SafeEx: A tool for integrity enforcement in UML conceptual schemas

Master in Information Technology

Tutor: Ernest Teniente

Student: Ahmet Oktay Kabar

August 2014
Acknowledgments

I would like to thank everybody who helped me to realize my thesis successfully. Firstly, I would like to thank Xavier Oriol for all the support he gave me throughout the work, and for great collaboration. Also, I am grateful to my tutor, Ernest Teniente, for helping and supporting me and for his and patience and professionalism.

I would also like to appreciate especially Prof. Conrado Martinez for supporting and helping me during this master period and all the lecturers of the UPC who taught me very great courses and very deep knowledge during this master, which not only helped me huge effect on this thesis, but also is very important for my career and my professional life.

More importantly, I would like to thank to my family, who has always supported me during this master study period. Without their support, I would not manage to complete this. Lastly, my all thanks go to my classmates and friends for make this thesis happen. I really enjoyed studying with them during this 2 years.
Figures

Figure 1 – UML design examples ........................................................................................................... 12
Figure 2 – UML class diagram ............................................................................................................... 13
Figure 3 – OCL contract ......................................................................................................................... 14
Figure 4 - Cardinality ............................................................................................................................. 15
Figure 5 – Converted XMI file .............................................................................................................. 17
Figure 6 - Java .......................................................................................................................................... 28
Figure 7 - Eclipse ...................................................................................................................................... 28
Figure 8 - ArgoUML .................................................................................................................................. 29
Figure 9 - DES .......................................................................................................................................... 30
Figure 10 – Class diagram of reused code ............................................................................................. 32
Figure 11 – UML diagram ....................................................................................................................... 33
Figure 12 – Class representation in XMI ................................................................................................. 33
Figure 13 - Parameter representation in XMI .......................................................................................... 34
Figure 14 – Des inclusion into development .......................................................................................... 39
Figure 15 – DES main page ...................................................................................................................... 40
Figure 16 – Instance insertion algorithm ............................................................................................... 44
Figure 17 - Unsafe variable correction ................................................................................................... 46
Figure 18 – Unsafe error correction algorithm ....................................................................................... 47
Figure 19 – Integrity constraint detection algorithm ............................................................................... 49
Figure 20 – Fixing integrity constraint problem ..................................................................................... 53
Figure 21 – Fixing additional integrity constraint problem ........................................................................ 58
Figure 22 – Sample UML/OCL diagram ................................................................................................. 59
Figure 23 – Fixing errors iteratively ...................................................................................................... 62
Figure 24 – Agile methodology ............................................................................................................... 65
Figure 25 – Initial planning ...................................................................................................................... 72
Figure 26 – Office resources .................................................................................................................... 74
Figure 27 – Human resources .................................................................................................................. 75
Figure 28 – Total cost ............................................................................................................................... 75
Content

Acknowledgments ........................................................................................................................................... 1

Figures .......................................................................................................................................................... 2

1. Overview .................................................................................................................................................... 8

  1.1) Introduction ......................................................................................................................................... 8

  1.2) Motivation ........................................................................................................................................... 9

  1.3) Project Goals ..................................................................................................................................... 10

  1.4) Document Structure .......................................................................................................................... 11

2. State of the Art ......................................................................................................................................... 12

  2.1) UML .................................................................................................................................................... 12

  2.2) OCL .................................................................................................................................................... 13

  2.3) UML/OCL Conceptual Schema ......................................................................................................... 14

    2.3.1) Contract ..................................................................................................................................... 14

    2.3.2) Constraints ................................................................................................................................. 15

  2.4) Schema Conversion ............................................................................................................................ 16

  2.5) Related Works ................................................................................................................................... 17

3) Analysis of SafeEx Tool and Design Decisions ...................................................................................... 19

  3.1) Requirement Analysis ....................................................................................................................... 19

    3.1.1) Functional Requirements ............................................................................................................ 21

      3.1.1.1) The system should get converted UML/OCL conceptual schema as an input ...................... 21

      3.1.1.2) The system should translate XMI file ................................................................................... 22

      3.1.1.3) The system should be available for inserting the instances ............................................... 22

