] modeling concepts via UML stereotypes in the sample models.

Summary of the Findings

- The UML Profile “Software Development Processes“ was found in several models.
- With the popular stereotypes “table“ and “column“ in the 92 models, classes were used as table constructs were the attributes presented the columns.

Limitations/Future Work. Because of the stereotype management of the EA modelling tool it could not clearly distinguished between UML model elements and model elements from other modelling languages. In this research not all UML model elements could be investigated, only those which could be definitely declared as UML model elements were considered. Therefore a task for further investigations could be to find a way, maybe in cooperation with Sparx Systems, for an exact interpretation of all UML model elements in an EA project. In addition more detailed researches on stereotypes could be made in future work. For example exact classifications of the stereotypes to the UML profile they belong, comparison between user created stereotypes and UML predefined stereotypes, or deeper investigations about how stereotype properties were used.

CHAPTER 5

Related Work

Metrics for UML models have been treated in many studies. UML metrics found in scientific work can be classified into 2 groups. First there is the extensive group of research work about UML quality metrics. Second there is the much smaller group of work dealing with UML usage metrics. The research done in this thesis belongs to the second group.

5.1 UML Quality Metrics

Land and Chaudron [17] did an empirical assessment of the completeness of UML designs. A UML model was seen as complete if for each model element its counterpart could be found. A counterpart for an use case is for example a sequence diagram or a class in a class diagram which is related to the use case.

In another work the complexity of UML class diagrams [10] was measured. One metric measured the number of generalization hierarchies in a class diagram for example.

A similar work [1] dealed with quantitative data metrics (number of classes, number of attributes) of UML models. Only a few metrics are discussed and explained, but not how UML models in general are constructed. For the calculation of the indicators simple OCL queries or scripting languages are used.

An excellent paper about UML class metrics is "A Survey of Metrics for UML Class Diagrams" [11]. This paper describes all the important UML class metrics which exist. All the metrics try to measure the quality of UML class diagrams. Such relevant metric are the CK metrics [5], Li and Henry's metrics [18], MOOD metrics [7] and many more. Some of the metrics measure complexity, coupling, inheritance and polymorphism in UML class diagrams. According to these published metrics, a Web service named "AnalysisWSService" was developed by two

professors from the University of Alcalá in Spain [1]. This service reads in an XMI file of the UML class diagram and calculates in total 37 different metrics about this diagram. The output is a structured HTML page with all the values of the metrics. This output can give the designers a good measurement about the quality of their UML class diagrams and can be used as indicator for software quality.

The Institute of Advanced Computer Science in Leiden, Netherlands, did some research in analyzing UML models [3]. Their main focus laid on goodness and quality of UML models and the variety of modeling styles. Some of the experiences and surveys in the past tried to find out if there is a correlation between class-count and the effort spend in modeling or does the usage of UML models improve software quality. They are also offering free model analyzing service if people send them their models per E-Mail. One interesting point they found out according to their studies were for example that developers apply more detail on critical and complex parts of a system. Another finding was that complexity and coupling was higher for classes that are modeled than not modeled.

Furthermore there is also a software design metrics tool for UML models[2]. This tool analyzes the quality of UML models by calculating for example the degree of coupling or the complexity of UML models. This should serve as a kind of quality benchmark for UML models.

5.2 UML Usage Metrics

Compared to the field of UML quality metrics, there are only a few papers which focused on the usage of UML in practice.

One paper focused on the usage of UML 1.x [6]. In this work the result of a survey about how UML 1.x is used was published. The authors developed a web survey and with the help of the Object Management Group (OMG), the link to the survey was shared all over the OMG members and all the other relevant organizations who are using UML. The main motivation of this project was to find best practices in using UML and to better understand how the language is used. The work analyzed the regular usage of 7 major UML analysis components in projects, which were 6 UML diagrams and Use Case Narratives. The participants were asked in how many projects the distinct UML components were used. The usage-sale consisted of none, < 1/3, 1/3 - 2/3, > 2/3 and in all projects. The results showed that class diagrams were used as most in projects, followed by use case diagrams and sequence diagrams. Use case and sequence diagrams had similar usage rates. In this work class diagrams were also used as most followed by sequence and use case diagrams, which both were almost same in usage as well. As fourth ranked in usage were Use Case Narratives, followed by activity, statechart and collaboration diagrams. Thus, activity can be seen as the fourth often used diagram type. These results are

[1] A Web Service for Calculating the Metrics of UML Class Diagrams. http://www.drdobbs.com/web-development/a-web-service-for-calculating-the-metric/240001719?pgno=1. Accessed: 2013-24-01.

[2] SDMetrics - The Software Design Metrics tool for the UML . http://www.sdmetrics.com. Accessed: 2013-21-01.

similar with the results in Figure 4.11 of Chapter 4.3 in this work, where activity diagrams were the fourth most used diagram type in the observed models. They draw the conclusion that people did not use some of the other UML components because of the lack of knowledge they had about these concepts. All in all according to the authors this research was a first step in finding best practices for UML usage.

Another relevant UML survey about the UML 2.x usage is presented in[3]. But just three questions were asked about the usage. The only question related to the researches of this work was about the diagram type usage. Class and Use Case diagrams were the top frequently used diagram types and the timing diagram was hardly used at all. Further, state machine and sequence diagrams were ranked as fourth and fifth most used diagrams, with slightly different usage rates. Accordingly, in this UML survey state machine diagrams played a more important role than in this work (compare with Figure 4.11, Chapter 4.3). One of the conclusions were that people still know little about all the UML 2.x concepts.

A current work about how UML is used in practice is "UML in Pracice" by Marian Petre [23]. In this paper, five patterns of UML use were identified in interviews with 50 different companies. The patterns were about if UML is used and when how deeply UML is involved in the company processes. In fact only 15 repliers did use UML. Mostly they used UML selectivly in some design parts, no one used it for all the design works in the company. The only question which had a similarity with this work was about the UML diagram usage. 5 different diagram types were used. Class diagrams by 7 users, sequence and activity diagrams by 6 users, state machine diagrams by 3 users and use case diagrams by 1 user. Comparing it with the results of Chapter 4.3, Figure 4.11 the first and second rank is the same.

To find a kernel UML with the most important UML constructs Erickson and Siau conducted a Delphi study [8]. For this, experts where ask to rate the importance of the 9 standard UML 1.x diagrams and related constructs. The scale reached from 1 to 5, 1 was very important and 5 was not important at all. From the results the mean values were calculated. The means of the diagram and constructs were as follows: class 1, use case 1.61, sequence 1.73, statechart 1.81, component 2.31, activity 2.41, collaboration 2.57, deployment 2.69, object 3.00. The diagrams and constructs of class, use case and sequence were the most important. Also in the results of this research work class, use case and interaction concepts and diagrams occurred as most over the 92 models (cf. Chapter 4.2, Figure 4.8 and Chapter 4.3, Figure 4.11). Completely different were the findings about the importance of statechart and object constructs. Objects were seem to be the least important UML constructs but in this work they were widely used in the sample models. State machine constructs were not relevant at all in the sample models, but according to the work of Erickson and Siau statechart is ranked as fourth most important UML diagram and construct.

Another survey where people were asked about the importance of the UML 1.x standard di-

[3]Project Pragamtics, LLC. http://www.projectpragmatics.com/Home/resources-for-you-1/the-uml-survey-results-are-in. Accessed: 2013-03-01.

agrams [13] had similarities with this work. There the most important UML diagrams were class, use case and sequence diagrams as well. Class and use case diagrams were seen as equal important followed closely by sequence diagrams. Statechart, object and activity were close together on the next ranks. Considered as least important were collaboration, component and deployment on the last three ranks. Again the only differences with this work were the importance of statechart and object diagrams.

The work with the most similarity, in case of the research methodology, was written by Reggio, Leotta, Ricca and Clerissi from the Universitiy of Genova [24]. There UML books, UML courses and UML tools were observed for the occurrence of distinct UML constructs. As more often a UML construct appeared as more important it was seen. All UML diagrams from version 2.x, several activity and use case diagram constructs were considered for this evaluation. Class, use case, sequence, activity and state machine diagrams had all an usage of over 90% in all sources. In under 50% of all sources composite structure, timing, interaction-overview and profile diagrams were found. The most relevant use case constructs were use case, actor, extend and include, which were all found in over 90% of the sources. From the 48 considered activity constructs only 9 were seen as very relevant. These 9 were action, control flow edge, initial and final node, decision/merge nodes, fork/join nodes, activity partition, object node and object flow edge. The construct activity was only found in 50% of the sources. Basically the results cover the findings of this work but there exist a few exceptions. First, state machine diagrams were much less relevant in the 92 evaluated sample models (see Figure 4.11, Chapter 4.3). Second, the activity by itself was one of the most used activity constructs in the models, further activity partition, object flow edges and object nodes were rarely used in the models (compare with Table 4.25, Chapter 4.4). Third, the extend and include concepts were also seen as much less relevant in the evaluated models than in [24] (compare with Table 4.24, Chapter 4.4).

According to the related work, we can say that for the two UML metrics groups two different methods of collecting necessary data were used. On one hand quality metrics were mostly calculated by analyzing concrete UML models. On the other hand for the UML usage metrics only surveys or the appearance of UML constructs in books, courses, and tools were evaluated to deduce how UML is used in practice. In contrast, this work calculated and evaluated the usage of UML by analyzing real world models. So far this approach was only used for UML quality metrics. Hence, this is the key difference to the related work about UML usage.

CHAPTER 6

Conclusion and Future Work

In total 92 EA models were analyzed, which extensively used UML. The aim of the work was to understand the practical usage of UML. The usage analysis of UML was split into the following 5 categories:

- UML Language Units
- UML Concepts
- UML Diagrams
- Relationships between UML Language Units
- Stereotype

According to these categories, research questions were formulated for each group, which we have answered in this work.

The considered UML concepts in this work were split into language units, related to UML language units. The results showed that the language units class, use case and interaction were used most. The least used language unit is state machine. In average (median) concepts from 2 different language units were found in one model. The language unit combination class and use case were found as most in the models. With clustering methods, models could be classified according to their used language units. 5 possible classifications for the 92 models could be found. The first class of models consist mainly of use case, object, activity and class concepts. The second and third model classes had their focus only on one language unit, either class or use case. The fourth class had its focus on the language units class, use case, interaction and object. The last model class consisted of 6 different language units, which were class, use case, interaction, component, deployment and composite.

For diagrams the exact same metrics were calculated as for the language units plus additional

diagram metrics. As for language units, the most used diagram types were class, use case and interaction. The least used diagram type is object. As for language units, also for diagrams in average 2 different diagram types were found in one model. Class and interaction diagrams were mostly used together in the models. The clustering method detected 5 different model classes according to their used diagram types. 4 of these model classifications had their focus on one specific diagram type, meaning that this diagram type appeared in almost all of the models within this cluster. One model group had its main focus on activity diagrams, one on class diagrams, one on use case diagrams and one on interaction diagrams. The fifth group consisted mainly of class, use case, interaction, component and deployment diagrams.

To sum it up diagram usage were strongly related to the language units usage in a model, because the model elements were usually drawn in the corresponded diagrams. If classes were modeled usually class diagrams could be found as well, if activities were modeled usually activity diagrams could be found as well and so on. The only difference was with the language unit object and object diagrams. Only in 1 model an object diagram could be found but in 25 models objects were modeled. Objects were usually modeled in interaction diagrams. Further there was found a correlation with interaction diagrams and actors. Actors were mainly found in use case and/or interaction diagrams. More models with interaction diagrams than interaction concepts were found, because in interaction diagrams concepts from several language units could be found. In general interaction diagrams visualized communications between objects, actors and life lines. In class diagrams mainly classes were found and in use case diagrams mainly use cases and actors.

For class, use case and activity models usually 3 diagrams of the related type were used in the model. For instance in a model with class concepts 3 class diagrams could be found in average. The other diagram types appeared in general once in a model. Therefore, models usually had only one diagram representing the visualization of interactions.

In most models only basic UML concepts could be found. Newly introduced UML concepts from version 1.5 and 2.x were rarely used. Many of the considered UML concepts were not found in the sample models. The most popular UML concepts were comments, classes, activities, life lines, objects, use cases, actors, associations, generalizations, messages, use case links and control flows. Comments were even found in the most models (i.e., 67 models). An analyzes of the comment content further showed that generally natural language (mostly with over 100 characters) were used instead of programming language, OCL constraints or other kind of formal calculations.

It seems that the possibility of modeling connections between different language units with EA was not widely used within the 92 models. Only two occurrences of strong cohesion could be found: A very strong cohesion between component and composite concepts and another one between use case, object and interaction components. For all the other language units which appear together in the models only few modeled connections, which the script is able to read out, could be found. For this evaluation the result is clear. Classes were used widely but were mod-

eled encapsulated, describing just a particular part of the system. Class concepts did not interact very often with other language units within the 92 sample models. Same is true for the language unit activity, state machine and deployment. Another result is that use cases were usually described further by interaction constructs, mostly by sequence diagrams and few communication diagrams. The communication partners in these diagrams were mainly actors, objects and life lines. In case of the strong component and composite cohesion the relationships were mostly between components or provided interfaces and part elements.

The top UML concepts which were extended by stereotypes were classes, attributes, operations, objects, dependency relationships and life lines. For objects and life lines the UML Profile "Software Development Processes" were used in most cases. Classes in combination with attributes and operations were modeled as table concepts.

This UML study covers the basic UML constructs. In future work an extended list of considered UML constructs could be analysed. As UML models only EA project files reachable via google were considered. In future work more UML models created with other tools like Eclipse UML 2 Tools [1], Visual Paradigm for UML [2], Modelus Suite [3], etc. could be observed.

Furthermore, companies could be asked directly for UML models, which can be analyzed. Considering more UML tools and ways of how to get UML models would lead to a bigger set of UML models. As bigger the amount of UML models, as more reliable and accurate the results about the UML usage in the practice are. It would be very interesting how similar or different the results of such UML studies are compared to this work.

Some of the used metrics in this work have limitations. In future work these limitations could be overcome. A variety of additional usage metrics could be used in future work as well. Metrics about multiplicity usage, naming conventions of elements and coloring of elements were probably interesting. An concrete example is the way of expressing relations between model elements. In the Appendix A we can see all the possible relationships the script is able to read out from the models. But of course also other ways of expressing relations could be used. As an example with naming conventions this could be made. For instance, an activity is implementing an class operation by having the same name like the operation or an activity describes a use case in the same way. This is another possibility how relations between model elements could be measured. There might be many more interesting usage metrics, which could be established in future work.

[1] http://www.eclipse.org/modeling/mdt/?project=uml2. Accessed: 2013-13-11
[2] http://www.visual-paradigm.com/product/vpuml. Accessed: 2013-13-11
[3] http://vektiva.com/modelus. Accessed: 2013-13-11

APPENDIX A

Appendix

A.1 UML Concepts

All UML concepts the implemented script is able to handle are listed in the following attached tables. Each table has 6 columns. A “Model Element“ represents an EA element which can be directly added to an EA model. An “UML Concept“ is in fact a “Model Element“ but in some cases for generating a specific UML element the created EA element needs further settings. So some model elements can occur in several variations. For example there are five different types of UML classes but EA only provides one model element for all of these five UML elements. To create an abstract class in EA, first a class element has to be created and afterwards in the properties of this EA model element the abstract attribute has to be activated. Therefore a distinction was made between the model element and the UML concept, as one type of a model element can represent different UML concepts. A class can be a class, an association class, an abstract class, an active class or a parameterized class but it will be always treated as class in the evaluation.