      3.1.1.4) The system should be available for inserting the Dependency Constraint Equations .......... 23
3.1.1.5) The system should be able to detect the integrity constraint problems................................. 23
3.1.1.6) The system should propose solution for the constraint problem .............................................. 24
3.1.1.7) The system should solve the constraint problem ........................................................................ 24
3.1.2) Non-Functional Requirements ...................................................................................................... 25
3.1.2.1) The tool must be compatible with any kind of computer or laptop. .............................................. 25
3.1.2.3) The tool must be compatible with UML/OCL Schema Tools ...................................................... 26
3.2) Analysis of Architectural Decisions .................................................................................................... 26
3.2.1) Datalog Educational System(DES) ................................................................................................. 26
3.3) Development of Platform and Technological Decisions .................................................................... 28
3.3.1) Java .............................................................................................................................................. 28
3.3.2) Eclipse .......................................................................................................................................... 28
3.3.3) ArgoUML ...................................................................................................................................... 29
3.3.4) Datalog Educational System .......................................................................................................... 30
4. Development of SafeEx Tool .................................................................................................................. 30
4.1) Code Reusing ................................................................................................................................... 31
4.1.1) UML/OCL Schema Conversion .................................................................................................... 32
4.1.2) XMI Converter ............................................................................................................................. 34
4.2) Event Dependency Constraint ........................................................................................................... 35
4.3) Literals .............................................................................................................................................. 36
4.3.1) Ordinary Literal ............................................................................................................................. 36
4.3.2) BuiltIn Literal ............................................................................................................................... 37
4.5) Instance ............................................................................................................................................ 37
4.6) Detecting Integrity Constraint Violation ............................................................................................ 37
4.6.1) Getting all EDC ............................................................................................................................ 38
4.6.2) Obtaining DES Connection ................................................................. 38
4.6.3) EDC Syntax Correction for DES .......................................................... 41
4.6.4) Instance Syntax ..................................................................................... 43
4.6.5) Inserting Instances into DES ................................................................. 43
4.6.6) Inserting EDC into DES ....................................................................... 44
4.6.7) Detecting Integrity Constraint Violation ............................................. 48
4.7) Finding Problematic Instance in EDC ..................................................... 50
4.8) Integrity Constraint Maintenance ............................................................ 52
   4.8.1) Converting Event-Dependency Constraint (EDC) to Repair-Generating Dependency (RGD) ........... 53
   4.8.2) Instance Insertion to fix Problem ...................................................... 55
   4.8.3) Validation Solution of Integrity Constraint Violation ....................... 57
   4.8.4) Additional Integrity Constraint Problems ....................................... 57
   4.8.4) Iterative Approach to Fix Integrity Constraint(IC) Violation ............ 61
4.9) Development Process ............................................................................. 63
   4.9.1) Agile Software Development ............................................................ 64
   4.9.2) Chosen Process Model for the Project ............................................. 65
4.9) SafeEx Tool Iterations .......................................................................... 67
   4.9.1) Iteration I: Familiarize with Conceptual Schema, UML/OCL and Integrity Constraint .......... 67
   4.9.2) Iteration II: Perform the Integration of Schema Conversion ............. 68
   4.9.3) Iteration III: Perform the implementation of Repair-Generating Dependency (RGD) .......... 68
   4.9.4) Iteration IV: Perform the implementation of Integrity Constraint Violation Checking .......... 69
   4.9.5) Iteration V: Perform the Implementation of Integrity Constraint Maintenance ............... 70
   4.9.6) Iteration VI: Perform GUI .................................................................. 71
5. Project Planning and Cost ......................................................................... 71
5.1) Initial Planning ............................................................................................................. 71
5.2) Final Planning ............................................................................................................. 73
5.3) Project Cost Estimation .............................................................................................. 73
  5.3.1) Hardware Resources ............................................................................................. 74
  5.3.2) Software Resources ............................................................................................. 74
  5.3.3) Office Resources .................................................................................................. 74
  5.3.4) Human Resources ................................................................................................ 75
6. Conclusions ..................................................................................................................... 76
  6.1) Project Results .......................................................................................................... 76
  6.2) Personal Evaluation .................................................................................................. 76
  6.3) Future Work .............................................................................................................. 77
Bibliography ....................................................................................................................... 79
Acronyms ............................................................................................................................. 81
Appendix ............................................................................................................................... 83
Appendix-1 ........................................................................................................................... 83
1. Overview

1.1) Introduction

In Software Engineering industry has been growing amazingly during the last 2 decades. Currently everything that we use has a software and program inside. People start to trust automated machines in every moment of life. They started to use it in everywhere. However, to make these kind of technologies very trustable and usable, everything should be planned very carefully. Because when we talk about software industry, it does not only mean programming but also designing and planning. To build a program very efficiently, firstly software modelling should be done correctly and successfully. Even the last decade, there has been significant improvements in the industry, there are still some problems and shortcoming while using software modelling methodologies. If an error or a mistake is made during modelling, it eventually affects the software programming methodology, and finally it affects the real program because of the first mistake in modelling.

Of course software modelling is very big ocean and it is not possible know every edge of the ocean. In this thesis, the main subject would be UML/OCL conceptual schema. The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system [1]. It was founded by Grady Booch, Ivar Jacobson and James Rumbaugh in 1995. After continuing to develop the language it has become one of the most advanced modelling language in software engineering field. Currently it is such as a standard of software modelling and is nearly used in every software modelling techniques. The reason it became very popular was, any kind of activities in software modelling, interactions between software components, the relationships between entities and the user interface (external) can be described in a very easy way.

There are some exceptions that UML can’t define inside of it. For instance, in UML the rules for the elements can’t be defined. Nowadays it has become one of the most important factor in software modelling. [1]. Nowadays to define the rules in UML diagrams OCL [2] language is used and with using this one, in a UML diagram can define nearly everything in software modelling.

The one of the important factors in software modelling is build a very clear and errorless conceptual schema. Conceptual schema is a high-level description of a business's informational needs.
It typically includes only the main concepts and the main relationships among them. [1]. It includes two sections such as behavioral and structural knowledge. In the following chapters these two knowledge will be explained very deeply. For designing a correct software modelling these two knowledge have crucial factor and they should be done without and mistake. The structural part of the conceptual knowledge includes attributes; associations between classes and integrity constraints, that define conditions that the instances of the diagram must fulfill.

This document explains and shows the results of the elaboration of the master final project, devoted to the problem of the integrity constraint violations in OCL design of data. More specifically, the project addresses the integrity constraint problems in a given OCL contract and proposes an iterative solution for the violations which will be explained in detailed way in the following sections of the master thesis document.

1.2) Motivation

The first idea of the project came from the research done in the Department of the Services and Systems Engineering of the Polytechnic University of Catalonia. In the paper [3] authors Xavier Oriol, Ernest Teniente, Albert Tort presented a new approach for design and evolution some operations in Conceptual Schemas. The main purpose of this master thesis is to analyze this theoretical approach from practical point of view, and provide the implementation of the method in order to be able to evaluate the approach by performing experiments.

In addition to this, in the paper [4] authors Guillem Rull, Carles Farré, Anna Queralt, Ernest Teniente, Toni Urpí stated some validation methods in conceptual schemas. Also paper demonstrates interface for the validation. This paper also is useful for my thesis because the paper is so related with the thing that I would like to do and also it has the interface that I can inspire from. However by using these 2 paper it can be said that it is good enough to have an idea and the motivation for this master thesis.
1.3) Project Goals

The main goal of this project is the implementation of the integrity constraint detection method and try to fix these integrity constraint errors in an iterative way. The objectives of the project are the following:

1) Perform the implementation of Repair-Generating Dependency (RGD) by using Event-Dependency Constraints. The development consists of analysis and definition of the requirements, design, implementation and testing. It includes the design of the necessary data structures, choosing of the technologies to be used, the existing libraries for reuse etc., implementation of the algorithms and providing the user interface.

2) Perform the implementation of Integrity Constraint Violation Checking. The development of the software implementing the IC Violation Checking includes planning, analysis and definition of the requirements, design, implementation and testing. It includes the design of the necessary data structures, choosing of the technologies to be used, the existing libraries for reuse etc., implementation of the algorithms and providing the user interface.

3) Perform the implementation of Integrity Constraint Maintenance. After getting violations on step 4, the development of the software implementing the IC Violation Maintenance method includes planning, analysis and definition of the requirements, design, implementation and testing. It includes the design of the necessary data structures, choosing of the technologies to be used, the existing libraries for reuse etc., implementation of the algorithms and providing the user interface.

4) Perform the experiments of the implemented system. A suite of test cases
should be elaborated and the testing performed in order to evaluate the Violation detection and maintenance both in terms of the results quality and time spent for getting the solutions from the input data.

The requirements which need to be satisfied during the elaboration of the project are the following:

1) Familiarize with Conceptual Schema, UML/OCL and Integrity Constraint Before starting the implementation it is necessary to examine carefully and understand the Conceptual Schema. Also it is useful to familiarize with other works on the similar topics to get a broader view of the previous attempts of solving raised issues.

2) Document the Project. The work on the project should be documented and in the end the project report should be produced containing theoretical part as well as the description of the development process and the experiment results.

1.4) Document Structure

This document is a final report of the work performed during the elaboration of the master project. It is organized as follows:

Section 2 presents the State of the Art in the field of multidimensional design. First it introduces the main concepts of UML, OCL and Conceptual schema, later on the problem of the integrity constraint is defined more clearly and the approach to solve it are discussed.

Section 3 provides the detailed description of analyses and design decisions of SafeEx tool based on requirements.

Section 4 covers the development part of the project, including the description of the development process, the architecture of the application, the design of the data structures and the algorithms. The chapter also contains the overview of the technologies used for the implementation and the libraries and previously implemented packages, which will be reused.
Section 5 discusses the aspects of project planning and the costs related to the elaboration of the project.

Section 6 presents the results of the projects and the conclusions extracted from this work. The ideas for future work are also proposed.

2. State of the Art

This section first introduces the main concepts of Conceptual schema, later on the problem of the integrity constraint is defined more clearly and the approach to solve it are discussed.

2.1) UML

Unified Modelling Language is very strong tool to design a model in Software Engineering Area. As it is mentioned before the meaning of UML is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system. In these days to design a software project generally UML is used for modelling. Many things can be defined by using UML. In Figure-1 it can be seen an example of UML design as Component diagram, sequence diagram and use case.

As it is seen in the Figure-1 by using UML it is very easy to simplify a complex modelling. Also it increases the readability and portability of the program. What it is interested in this project is structural
schema part of UML which is class diagram and the relations. In the following Figure-2 it can be seen a class diagram example of the UML.

![Figure 2 – UML class diagram](image)

In Figure-2, it shows the relations between classes in UML representation. Each class has different attributes and relations.

Current technology of UML is in very high level. There are plenty of tools that automatically converts UML diagrams into code. For this reason, nowadays designing in UML is very crucial.

2.2) OCL

After having been used UML very widely the software modelling [5] techniques have been developed tremendously. In so much that, in some cases UML has become not sufficient to use solely. Because to define some rules in the software modelling, just using UML is not enough. That’s why a new language model has been proposed by IBM. Object Constraint Language (OCL) is a helpful and additional language to use in UML. It completes some deficient part of the UML. An OCL context defines a situation in which statement is valid or not. Also in OCL, coding syntax such as if, then, else are able to be used.
Using OCL with UML becomes a very powerful tool for modeling a software efficiently and in a correct way.

2.3) UML/OCL Conceptual Schema

An UML conceptual schema includes full definition of class diagrams and its relations between them. A conceptual schema can define everything that is needed for programming a software model. For instance in Figure-2 it can be seen all the relations among classes. As it is mentioned in Section 1 conceptual schema divides into 2 parts such as structural and behavioral parts. Structural part consists of classes, its attributes, the relations between classes, integrity constraints. To define and demonstrate integrity constraints generally OCL is used. The second part which is behavioral part includes operations which are needed by the system itself. In UML, these contracts are defined as pre and post conditions, and it implies in which situation condition should keep instances before and after of execution of the operation. These pre and post conditions of operations defined and described in OCL in a very detailed way.