The columns “EA Type“, “EA MetaType“, “EA Subtype“ and “Additional AI Calls“ correspond to the appropriate API calls in EA to find out which UML concept the observed EA model element in the models represent. For some model elements different subtype values are possible. For example a model element is declared as class when the type and the metatype has the value “Class“ and the subtype is either 0 or 1 or 2 or 3. All four subtype values are possible to determine that this EA model element is an UML class element. In such cases the different subtype values do not declare any semantical differences as all other model element properties are equal except the subtype. Why EA sometimes save different subtypes for the same model element could not found out clearly, probably different EA versions save model elements in different ways or it sometimes declare the way how this model element was created (for example via the diagram toolbox or directly in the project browser). That some subtypes had no semantical influence on the UML concept it represents in real was determined by manually observing all elements which were declared as unclassified by the script. If the model element was clearly representing a class for example the subtype was added to the list of possible values a class can

have as subtypes in EA.

In the next few pages all model elements which the script is able to read out, grouped by the language units they belong to, are listed.

Table A.1: The Considered Class Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Class	Class	Class	Class	0,1,2,3	
Class	Association Class	Class	AssociationClass	17	
Class	Abstract Class	Class	Class	0,1,2,3	EA.Element.Abstract == 1
Class	Active Class	Class	Class	0,1,2,3	EA.Element.IsActive == 1
Class	Parameterized Class	Class	Class	0,1,2,3	SQL request on DB table t_xref for corresponded EA.Element if column Description has Value == “Type=ClassifierTemplateParameter“
Attribute	Class Attribute	Attribute			
Operation	Class Operation	Operation			
OperationParameter	Class Operation Parameter	OperationParameter			&& EA.OperationParam.Type == ““
Interface	Interface	Interface	Interface	0,1,2,8	EA.Element.StereotypeEx == “interface“
Signal	Signal	Signal	Signal	0	
Nary Association	Nary Association	Association	Association	0	
Enumeration	Enumeration	Class	Enumeration	0,8	EA.Element.StereotypeEx == “enumeration“
PrimitiveType	PrimitiveType	PrimitiveType	PrimitiveType	0	
DataType	DataType	DataType	DataType	0	

Table A.2: The Considered Activity Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
ActivityElement	ActivityElement	Activity	Activity	0,8	
SequenceNode	SequenceNode	Activity	SequenceNode	19	
SequenceNode	SequenceNode with Pin(s)	Activity	SequenceNode	19	Iterate through All sub elements IF EA.Element.Type == “ActionPin“ THEN Node has Pin(s)
SequenceNode	SequenceNode with Expansion Node(s)	Activity	SequenceNode	19	Iterate through All sub elements IF EA.Element.Type == “ExpansionNode“ THEN Node has Node(s)
SequenceNode	SequenceNode with Activity Parameter(s)	Activity	SequenceNode	19	Iterate through All Sub Elements IF EA.Element.Type == “ActivityParameter“ THEN Node has Parameter(s)
Action	AtomicAction	Action	Action	0,8	
Action	CallBehaviorAction	Action	CallBehaviorAction	0,8	
Action	AcceptCallAction	Action	AcceptCallAction	0,8	
Action	AcceptEventAction	Action	AcceptEventAction	0,1,8	
Action	AcceptEventTimerAction	Action	AcceptEventTimerAction	0,8	
Action	AddStructuralFeature-ValueAction	Action	AddStructuralFeatureValueAction	0,8	
Action	AddVariableValueAction	Action	AddVariableValueAction	0,8	
Action	BroadcastSignalAction	Action	BroadcastSignalAction	0,8	
Action	ClearAssociationAction	Action	ClearAssociationAction	0,8	
Action	ClearStructuralFeatureAction	Action	ClearStructuralFeatureAction	0,8	
Action	ClearVariableAction	Action	ClearVariableAction	0,8	
Action	CreateLinkAction	Action	CreateLinkAction	0,8	
Action	CreateLinkObjectAction	Action	CreateLinkObjectAction	0,8	
Action	CreateObjectAction	Action	CreateObjectAction	0,8	
Action	CallOperationAction	Action	CallOperationAction	0,8	
Action	DestroyLinkAction	Action	DestroyLinkAction	0,8	
Action	DestroyObjectAction	Action	DestroyObjectAction	0,8	

Continued on Next Page

Table A.2 – Continued From Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Action	HyperlinkAction	Action	HyperlinkAction	0,8	
Action	RaiseExceptionAction	Action	RaiseExceptionAction	0,8	
Action	ReadExtentAction	Action	ReadExtentAction	0,8	
Action	ReadIsClassifiedObjectAction	Action	ReadIsClassifiedObjectAction	0,8	
Action	ReadLinkAction	Action	ReadLinkAction	0,8	
Action	ReadLinkObjectEndAction	Action	ReadLinkObjectEndAction	0,8	
Action	ReadLinkObject-EndQualifierAction	Action	ReadLinkObjectEndQualifierAction	0,8	
Action	ReadSelfAction	Action	ReadSelfAction	0,8	
Action	ReadStructuralFeatureAction	Action	ReadStructuralFeatureAction	0,8	
Action	ReadVariableAction	Action	ReadVariableAction	0,8	
Action	ReclassifyObjectAction	Action	ReclassifyObjectAction	0,8	
Action	RemoveStructuralFeature-ValueAction	Action	RemoveStructuralFeatureValueAction	0,8	
Action	RemoveVariableValueAction	Action	RemoveVariableValueAction	0,8	
Action	ReplyAction	Action	ReplyAction	0,8	
Action	SendObjectAction	Action	SendObjectAction	0,8	
Action	SendSignalAction	Action	SendSignalAction	0,8	
Action	StartOwnedBehaviorAction	Action	StartOwnedBehaviorAction	0,8	
Action	TestIdentifyAction	Action	TestIdentifyAction	0,8	
Action	ValueSpecificationAction	Action	ValueSpecificationAction	0,8	
Action	WriteLinkAction	Action	WriteLinkAction	0,8	
Action	WriteStructuralFeatureAction	Action	WriteStructuralFeatureAction	0,8	
Action	WriteVariableAction	Action	WriteVariableAction	0,8	
ActionPin	ActionPin	ActionPin	ActionPin	0	
ActionPin	ActionPin - ClassifierType	ActionPin	ActionPin	0	EA.Element.ClassifierID != 0
ActionPin	ActionPin - PrimitiveType	ActionPin	ActionPin	0	SQL request on DB table t_object for corresponded EA.Element if column Classifier_guid is not empty
ActionPin	ActionPin - NoType	ActionPin	ActionPin	0	SQL request on DB table t_object for corresponded EA.Element if column Classifier_guid is empty
ObjectNode	ObjectNode	ObjectNode	ActionPin	0	
ActivityParameter	ActivityParameter	ActivityParameter	ActivityParameter	0	