2.3.1) Contract

Contracts demonstrate the situation of the condition. The conditions divide into 2 parts; pre-condition and post-condition. Pre-condition shows the initial statement of the contract, and the post-condition shows the expected result of the condition when the operation executed. In Figure-3

```ocl
context Meeting::checkDate():Bool
post: result = self.participants->collect(meetings) ->forAll(m | m<> self and m.isConfirmed implies (after(self.end, m.start) or after(m.end, self.start)))

context Meeting::isConfirmed ()
post: result = self.checkDate() and self.numConfirmedParticipants > 2

context Meeting :: duration (): Time
post: result = timeDifference (self.end, self.start)

context Person::numMeeting (): Nat
post: result = self.meetings -> size

context Person :: numConfirmedMeeting (): Nat
post: result = self.meetings -> select(isConfirmed) -> size
```

*Figure 3 – OCL contract*

In the second example it shows in which condition we can say that a meeting is confirmed. In this situation, to realize a meeting a result should have a date and also should be at least 3 participants,
without these 2 conditions meeting can’t be done, or to calculate the duration of the meeting the post-
condition should have start end time. If there is a pre-condition, it also should be defined in the contract.

2.3.2) Constraints

Actually integrity constraint means a violation in database when an insert, update, or delete
statement violates a primary key, foreign key, check, or unique constraint or a unique index. Generally
when a UML is designed and some instances are defined, cardinality constraints might be occurred. For
instance; in Figure-4,

Assuming that in the database there is an employee E, such that E manages a department D, and
is added in the database that E manages another department D2 without deleting the relationship
between E and D. In this case when instances are inserted in this surely that there will be a constraint
violation. Because an employee should manage at most 1 department. In this case, if E manages D
department and E wants to manage department D2, in this case for sure there will be a cardinality
constraint error. These theoretical approach will be explained in the following sections in a detailed way.

Another constraint problem might be integrity constraint problem which creates some error
while insertion of the instances. Actually it is nearly same thing with cardinality constraint but only
difference is it is done with instances, in this case there are some real instances like insertion or deletion,
and they causes the integrity constraint violations. According the paper [3], “An operation is executable
if there is at least one state of the information base in which its preconditions are satisfied and such that
the new state obtained from applying its post conditions satisfies all the integrity constraints”. It means
that in a contract, the post condition must satisfy all the integrity constraints.
2.4) Schema Conversion

While making a software modelling as it is mentioned before some specific languages are used such as UML and OCL. However these languages just increases the readability and portability of a software program. In order to use these types of software modelling, there should be used some conversions between UML/OCL and code. There are lots of tools that make the conversion and the current technology uses them a lot. Nowadays this UML conversion tools are a must. Because, once the UML design is completed there should be a tool that makes the conversion directly. The conversion might be different according the necessity of the project. There can be converted directly into code like; C++, Java or C# or there can be other methods like XMI, XML etc. The advantageous of using the converting tools is to create a basic skeleton of the code in a second and increase the readability of the project by anyone which is enrolled to that project.

- Pure Code conversion: Converting UML diagrams to code is difficult task, because while modelling a software everything should be defined in very clear way, a mistake that is made at the beginning of the project might affect the rest of the project cycles. While making a conversion some steps should take into account very carefully.
  - Cardinality relations between classes: Each cardinality represents a relation with another class. While defining cardinalities, all the possibilities should be considered.
  - Attributes: Each attributes have to be defined in a proper way. Unnecessary attributes might affect the future plan of the work. Also, defining attributes in wrong classes might affect the plan too.
  - Class types: The type of the class maybe the most important issue while designing the software. A wring type of class affects whole plan, and unfortunately it causes to return at the beginning of the project.

- Metadata conversion: Considering pure code conversion, metadata conversion is much simpler staff for UML/OCL. It just converts everything that is written in the
UML to another format. In Figure 5, it can be seen a conversion from UML/OCL to xml.

Figure 5 – Converted XMI file

It mainly includes everything a class has, name of the association, publicity, aggregation and multiplicity. These types of conversion generally used for checking UML model validation and correction. Also in this thesis this type of conversion modelling is used. In the following chapter it will be explained in detailed way.

2.5) Related Works

The problem has been addressed by a number of research works. In [3] the authors presented a comprehensive overview of the approach of fix the constraint problem.

The methodology can be divided into some parts:

- **Converting to UML/OCL into Logic:**

  Assume that currently there is a UML/OCL conceptual schema. The idea is proposed by in this paper [6]. In the UML, each integrity constraint is represented into logic one by one. According to the [6], all the integrity constraint results assumed as false in the beginning. Because the reason that if the user inserts or deletes the specified instances which are stated exactly in one logic, the program
should identify and catch an integrity constraint in that logic. It should be bear in the mind that the logic system is just a basic representation of the UML/OCL schema.

- **Getting Event Dependency Constraints (EDC) from the Logic:**

  Every constraint gets from logic of the UML/Schema. This logic generates lots of dependencies and these dependencies creates a plenty of varieties as a constraint. If we think about the dependencies, each dependency of the constraint presents different situation, and each situation presents the possible case of the constraint violation.