Continued on Next Page

Table A.2 – Continued From Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
ActivityParameter	ActivityParameter - ClassifierType	ActivityParameter	ActivityParameter	0	EA.Element.ClassifierID != 0
ActivityParameter	ActivityParameter - PrimitiveType	ActivityParameter	ActivityParameter	0	SQL Request on DB table t_object for correspondedEA.Element if column Classifier_guid is empty
ActivityParameter	ActivityParameter - NoType	ActivityParameter	ActivityParameter	0	SQL Request on DB table t_object for corresponded EA.Element if column Classifier_guid is empty
ExpansionRegion	ExpansionRegion in iterative mode	ExpansionRegion	ExpansionRegion	0,8	Iterate through CustomProperties IF CustomProperty.Name == “mode“ && CustomProperty.Value == “iterative“
ExpansionRegion	ExpansionRegion in parallel mode	ExpansionRegion	ExpansionRegion	0,8	Iterate through CustomProperties IF CustomProperty.Name == “mode“ && CustomProperty.Value == “parallel“
ExpansionRegion	ExpansionRegion in stream mode	ExpansionRegion	ExpansionRegion	0,8	Iterate through CustomProperties IF CustomProperty.Name == “mode“ && CustomProperty.Value == “stream“
ExpansionRegion	ExpansionRegion with Expansion Node(s)	ExpansionRegion	ExpansionRegion	0,8	Iterate through Sub Elements IF EA.Element.Type == “ExpansionNode“
Interruptible-ActivityRegion	InterruptibleActivityRegion	InterruptibleActivityRegion	InterruptibleActivityRegion	0	
Interruptible-ActivityRegion	InterruptibleActivityRegion with InterruptFlow(s)	InterruptibleActivityRegion	InterruptibleActivityRegion	0	Iterate through linked connectors IF EA.Connector.Type == “InterruptFlow“
ActivityPartition	ActivityPartition	ActivityPartition	ActivityPartition	0	
ExpansionNode	ExpansionNode	ExpansionNode	ExpansionNode	0	
DataStore	DataStore	Object	DataStore	5,8	EA.Element.StereotypeEx == “datastore“
CentralBufferNode	CentralBufferNode	CentralBufferNode	CentralBufferNode	0	

Continued on Next Page

Table A.2 – Continued From Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
InitialNode	InitialNode	StateNode	Pseudostate	100	
FinalNode	FinalNode	StateNode	Pseudostate	101	
FlowFinalNode	FlowFinalNode	StateNode	Pseudostate	102	
MergeNode	MergeNode	Decision, MergeNode	DecisionNode, MergeNode	0	(InConnectors > OutConnectors) && (OutConnectors == 1)
DecisionNode	DecisionNode	Decision, MergeNode	DecisionNode, MergeNode	0	(OutConnectors > InConnectors) && (InConnectors == 1)
DecisionMergeNode	DecisionMergeNode	Decision, MergeNode	DecisionNode, MergeNode	0	(OutConnectors > 1) && (InConnectors > 1)
Unused DecisionMergeNode	Unused DecisionMergeNode	Decision, MergeNode	DecisionNode, MergeNode	0	(OutConnectors <= 1) && (InConnectors <= 1)
JoinNode	JoinNode	Synchronization	Synchronization	0,1	(InConnectors > OutConnectors) && (OutConnectors == 1)
ForkNode	ForkNode	Synchronization	Synchronization	0,1	(OutConnectors > InConnectors) && (InConnectors == 1)
JoinForkNode	JoinForkNode	Synchronization	Synchronization	0,1	(OutConnectors > 1) && (InConnectors > 1)
Unused JoinForkNode	Unused JoinForkNode	Synchronization	Synchronization	0,1	(OutConnectors <= 1) && (InConnectors <= 1)
ExceptionHandler	ExceptionHandler	ExceptionHandler	ExceptionHandler	0,8	
ExceptionHandler	ExceptionHandler with InterruptFlow(s)	ExceptionHandler	ExceptionHandler	0,8	Iterate through linked connectors IF EA.Connector.Type == “InterruptFlow“
Structured-ActivityNode	StructuredActivityNode	StructuredActivityNode	StructuredActivityNode	16	
Structured-ActivityNode	StructuredActivityNode with Pin(s)	StructuredActivityNode	StructuredActivityNode	16	Iterate through Sub elements IF EA.Element.Type == “ActionPin“
Structured-ActivityNode	StructuredActivityNode with Expansion Node(s)	StructuredActivityNode	StructuredActivityNode	16	Iterate through Sub elements IF EA.Element.Type == “ExpansionNode“

Continued on Next Page

Table A.2 – Continued From Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
LoopNode	LoopNode	LoopNode, Activity	LoopNode	17	
LoopNode	LoopNode with Pin(s)	LoopNode, Activity	LoopNode	17	Iterate through Sub elements IF EA.Element.Type == “ActionPin“
LoopNode	LoopNode with Expansion Node(s)	LoopNode, Activity	LoopNode	17	Iterate through Sub elements IF EA.Element.Type == “ExpansionNode“
ConditionalNode	ConditionalNode	ConditionalNode	ConditionalNode	18	
ConditionalNode	ConditionalNode with Pin(s)	ConditionalNode	ConditionalNode	18	Iterate through Sub elements IF EA.Element.Type == “ActionPin“
ConditionalNode	ConditionalNode with Expansion Node(s)	ConditionalNode	ConditionalNode	18	Iterate through Sub elements IF EA.Element.Type == “ExpansionNode“

Table A.3: The Considered Interaction Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
LifeLine	LifeLine	Sequence	Sequence	0	
CombinedFragment	CombinedFragment - Alt	CombinedFragment	CombinedFragment	0	
CombinedFragment	CombinedFragment - Opt	CombinedFragment	CombinedFragment	1	
CombinedFragment	CombinedFragment - Break	CombinedFragment	CombinedFragment	2	
CombinedFragment	CombinedFragment - Par	CombinedFragment	CombinedFragment	3	
CombinedFragment	CombinedFragment - Loop	CombinedFragment	CombinedFragment	4	
CombinedFragment	CombinedFragment - Critical	CombinedFragment	CombinedFragment	5	
CombinedFragment	CombinedFragment - Neg	CombinedFragment	CombinedFragment	6	
CombinedFragment	CombinedFragment - Assert	CombinedFragment	CombinedFragment	7	
CombinedFragment	CombinedFragment - Strict	CombinedFragment	CombinedFragment	8	
CombinedFragment	CombinedFragment - Seq	CombinedFragment	CombinedFragment	9	
CombinedFragment	CombinedFragment - Ignore	CombinedFragment	CombinedFragment	10	
CombinedFragment	CombinedFragment - Consider	CombinedFragment	CombinedFragment	11	
Gate	Gate	MessageEndpoint	Gate	2	
InteractionState	InteractionState - Invariant	InteractionState	InteractionState	0	
InteractionState	InteractionState - Continuation	InteractionState	InteractionState	1	
MessageEndpoint	MessageEndpoint	MessageEndpoint	MessageEnd	0	
Interaction	Interaction	Interaction	Interaction	0,8	
InteractionParameter	InteractionParameter	Interaction	Interaction	0,8	SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “Type=Parameter“
InteractionOccurrence	InteractionOccurrence	InteractionOccurrence	InteractionOccurrence	0	
MessageLabel	MessageLabel	MessageEndpoint	Comment	4	
State Lifeline	State Lifeline	TimeLine	TimeLine	0	
Value Lifeline	Value Lifeline	TimeLine	TimeLine	1	