- **Getting Repair Generating Dependencies (RGD) from EDC:**

  Once all the EDC’s are received successfully, than EDC’s converted to the RGD’s. As it is understood from its name, each RGD represents the solution of the repairing EDC’s In other words, actually each RGD is got from EDC and each RGD a different representation of EDC but they explain how the problem can be solved by looking Event Dependency Constraints.

- **Fixing Integrity Constraint problem by using RGD’s:**

  To fix the integrity constraint problems, the algorithm needs all the RGD’s which is got from EDC’s. Once RGD building process finishes the program inserts or deletes the instances which is specified in RGD. Each RGD tells different instance problems, or some RGD’s might have errorless instances. What is needs to be done is, find RGD’s that proposes the solution for the problems, and apply the solutions by inserting the instances.

  This master thesis basically based on this paper. Because it represents the realization of the solution that proposed in the paper. Thanks to the Xavier Oriol and Ernest Teniente, they already have
great explanation for the understanding algorithms, problems and providing the solutions for these problems.

There is another paper [4], helps me to think about the interface of the SafeEx tool. It has also some great explanations about UML/OCL conceptual schemas and instances, it also has integrity constraint problem identification and during the creating API, some parts of the AuRUS tool had been looked for the inspiration.

3) Analysis of SafeEx Tool and Design Decisions

In this section requirements of the system will be explained and analyzed carefully. Requirement will be elaborated if they are required for the system (functional requirements) or if they required the type of the implementation which is non-functional requirements. After that, the design methodologies for the tool will be explained, then design, lastly the technology decisions will be explained in amplifiable manner.

3.1) Requirement Analysis

For the requirements a simplified template proposed has been used (Volere Requirements Specification Template, 2014). The sections of the template either are included or not presented in this project for the following reasons:

| ✓ Requirement Number | Unique number which identifies the requirement, so the requirement can be referenced. |
| ✓ Requirement Type | Represents objective number, which the requirement belongs to. It serves as a reminder of which objective this requirement relates to. |
| ✓ Description | A short description explaining the purpose of the requirement. |
| ✓ Reason | Justification of the requirement. |
| ✓ Fit Criterion | Useful approach to easily identify if the requirement is complete or not. It has mostly included for better understanding the intention and objective of the requirements that might confuse. Also helps to understand when the requirement is fulfilled. |
| × Customer Satisfaction/Dissatisfaction | Measuring the impact of the existence or absence of this requirement on the client. It is not included because it is not used in the process of choosing which requirements to implement sooner or later. |
| ✓ Dependencies | Impact of the requirement on the other requirements. In another words, keeping track of other requirements that have an impact on the requirement. |
| × Conflicts | Keeping track of other requirements that disagree with this one. In this project there are no such requirements that cannot coexist, so it is omitted. |
| × Supporting Materials | Pointers to the document that can help for understanding the requirement. In the current state of the project material support is still very low, so it is omitted. |
| × History | Dates of the creation and the modification of the requirement. It is not necessary to consider since the methodology does not take into account the dates. |

Additionally, a title that allows understanding quickly the purpose of the requirement is included.
In some requirements, in order to facilitate the contextualization and understanding its description and to allow locating the system better, the demonstrative image of the result is added following of the Fit Criterion section. Also dependencies section may vary according to circumstance can be included or not.

<table>
<thead>
<tr>
<th>#[Number])([Type])</th>
<th>[Title]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>[Description]</td>
</tr>
<tr>
<td>Reason</td>
<td>[Reason]</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>[Fit Criterion]</td>
</tr>
<tr>
<td>Dependencies</td>
<td>[Dependencies]</td>
</tr>
</tbody>
</table>

3.1.1) Functional Requirements

3.1.1.1) The system should get converted UML/OCL conceptual schema as an input

<table>
<thead>
<tr>
<th>#1</th>
<th>The system should convert conceptual schema into XMI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The diagram should be in UML/OCL conceptual schema. It has attributes, relations, instances and contracts. The diagram should have enough information to compute by the system.</td>
</tr>
<tr>
<td>Reason</td>
<td>A conversion tool necessary to compute the diagram in the system. Because as a graphical representation of the UML/OCL conceptual schema program does not understand necessary process that needs to be computed.</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>Validate if the system has a xmi tool.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>None</td>
</tr>
</tbody>
</table>
### 3.1.1.2) The system should translate XMI file

<table>
<thead>
<tr>
<th>#2</th>
<th>The system should translate XMI file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The XMI includes all the information that is got from schema. The metadata file is an input for Logic rules. Logic rules has the information about constraints of the specified logic diagrams</td>
</tr>
<tr>
<td>Reason</td>
<td>The conversion should be made because of the computing algorithms. The system works like logic computing system and each logic rule definition has the instance rules. The program needs to read as logic instead of metadata file.</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>The system should have logic rule definitions by using XMI metadata file</td>
</tr>
<tr>
<td>Dependencies</td>
<td>1</td>
</tr>
</tbody>
</table>

### 3.1.1.3) The system should be available for inserting the instances

<table>
<thead>
<tr>
<th>#3</th>
<th>The system should be available for inserting the instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The tool has stored instances inside of it. These instances will be used later if there is a constraint violation or not. Each instance is inserted and checked separately.</td>
</tr>
<tr>
<td>Reason</td>
<td>To have some results in the database part, the instance should be inserted to the system. Without inserting instances, the system can’t detect the constraint violations.</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>Successful instance insertion</td>
</tr>
<tr>
<td>Dependencies</td>
<td>1,2</td>
</tr>
</tbody>
</table>
3.1.1.4) The system should be available for inserting the Dependency Constraint Equations