Table A.4: The Considered State Machine Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
StateMachine	StateMachine	StateMachine	StateMachine	0,8	
ProtocolStateMachine	ProtocolStateMachine	StateMachine	ProtocolStateMachine	0,8	
State	State	State	State	0,8	
State	State with Regions	State	State	0,8	SQL request on DB table t_xref for corresponded EA.Element if column Description has Value == “@PAR;Name=“
State	SubmachineState	State	State	0,8	Iterate through CustomProperties IF CustomProperty.Name == “isSubmachineState“ && CustomProperty.Value == “-1“
State	State - isOrthogonal	State	State	0,8	Iterate through CustomProperties IF CustomProperty.Name == “isOrthogonal“ && CustomProperty.Value == “-1“ }
State	State - isSimple	State	State	0,8	Iterate through CustomProperties IF CustomProperty.Name == “isSimple“ && CustomProperty.Value == “-1“ }
StateBehaviorElement	StateBehaviorElement				Iterate through Operations
StateBehaviorElement	StateBehavior - Entry				EA.Method.ReturnType == “entry“
StateBehaviorElement	StateBehavior - Do				EA.Method.ReturnType == “do“
StateBehaviorElement	StateBehavior - Exit				EA.Method.ReturnType == “exit“
StateBehaviorElement	StateBehavior - Any				!= “entry“ AND “do“ AND “exit“
InitialPseudoState	InitialPseudoState	StateNode	Pseudostate	3	
FinalState	FinalState	StateNode	FinalState	4	
Junction	Junction	StateNode	Pseudostate	10	
Choice	Choice	StateNode	Pseudostate	11	
Terminate	Terminate	StateNode	Pseudostate	12	
ExitPoint	ExitPoint	StateNode	Pseudostate	14	
EntryPoint	EntryPoint	StateNode	Pseudostate	13	
ShallowHistoryNode	ShallowHistoryNode	StateNode	Pseudostate	5	

Continued on Next Page

Table A.4 – Continued from Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
ShallowHistoryNode	ShallowHistoryNode with default target	StateNode	Pseudostate	5	(OutConnectors >= 1)
DeepHistoryNode	DeepHistoryNode	StateNode	Pseudostate	15	
DeepHistoryNode	DeepHistoryNode with default target	StateNode	Pseudostate	15	(OutConnectors >= 1)
SynchNode	SynchNode	StateNode	Pseudostate	6	
SignalTrigger	SignalTrigger	Trigger	Trigger	0	Iterate through CustomProperties IF CustomProperty.Name == “kind“ && CustomProperty.Value == “Signal“
SignalTrigger	SignalTrigger with Specification	Trigger	Trigger	0	SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “RefGUID=“
CallTrigger	CallTrigger	Trigger	Trigger	0	Iterate through CustomProperties IF CustomProperty.Name == “kind“ && CustomProperty.Value == “Call“
CallTrigger	CallTrigger with Specification	Trigger	Trigger	0	SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “RefGUID=“
TimeTrigger	TimeTrigger	Trigger	Trigger	0	Iterate through CustomProperties IF CustomProperty.Name == “kind“ && CustomProperty.Value == “Time“
TimeTrigger	TimeTrigger with Specification	Trigger	Trigger	0	SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “RefGUID=“
ChangeTrigger	ChangeTrigger	Trigger	Trigger	0	Iterate through CustomProperties IF CustomProperty.Name == “kind“ && CustomProperty.Value == “Change“
ChangeTrigger	ChangeTrigger with Specification	Trigger	Trigger	0	SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “RefGUID=“
AnyTrigger	AnyTrigger	Trigger	Trigger	0	Iterate through CustomProperties IF

Continued on Next Page

Table A.4 – Continued from Previous Page

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
AnyTrigger	AnyTrigger with Specification	Trigger	Trigger	0	CustomProperty.Name == “kind“ && CustomProperty.Value == ““ SQL Request on DB Table t_xref for corresponded EA.Element if column Description has Value == “RefGUID=“

Table A.5: The Considered Use Case Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI calls
UseCase	UseCase	UseCase	UseCase	0,8	
Boundary	Boundary	Boundary	Boundary	0	
Actor	Actor	Actor	Actor	0,8	

Table A.6: The Considered Object Concepts

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Object	Object	Object	Object	0,8	

Table A.7: The Considered Component Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Component	Component	Component	Component	0,8	
Packaging Component	Packaging Component	Package	Package	20	
ProvidedInterface	RequiredInterface	RequiredInterface	RequiredInterface	1	
RequiredInterface	ProvidedInterface	ProvidedInterface	ProvidedInterface	0	

Table A.8: The Considered Deployment Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Artifact	Artifact	Artifact	Artifact	0,1	
DeploymentSpecification	DeploymentSpecification	DeploymentSpecification	DeploymentSpecification	0	
Device	Device	Device	Device	0	
ExecutionEnvironment	ExecutionEnvironment	ExecutionEnvironment	ExecutionEnvironment	0	
NodeElement	NodeElement	Node	Node	0	

Table A.9: The Considered Composite Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Collaboration	Collaboration	Collaboration	Collaboration	0,8	
Collaboration Use	Collaboration Use	CollaborationOccurrence	CollaborationOccurrence	0,8	
PartElement	PartElement	Part	Part	0	
Port	Port	Port	Port	0	

Table A.10: The Considered Auxiliary Construct Elements

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
InformationItem	InformationItem	InformationItem	InformationItem	0	
Comment	Comment	Note	Note	0,1,2	
Constraint	Constraint	Constraint	Constraint	0	

A.2 UML Diagrams

All considered UML diagrams are listed in Table A.11. The column *"Diagram Type"* stands for the UML diagram type. Column *"EA Type"* and *"EA MetatType"* correspond to the appropriate API calls in EA to find out which UML diagram type the observed EA diagram element in the models represent.

In Table A.12 all considered EA connectors, which were categorized to the different UML relationships, are listed. This Table is defined like the Tables in the Appendix A.1 above.

The last Table A.13 in this section lists all considered UML relationships, which were not expressed by an EA connector element.

Table A.11: The Considered Diagram Types

Diagram Type	EA Type	EA MetaType
ClassDiagram	Logical, Class	empty
UseCaseDiagram	Use Case UML Behavioral::Use Case	empty
StateMachineDiagram	Statechart, UML Behavioral::State Machine	empty
ActivityDiagram	Activity, UML Behavioral::Activity	empty
SequenceDiagram	Sequence, UML Behavioral::Sequence	empty
DeploymentDiagram	Deployment	empty
ComponentDiagram	Component	empty
CompositeStructureDiagram	CompositeStructure	empty
ObjectDiagram	Object	empty
PackageDiagram	Package	empty
CommunicationDiagram	Collaboration	empty
TimingDiagram	Timing	empty
InteractionOverviewDiagram	InteractionOverview	empty
SYSML::ActivityDiagram	Activity	SysML1.2::Activity
SYSML::BlockDefinition	Logical	SysML1.2::BlockDefinition
SYSML::InternalBlock	CompositeStructure	SysML1.2::InternalBlock
SYSML::PackageDiagram	Package	SysML1.2::Package
SYSML::ParametricDiagram	CompositeStructure	SysML1.2::Parametric
SYSML::RequirementDiagram	Custom	SysML1.2::Requirement
SYSML::SequenceDiagram	Sequence	SysML1.2::Sequence
SYSML::StateMachineDiagram	Statechart	SysML1.2::StateMachine
SYSML::UseCaseDiagram	Use Case	SysML1.2::UseCase