<table>
<thead>
<tr>
<th>#4</th>
<th>The system should be available for inserting the Dependency Constraint Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The tool has stored instances inside of it. These instances will be used if there is a constraint violation or not.</td>
</tr>
<tr>
<td>Reason</td>
<td>To check these instances creates violation or not all dependency constraints should be inserted into system</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>Successful instance insertion</td>
</tr>
<tr>
<td>Dependencies</td>
<td>1,2</td>
</tr>
</tbody>
</table>

3.1.1.5) The system should be able to detect the integrity constraint problems

<table>
<thead>
<tr>
<th>#5</th>
<th>The system should be able to detect the integrity constraint problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The tool has stored instances inside of it. These instances inserted by the user. After insertion, a detection process works for the integrity constraint problems.</td>
</tr>
<tr>
<td>Reason</td>
<td>To identify problems and to understand reason of the violation, the system should send an alert to the user for the violation</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>If there is a violation, the system should catch it</td>
</tr>
<tr>
<td>Dependencies</td>
<td>1,2,3,4</td>
</tr>
</tbody>
</table>
### 3.1.1.6) The system should propose solution for the constraint problem

<table>
<thead>
<tr>
<th>#6</th>
<th>The system should propose solution for the constraint problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The finding the problems is not only purpose. If the system finds a problem, a proper solution should be proposed to the user by the system.</td>
</tr>
<tr>
<td><strong>Reason</strong></td>
<td>After detecting a problem, to get a solution the system should warn the user.</td>
</tr>
<tr>
<td><strong>Fit Criterion</strong></td>
<td>The system should propose a proper solution propose</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
<td>3, 4, 5</td>
</tr>
</tbody>
</table>

### 3.1.1.7) The system should solve the constraint problem

<table>
<thead>
<tr>
<th>#6</th>
<th>The system should solve the constraint problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Once system finds the problem, a proper solution is proposed to the user. The user is able to use the solution which is given by the system. The solution has to fix the constraint problem itself.</td>
</tr>
<tr>
<td><strong>Reason</strong></td>
<td>If there is a violation in the system, it should be solved by user with inserting related instance</td>
</tr>
<tr>
<td><strong>Fit Criterion</strong></td>
<td>The system should not have constraint problems after execution</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
<td>4, 5, 6</td>
</tr>
</tbody>
</table>
3.1.2) Non-Functional Requirements

3.1.2.1) The tool must be compatible with any kind of computer or laptop.

<table>
<thead>
<tr>
<th>#1</th>
<th>The tool must be compatible with any kind of computer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The usability and portability are very important issues in nowadays, the system should be compatible in any environment to get more usability and portability.</td>
</tr>
<tr>
<td><strong>Reason</strong></td>
<td>The system should be used in any platform, any computer, any OS without any error.</td>
</tr>
<tr>
<td><strong>Fit Criterion</strong></td>
<td>The system should work both Linux, MacOSx, Windows and any kind of computer.</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
<td></td>
</tr>
</tbody>
</table>

3.1.2.2) The tool must be compatible with Deductive Database System

<table>
<thead>
<tr>
<th>#2</th>
<th>The tool must be compatible with DES System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Datalog Educational System (DES) is a tool for deductive database system [7] but it can also be used to check integrity constraint problems.</td>
</tr>
<tr>
<td><strong>Reason</strong></td>
<td>The system should be compatible with DES in order to use the system like instance database.</td>
</tr>
<tr>
<td><strong>Fit Criterion</strong></td>
<td>The system should be integrated successfully</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
<td></td>
</tr>
</tbody>
</table>
3.1.2.3) The tool must be compatible with UML/OCL Schema Tools

<table>
<thead>
<tr>
<th>#3</th>
<th>The tool must be compatible with UML/OCL Schema Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The tool should be a UML/OCL designing tool. It can be also defined contracts in OCL and it can be extracted schema as xmi files</td>
</tr>
<tr>
<td>Reason</td>
<td>The system should be compatible with the files that is exported from schema tool (ArgoUML)</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>The file is created</td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
</tr>
</tbody>
</table>

3.2) Analysis of Architectural Decisions

3.2.1) Datalog Educational System (DES)

[8] Datalog Educational System (DES) is a free, open-source, multiplatform, portable, Prolog-based implementation of a deductive database system. It can be used for duplicate elimination, restricted predicates, integrity constraints, ODBC connections to external relational database management systems (RDBMSs), Datalog and SQL tracers, a textual API for external applications, and novel approaches to hypothetical SQL queries and Datalog rules, declarative debugging of Datalog queries and SQL views, test case generation for SQL views, modes, null values support, (tabled) outer join and aggregate predicates.

There are couple of reasons to use that tool in the system:

- The system is compatible with Java which means that the system can be integrated into the current tool.
• The system can get input as instances. This is another good property of the DES. Because when the system tries to enter an instance normally it should be inserted in a DB and it should be checked inside of it, but DES system does it automatically. The tool has a DB-like system and when the user inserts an instance, it directly stores the instance inside of the tool.

• It can compute the queries and tell the result according to inserted instance which means that if the user inserts instances to the DES system and runs a rule, after running it he/she is able to see the result of the query directly. It stores the instances and computes the query according to given instances.

Of course there are some disadvantageous of the system:

• The result system is not very well designed, sometimes getting results might be very complicated.