Table A.12: The Considered UML Connectors

Model Element	UML Concept	EA Type	EA MetaType	EA Subtype	Additional AI Calls
Association	Association	Association	Association	Class, *empty*	
Generalization	Generalization	Generalization	Generalization	*empty*	
Realisation	Realisation	Realisation	Realisation	*empty*	EA.Connector.Stereotype == “realize“, EA.Connector.Stereotype == ““
Dependency	Dependency	Dependency	Dependency	*empty*	
Aggregation	Aggregation	Aggregation	Aggregation	Weak, *empty*	
Composition	Composition	Aggregation	Aggregation	Strong	
ControlFlow	ControlFlow	ControlFlow	ControlFlow	*empty*	
ObjectFlow	ObjectFlow	ObjectFlow	ObjectFlow	*empty*	
UseCaseLink	UseCaseLink	UseCase	UseCaseLink	*empty*	
Extend	Extend	UseCase	UseCaseLink	Extends	EA.Connector.StereotypeEx == “extend“
Include	Include	UseCase	UseCaseLink	Includes	EA.Connector.StereotypeEx == “include“
Message Asynchron	Message Asynchron	Sequence	Sequence	*empty*	
Message Asynchron - New	Message Asynchron - New	Sequence	Sequence	New	
Message Asynchron New - Reply	Message Asynchron New - Reply	Sequence	Sequence	New	EA.Connector.MiscData(3) == “1“
Message Asynchron - Delete	Message Asynchron - Delete	Sequence	Sequence	Delete	
Message Asynchron - Delete - Reply	Message Asynchron - Delete - Reply	Sequence	Sequence	Delete	EA.Connector.MiscData(3) == “1“
Message Synchron	Message Synchron	Sequence	Sequence	*empty*	
Message Synchron - New	Message Synchron - New	Sequence	Sequence	New	
Message Synchron - New - Reply	Message Synchron - New - Reply	Sequence	Sequence	New	EA.Connector.MiscData(3) == “1“
Message Synchron - Delete	Message Synchron - Delete	Sequence	Sequence	Delete	
Message Synchron - Delete - Reply	Message Synchron - Delete - Reply	Sequence	Sequence	Delete	EA.Connector.MiscData(3) == “1“
Transition	Transition	StateFlow	Transition	*empty*	
Trace	Trace	Dependency	Trace	*empty*	EA.Element.StereotypeEx == “trace“
Abstraction	Abstraction	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “abstraction“

Continued on Next Page

Table A.12 – Continued from Previous Page

Model Element	Concept	Type	MetaType	Subtype	Additional AI Calls
Derive	Derive	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “derive“
Refine	Refine	Dependency	Refine	*empty*	EA.Element.StereotypeEx == “refine“
UseDependency	UseDependency	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “use“
Occurence	Occurence	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “occurrence“
Represents	Represents	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “represents“
Role Binding	Role Binding	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “role binding“
Assembly	Assembly	Assembly	Assembly	*empty*	
Delegate	Delegate	Delegate	Delegate	*empty*	
Nesting	Nesting	Nesting	Nesting	*empty*	
PackageImport	PackageImport	Package	PackageImport	*empty*	EA.Element.StereotypeEx == “import“
PackageMerge	PackageMerge	Package	PackageMerge	*empty*	EA.Element.StereotypeEx == “merge“
Manifest	Manifest	Manifest	Manifest	*empty*	EA.Element.StereotypeEx == “manifest“
TemplateBinding	TemplateBinding	TemplateBinding	TemplateBinding	*empty*	
NoteLink	NoteLink	NoteLink	NoteLink	*empty*	
InformationFlow	InformationFlow	InformationFlow	InformationFlow	*empty*	
Instantiate	Instantiate	Dependency	Dependency	*empty*	EA.Element.StereotypeEx == “instantiate“
CommunicationMessage	CommunicationMessage	Collaboration	Collaboration	*empty*	

Table A.13: Special Considered UML Relationships

Source Language Unit	Possible Target Language Units	Relationship Name	Description
State Machine	Act, Int, SM	StateBehavior2Behavior	The behavior of a state refers to a modeled behavior
Class	Cl, Int, SM, UC, Act, Comp, Deploy, Composite	StateBehavior2Operation	The behavior of a state represents an operation
Activity	Cl, Act, UC, Comp, Deploy, Composite	ActionPin2Classifier	An action pin which represents a classifier
Activity	Act, Int, SM	Action2Behavior	An action calls a behavior
Activity	Cl	Action2Signal	An action sends/receives a signal
Activity	Cl, Int, SM, UC, Act, Comp, Deploy, Composite	Action2Operation	An action calls an operation
Activity	SM	Action2Trigger	An action calls a trigger
Object	Cl, Act, UC, Comp, Deploy, Composite	Object2Classifier	A object represents a classifier
Activity	Cl, Act, UC, Comp, Deploy, Composite	ActivityParameter2Classifier	An activity parameter represents a classifier
Class	Act, Int, SM	ClassOperation2Behavior	A class operation calls a behavior
Class	Cl, Act, UC, Comp, Deploy, Composite	ClassOperationReturnType2Classifier	The return type of a class operation represents a classifier
Class	Cl, Act, UC, Comp, Deploy, Composite	ClassOperationParamType2Classifier	An operation parameter has a classifier as type
Class	Cl, Act, UC, Comp, Deploy, Composite	ClassAttributeType2Classifier	A class attribute has a classifier as type
State Machine	Cl, Act, UC, Comp, Deploy, Composite	StateBehaviorParamType2Classifier	A parameter of a state behavior represents a classifier
State Machine	Cl, Int, SM, UC, Act, Comp, Deploy, Composite	CallTrigger2Operation	A trigger calls an operation
State Machine	Cl	SignalTrigger2Signal	A signal calls a behavior
Interaction	Cl, Act, UC, Comp, Deploy, Composite	LifeLine2ClassifierInstance	The life line is an instance of a classifier
Interaction	Cl, Act, UC, Comp, Deploy, Composite	InteractionReturnType2Classifier	The interaction return type has a classifier as type
Interaction	Cl, Act, UC, Comp, Deploy, Composite	InteractionParameterType2Classifier	An interaction parameter has a classifier as type
Interaction	Act, Int, SM	InteractionOccurrence2Behavior	An interaction occurrence which calls a behavior
Use Case	Cl, Act, UC, Comp, Deploy, Composite	UseCase2Classifier	The use case has an instance as classifier type
Use Case	Cl, Act, UC, Comp, Deploy, Composite	Actor2Classifier	The actor has an instance as classifier type
	*	Element2Diagram	A model element which is described in an diagram

Cl = Class, UC = Use Case, Int = Interaction, Act = Activity

Comp = Component, Deploy = Deployment, SM = State Machine

A.3 Model Sources

A full list of all observed EA models can be found in Table A.14. The full URL for each EA model is provided in column *"SOURCE"*. The values in column *"EAP FILE"* represent the ea file names as provided in the project website of this work.

Table A.14: Model Sources (Accessed - 2013-12-03)