• The syntax is very strict. The accepted syntax hast to be in same format. For instance; instances should not be entered with capital letter, query variables can’t be lowercase.

• Yes, writing queries using not is quite tricky in the DES system because a query must always be safe. A query would be safe if and only if every variable appearing inside a "not (...)" also appears in a positive literal of the same query.

    However, coding all these properties from 0 is much harder than handle the disadvantageous of the system. That’s why in the tool, using the DES system would be best option.
3.3) Development of Platform and Technological Decisions

3.3.1) Java

It is clear that for that kind of tool an object-oriented programming tool should be used. To develop such kind of system actually there are two options C# [9] and Java [10]. However, the tutor of the project Ernest Teniente suggests that Java would be better option to build such kind of a program. Also, Xavier Oriol has some initial codes which were written in Java. That’s why Java is chosen as programming language tool for development phase.

There is another advantage is using java is that it is really usable with other tools which means instead of C#, Java has more compatibility with other tools.

3.3.2) Eclipse

Figure 6 - Java

Figure 7 - Eclipse
Eclipse [11] is a language Integrated development environment (IDE) for Java. The reason chosen Eclipse for the project is just about experience. Because in my previous work and study experiences I only used Eclipse for Java. That’s why it is chosen according familiarity.

3.3.3) ArgoUML

![ArgoUML](image)

*Figure 8 - ArgoUML*

ArgoUML [12] is a tool for modelling UML/OCL conceptual schemas. It is another tool for the project.

The good property of the ArgoUML is [13], it has an option to convert schemas to XMI file. This is needed in the project because the program should get XMI file as input. For this reason all the schema drawings are made in ArgoUML by Xavier Oriol and imported in XMI format. Of course there are plenty of tools that converts schemas to XMI but this one previously used by Xavier Oriol and it is fits with project perfectly.
3.3.4) Datalog Educational System

Datalog Educational System (DES) is a very powerful tool to detect integrity constraints for given instances. Previously, it is explained in Section 3.2.1. Because of the reasons that is explained in that section, the tool is used for the system.

4. Development of SafeEx Tool

As it is mentioned in previous sections, while developing the tool some decisions had been made. If we summarize the decisions; which technology has been used previously for detecting integrity constraints, which tools would be better while developing tool, which extra programs could increase the time efficiency of the program, descriptions of the program that is used in the tool are explained. It is important for the planning of the project because without measuring the cost, deciding the tools that will be used or investigating previous works that have been made, developing the tool would cost unnecessarily more. For the sake of the programmer and increase the efficiency all the steps until so far explained in the previous sections.

In this section more sophisticated part that will be explained. All the details about the coding decisions, algorithms, data structures, other software integrations will be explained intensively.
4.1) Code Reusing

In section 2, it is explained that the some parts of the project have already been developed by Xavier Oriol. There are plenty of classes inside of the library however there are some classes mostly are used among them.

There are some important points in this libraries that should be emphasized. For the project one of the most important factors is to convert the schema to xmi [14] files automatically. Because if the tool gets a schema directly it does not recognize anything. For this reason, first a proper schema should have in some other tool. Also to convert the tool into the logic is another problematic issue.

In the library there are some methods that have already been coded for logic conversion and constraint results. There are some missing parts in the libraries and theses added later into the classes’ afterwards. Also there are some methods that was taken as an inspiration. For instance, some methods were doing other jobs however the process should be done in the thesis was little bit different. For this reason some methods has been got as reference and modified in another method like it is a completely new method. However reusing the code has made some disadvantageous for the tool. For instance; there are some checkpoints in the project, and to pass the checkpoint some previous works should be analyzed and understood carefully. Nevertheless the code has been written perfectly, understanding the development part of the whole library took 2, 3 weeks. To be sincere the first times it was too difficult and tough to analyze whole code in a short time but afterwards by using the classes and methods it has been got used to it and it accelerates the pace of the project and thesis plan tremendously. The following diagram it can be seen sample class relationships in used code;
4.1.1) UML/OCL Schema Conversion

In this section how the schema conversion is done will be explained. Also it will be mentioned why it is converted into a different form. As it is known, to draw a UML class there are some tools are used widely. Developers prefers to use these kind of tools because it converts the schema into the coding or metadata version.

In the following figure it can be seen a schema with UML/OCL.
This is a normal UML diagram with some OCL contracts. To draw this diagram ArgoUML tool has been used by Xavier Oriol. When the schema is drawn, there should be done a conversion into xmi file. Because as previously explained, solely schema does not mean anything for the tool. There should be something that could work and convert on it. ArgoUML is one of the tools that can make the conversion between schema and xmi file. For this reason to get the conversion, the tool is used.

When it is drawn a schema such as Figure-11 in ArgoUML, what is need to be done is just export as an xmi file. In the ArgoUML there is an option that does the conversion automatically.

In the XMI files there are all the classes, relations, attributes, cardinalities and contracts.

4.1.1.1) Class Representation in xmi File

Each class in xmi should be represented separately. According to the Figure-12

```
<XML:Class xmi.id = '5-84-84-25--6F0387db:13b8a24d1ba':8000:0000000000000866' name = 'Physician' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'/>
```

*Figure 12 – Class representation in XMI*
It is the Physician class. In the description, it includes a unique id for the class, the name of the class, the type of the class (public, private etc.) etc. In xmi file each class is represented like Physician class. Also if the class is an abstract class or if it has some other properties it should be defined in the xmi as well.