SOURCE	EAP FILE
https://github.com/hoggier/DAPIV/blob/master/Academico.eap	Academico.eap
http://trac.assembla.com/GrupoTallerProgramacion2/browser/doc/adminEdificio.eap	adminEdificio.eap
http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CEUQFjAA&url=http%3A%2F%2Fdl.dropbox.com%2Fu%2F45486%2Farc42-downloads%2Farc42-V40-EN.eap&ei=3K3tULaiM9GSswbK5oHQDA&usg=AFQjCNGjmDTzUwEa_6WKPagrRFaLOew6mQ&sig2=JNDGeRpt8yjBr9wCuEdEFg&bvm=bv.1357316858,d.d2kHUM2ZJ8TLtAaJm4DYCg&usg=AFQjCNGjmDTzUwEa_6WKPagrRFaLOew6mQ&sig2=zDMVezkDV4PIRWQU_xqmOQ	arc42-V40-EN.eap
http://assetsdev.atc.gr/trac/browser/assets/trunk/z_project_setup/documentation/assets-models.eap?rev=4624	Assets-models.eap
http://code.google.com/p/my-ibs/source/browse/trunk/+my-ibs/BookStore.eap	BookStore.eap
http://code.google.com/p/cockus3d/source/browse/tags/core/c3d.eap	c3d.eap
gforge.nci.nih.gov/frs/download.php/5831/caBIO42.eap	caBIO42.eap
http://code.google.com/p/pizza/source/browse/branches/tp2Reentega/diagramas/secuencias/registrarPedido/calcularTiempo.EAP?r=252	calcularTiempo.EAP
https://www.assembla.com/code/cartech/subversion/nodes/Modely/CarTech.eap	CarTech.eap
http://code.google.com/p/umm2-addin/source/browse/trunk/CCLImporter/input/CCL08A.eap?r=78	CCL08A.eap
http://proj.badc.rl.ac.uk/pimms/browser/ControlledVocabs/trunk/Activity/CMIP5_Experiments/CMIP5_experiments.eap?rev=7	CMIP5_experiments.eap
https://www.assembla.com/code/cod4tv/subversion/nodes/Documentation/codtv.eap	codtv.eap
http://trac.assembla.com/counter_speechport/browser/cspeechport.eap?rev=5	cspeechport.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU05.Registrar presupuesto de orden de trabajo com.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU05.Registrar presupuesto de orden de trabajo.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU14.Generar Notificacion al Cliente com.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU14.Generar Notificacion al Cliente.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU70.Generar Informe de Reparaciones com.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	CU70.Generar Informe de Reparaciones.eap
http://trac.openmicroscopy.org.uk/ome/browser/ome-xml/Documentation/Diagrams/Enterprise?rev=178	DataModel.eap
http://trac.lternet.edu/trac/NIS/browser/trunk/DataPackageManager/documents/DataPackageManager.eap	DataPackageManager.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	Diagram aktivit JK,TR.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	Diagram aktivit JN,MZ.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	Diagram aktivit MZ.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	Diagram aktivit PD,MM.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Diagrama de Clases de Analisis.eap

Continued on Next Page

Table A.14 – Continued from Previous Page

SOURCE	EAP FILE
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	Use case JN,MZ.eap
https://www.assembla.com/code/HP_UTN/subversion/nodes/Diagramas%20UML.eap?rev=87	Diagramas UML.eap
http://pm.stu.cn.ua/repositories/changes/any2any/trunk/docs/diplom.eap	diplom.eap
http://trac.assembla.com/soray/browser/user/Marcell/DiplomaThesis%20%282%29.eap?rev=390	DiplomaThesis(2).eap
http://myhomemdz.dyndns.org/viewvc/freedom/trunk/Dise%C3%B1o%20de%20Sistemas.eap?view=log&r1=45&pathrev=46	Diseño de Sistemas.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	DTEs.eap
http://ndg-security.ceda.ac.uk/browser/trunk/NDGSecurity/documentation/esgInteroperabilityForIPCCar5/esg.eap?rev=7917	esg.eap
http://ndg-security.ceda.ac.uk/browser/TI12-security/trunk/documentation/esgInteroperabilityForIPCCar5/esg-ipcc-ar5.eap?rev=4680	esg-ipcc-ar5.eap
http://www.wuala.com/antunes20/ANDROID/esof_SVE.eap	esof_SVE.eap
http://trac.lternet.edu/trac/NIS/browser/trunk/DataPortal/documents/EventSubscriptionService.eap	EventSubscriptionService.eap
http://exdb.fit.cvut.cz/browser/doc/EXDB.eap?rev=1068	EXDB.eap
http://trac.assembla.com/antrad_svn/browser/trunk/UftSw/model/FirstBlood.EAP?rev=120	FirstBlood.eap
http://trac.lternet.edu/trac/NIS/browser/trunk/Gatekeeper/documents/gatekeeper.eap	gatekeeper.eap
http://code.google.com/p/g0c/source/browse/GoC.eap	GoC.eap
https://www.assembla.com/code/SIwP/subversion/nodes/honda.eap	honda.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/K%20odevzd%C3%A1n%C3%AD/Iterace1%20%28bez%20DT%29.eap	Iterace1 (bez DT).eap
http://code.google.com/p/2-1-risiko/source/browse/trunk/Risiko/UML/Aufgabe_01/klassendiagramm_final.EAP	klassendiagramm_final.EAP
http://hitsp.wikispaces.com/file/detail/Library+May07.eap	Library May07.eap
https://github.com/Regala/Micro-Machines-Java/blob/master/LPOO_Proj2_UML.eap	LPOO_Proj2_UML.eap
http://ndg-security.ceda.ac.uk/browser/trunk/NDGSecurity/documentation/MashMyData/MashMyData.eap?rev=7917&order=name	MashMyData.eap
http://hssp-implementation.wikispaces.com/file/detail/metamodel.eap	metamodel.eap
http://code.google.com/p/mille-kanallies/source/browse/trunk/Mill/doc/Mille-Kanallies-ClassDiagram-Raw.eap	Mille-Kanallies-ClassDiagram-Raw.eap
http://code.google.com/p/netcat-explained/source/browse/miniNetcat/Documentacion/miniNetcat-project.eap	MiniNetcat-project.eap
https://www.assembla.com/code/dt-localization/subversion/nodes/trunk/Analysis/model.eap?rev=39	model.eap
http://code.google.com/p/sidov/source/browse/trunk/Repositorio/Requerimientos/ModeladoSIDOV.eap?r=113	ModeladoSIDOV.eap
http://code.google.com/p/nyx/source/browse/models.eap	models.eap
http://code.google.com/p/studiadrugiegostopnia/	moduly.eap
http://code.google.com/p/netcat-explained/source/browse/netcat-explained-project.eap	netcat-explained-project.eap
https://view.softwareborsen.dk/Softwareborsen/oio-desktop/docs/OIO-Desktop%20Architecture.eap?view=log	OIO-Desktop Architecture.eap
http://trac.openmicroscopy.org.uk/ome/browser/ome-xml/Documentation/Diagrams/Enterprise?rev=178	OmeroDbDiagrams.eap

Continued on Next Page

Table A.14 – Continued from Previous Page

SOURCE	EAP FILE
http://fisheye.ow2.org/browse/Sirocco/sandbox/pawel/OpenCloud-Placement/OpenCloud_model.eap?hb=true	OpenCloud_model.eap
http://trac.lternet.edu/trac/NIS/browser/documents/system-design/PASTA.eap	Pasta.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Patron State.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Patrones.eap
http://bio-models.svn.sourceforge.net/viewvc/bio-models/trunk/object_models/enterprise_architect/phenotype.eap?view=log	phenotye.eap
http://trac.openmicroscopy.org.uk/ome/browser/ome-xml/Documentation/Diagrams/Enterprise?rev=178	PostEvolution.eap
http://trac.assembla.com/ppro-bofe/browser/Reports/2/ppro2.eap	ppro2.eap
http://assetsdev.atc.gr/trac/browser/assets/trunk/services/preservation-riskmanagement/src/model/preservation-riskmanagement.eap?rev=6426	preservation-riskmanagement.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Primera Entrega - Modelo de Dominio - Version 2.1.eap
http://chomikuj.pl/mredwanz/semestr+V+%28systemy%29/In*c5*bcynieria+Oprogramowania/laboratorium/projekt,425192160.EAP	project.eap
http://trac.assembla.com/remigol_projekt/browser/projekt.eap?rev=66	Projekt (2).eap
http://trac.assembla.com/remigol_projekt/browser/projekt%2Bmoj.eap?rev=71	projekt+moj.eap
http://code.google.com/p/proyecto-final-alquileres/source/browse/trunk/Diagramas/ProyectoFinal.EAP?r=18	ProyectoFinal.eap
https://project.fit.cvut.cz/trac/UHKT2/browser/SP1/1.%20Iterace/Soubory%20pro%20EA	První návrh UHKT JN,MZ.eap
https://trac.lternet.edu/trac/NIS/browser/trunk/DataManager/documents/QualityModel.eap	QualityModel.eap
http://code.google.com/p/si-laundry/source/browse/Robustness-5208100034-update.eap	Robustness-5208100034-update.eap
http://code.google.com/p/s1n4c3c/source/browse/S1N4C3C/S1N4C3C_v01.eap?r=4	S1N4C3C_v01.eap
http://trac.openmicroscopy.org.uk/ome/browser/ome-xml/Documentation/Diagrams/Enterprise?rev=178	ScreenWell.eap
https://www.assembla.com/code/se-space/subversion/nodes/Sequence%20Diagrams.eap?rev=15	Sequence Diagrams.eap
http://code.google.com/p/sidov/source/browse/trunk/Repositorio/Requerimientos/Sidov.EAP?r=350	Sidov.eap
http://code.google.com/p/studiadrugiegostopnia/	sklep iconix.eap
http://dev.herasaf.org/source/browse/ERCPRA/trunk/herasaf-ercpra-documentation/diagrams	Solution_GEF_DirectEdit_Structure.eap
http://dev.herasaf.org/source/browse/ERCPRA/trunk/herasaf-ercpra-documentation/diagrams	Solution_GEF_Domain-DiagramModel.eap
http://dev.herasaf.org/source/browse/ERCPRA/trunk/herasaf-ercpra-documentation/diagrams	Solution_GEF_Structure.eap
http://dev.e-taxonomy.eu/trac/attachment/wiki/Revisionary_Models/MergedModel1/UnifiedModel.eap	UnifiedModel.eap
http://pamediakopes.wikispaces.com/file/detail/use_case_model.eap	use_case_model.eap
https://www.assembla.com/code/se-space/subversion/nodes/UseCase%20Diagram.eap?rev=15	UseCase Diagram.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Vista Arquitectonica de Despliegue – Componentes.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Vista Arquitectonica de la Funcionalidad.eap
code.google.com/p/taller-mecanico-dsi2011-tripode/source/browse/trunk/	Vista Arquitectonica de Subsistemas e Interfaces.eap
http://code.google.com/p/absolutdocs/source/browse/trunk/doc/disenio/componentes/visualizador/Visualiza	Visualizador.eap

Continued on Next Page

Table A.14 – Continued from Previous Page

SOURCE	EAP FILE
dor.eap?r=362	
http://assetsdev.atc.gr/trac/browser/assets/trunk/services/visual-loganalysis/src/main/model/visual-loganaly sis.eap?rev=12854	Visual-loganalysis.eap
http://code.google.com/p/voip-sec/	VoIPSecCPU.eap
http://code.google.com/p/voip-sec/	VoIPSecTime.eap
http://taskman.eionet.europa.eu/projects/reportnet/wiki/WiseDS	wiseDS_architecture_v0_9.eap
http://code.google.com/p/studiadrugiegostopnia/	Wniosek o urlop.eap

Bibliography

[1] A. Baroni and F. Abreu. Formalizing Object-Oriented Design Metrics upon the UML Meta-Model. In *16th Brazilian Symposium on Software Engineering*, 2002.

[2] C.H. Brase and C.P. Brase. *Understandable Statistics: Concepts and Methods*. BROOKS COLE Publishing Company, 2011.

[3] M. R. V. Chaudron. Quality Assurance for UML Modeling. In *Software Quality Days, January, Vienna, Austria*, 2012.

[4] P. P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *Association for Computing Machinery (ACM) Transactions on Database Systems*, volume 1, pages 9-36, 1976.

[5] S. Chidamber and C. Kemerer. Towards a Metrics Suite for Object Oriented Design. In *Conference on Object-Oriented Programming: Systems, Languages and Applications (OOSPLA'91)*, volume 26, pages 197–211. SIGPLAN Notices, 1991.

[6] B. Dobing and J. Parsons. Current practices in the Use of UML. In *Proceedings of the 24th International Conference on Perspectives in Conceptual Modeling (ER'05)*, pages 2–11, Berlin, Heidelberg, 2005. Springer-Verlag.

[7] F. Brito e Abreu and R. Carapuça. Object-Oriented Software Engineering: Measuring and Controlling the Development Process. In *4th Interntional Conference on Software Quality, Mc Lean, VA, USA*, 1994.

[8] J. Erickson and K. Siau. Can UML be Simplified? Practitioner use of UML in separate Domains. In *proceedings of 12th International Workshop on Exploring Modeling Methods in System Analysis and Design (EMMSAD)*, pages 89–98, 2007.

[9] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Languange*. Object Technology Series. Addison Wesley Professional, 2004.

[10] M. Genero and M. Piattini. Empirical Validation of Measures for Class Diagram Structural Complexity through controlled Experiments. In *Proceedings of 5th International ECOOP Workshop on Quantitative Approaches in object-oriented Software Engineering (QAOOSE 2001), June, Budapest, Hungary*, 2001.

[11] M. Genero, M Piattini, and C. Calero. A Survey of Metrics for UML Class Diagrams. *Journal of Object Technology*, 4(9):55–92, November-December 2005.

[12] F.J. Gravetter and L.B. Wallnau. *Statistics for the Behavioral Sciences*. Available Titles Aplia Series. Wadsworth Cengage Learning, 2009.

[13] M. Grossman, J. E. Aronson, and R. V. McCarthy. Does UML make the Grade? Insights from the Software Development Community. *Information and Software Technology*, 47(6):383–397, April 2005.

[14] G.K. Gupta. *Introduction To Data Mining With Case Studies*. Prentice-Hall Of India Pvt. Limited, 2006.

[15] M. Hitz, G. Kappel, E. Kapsammer, and W. Retschitzegger. *UML 2 @ Work, Objektorientierte Modellierung mit UML 2*. dpunkt.verlag, 3. edition, 2005 (in German).

[16] C. Kobryn. Will UML 2.0 be Agile or Awkward? *Communications of the Association for Computing Machinery*, 45(1):107–110, January 2002.

[17] C. F. J. Lange and M. R. V. Chaudron. An Empirical Assessment of Completeness in UML Designs. In *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE'04)*, pages 111–121, 2004.

[18] W. Li and S. Henry. Object-Oriented Metrics that Predict Maintainability. *Journal of Systems and Software*, 23(2):111–122, 1993.

[19] Object Management Group. *OMG fUML Sepcification, Version 1.1, 2013*. Available at http://www.omg.org/spec/FUML/1.1/PDF/.

[20] Object Management Group. *OMG MDA Guide, Version 1.0.1, 2003*. Available at http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf.

[21] Object Management Group. *OMG UML Sepcification, Version 1.3, 2001*. Available at http://www.omg.org/spec/UML/1.3/PDF.

[22] Object Management Group. *OMG UML Sepcification, Version 2.4.1, 2011*. Available at http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF.

[23] M. Petre. UML in Practice. In *International Conference on Software Engineering (ICSE'13)*, 2013.

[24] G. Reggio, M. Leotta, F. Ricca, and D. Clerissi. Downsize the UML: A Preliminary Survey Detecting the Used Constructs. Technical Report DISI-TR-13-02, Department of Computer and Information Science - University of Genoa, 2013.

[25] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, Second Edition*. Pearson Higher Education, 2004.

[26] C. Rupp, S. Queins, and B. Zengler. *UML 2 Glasklar: Praxiswissen für die UML-Modellierung*. Hanser, 2007 (in German).

[27] J. Walkenbach. *Excel 2007 Power Programming with VBA*. Mr. Spreadsheet's Bookshelf. Wiley, 2011.