4.1.1.2) Parameter Representation in xmi File

Parameters also defined in xmi file. It is more or less same representation with class. Like in class there are some properties are shown in parameter part such as; id, name and type.

![Parameter representation in XMI](image)

4.1.1.3) OCL contracts Representation in xmi File

Unlike others the representation of the OCL is quite similar with original version in the code. Because the contracts are always described with words and there is nothing to convert it into the letter. It only includes the id, the name of the context and expression.

`<UML:Comment xmi.id = '5-84-84-25--6f0387db:13b8a24d1ba:-8000:00000000000000887' isSpecification = 'false' body = 'context MedicalTeam inv DirectorIsMember:
self.member >includesAll(self.manager)

context Physician inv SpecialistOfTeamsExpertise:
self.specialization >includesAll(self.team.expertise)

context MedicalTeam inv UniqueMemberShip:
not(self.oclIsKindOf(CriticalTeam)) or self.member.team-&gt;forAll(t|t = self)'/>

4.1.2) XMI Converter

XMI file is one of the input of the tool. Basically it is the first file to start everything in an order. Because once the file is inserted to the system it is converted into Event Dependency Constraints. To do it in automatic way firstly the xmi file should be included into project.
To get the conversion from xmi file to Event Dependency Constraints, the external library which is given by Xavier Oriol is used. UMLToEDCTranslator class basically gets the xmi schema and analysis inside of the file and converts to the schema into EDC diagrams. Because of having external libraries the operation is done automatically, and after that part, it can be said that the schema is converted to the Event Dependency Constraint.

4.2) Event Dependency Constraint

Event Dependency Constraint (EDC) is a representation of the possible violation of the schema. To put in a different way, when XMI is converted into the EDC, the program starts to get many EDC constraints. In the article [3] the main idea of the creating EDC constraint, and insert event rules to this to a deductive database to check if there is an integrity constraint or not. More basically, an EDC decides when a violation does occur while inserting instances. For instance the following lines includes examples of EDC.

@99 :- not(Manages(P0,MT0)), ins_Manages(P0,MT0), not(Physician(P0)), not(ins_Physician(P0))

@101 :- not(Manages(P0,MT0)), ins_Manages(P0,MT0), Physician(P0), del_Physician(P0)

@103 :- Manages(P0,MT0), not(del_Manages(P0,MT0)), Physician(P0), del_Physician(P0)

@105 :- not(Manages(P0,MT0)), ins_Manages(P0,MT0), not(MedicalTeam(MT0)),
       not(ins_MedicalTeam(MT0))

@107 :- not(Manages(P0,MT0)), ins_Manages(P0,MT0), MedicalTeam(MT0), del_MedicalTeam(MT0)

@109 :- Manages(P0,MT0), not(del_Manages(P0,MT0)), MedicalTeam(MT0), del_MedicalTeam(MT0)

As it is seen from the code, each EDC has a unique identification number and each EDC has different rules. Reading EDC’s might be complicated. For this reason it is better to demonstrate an example for explaining working principle of EDC and real meaning of it. Let’s assume that there is an employee named as E. He works at a department, at the same time he is the manager of the department D and he starts to manage another department which called as D2. However he wants to do it without giving up the management of department D. However this is not possible and E can’t manage two departments at the same time.
According to the example we should have the following Event Dependency constraint.

\[
\text{Employee}(E), \text{Manages}(E, D), \text{ins\_Manages}(E, D2), D<> D2, \text{not (del\_Manages}(E,D)) \rightarrow \bot
\]

It means that if there is an Employee E, if he manages a department D and if he wants to manage another department D2 and if he does not give up managing of department D, and if D and D2 departments are not equal, there is a problem. The problem is an employee can’t manage 2 departments at the same time.

Other thing is about the equality of the EDC. All the EDC’s are equal to not (true) value. Because when the given example is thought, it can be concluded that, if all of them assumed as true there is a problem occurred on the given EDC. If we think about example again, if we delete the management of first department, there won’t be any problem in the EDC. In other words, if one of them affects the false then EDC might detect a problem.

4.3) Literals

Literal is each object that can be seen in the EDC. If we think about the same example again;

\[
\text{Employee}(E), \text{Manages}(E, D), \text{ins\_Manages}(E, D2), D<> D2, \text{not (del\_Manages}(E,D)) \rightarrow \bot
\]

It would be concluded that there are 5 literals in this EDC such as; Employee(E), Manages(E,D), ins\_Manages(E,D2), D<>D2, not (del\_manages(E,D)). Basically literals includes two types;

4.3.1) Ordinary Literal

Ordinary literals include all the literal types except the one that demonstrates not equality. Again if we think about the same example;

\[
\text{Employee}(E), \text{Manages}(E, D), \text{ins\_Manages}(E, D2), D<> D2, \text{not (del\_Manages}(E,D)) \rightarrow \bot
\]

Ordinary literals are;

Employee(E)

Manages(E, D)

ins\_Manages(E, D2)
not (del_Manages(E,D))

which means that everything that is related to insertion(ins), deletion(del) or description(everything without ins and del) would be counted as ordinary literal.

4.3.2) BuiltIn Literal

Build in literal simply shows the equality of the instances;

D<>D2

4.5) Instance

Instance is one of the most important issue for this project. Because instances decide everything. There might be tons of instances in the tool and each instance could represent a different thing. Instances are defined by the user. If we think about the literals it would be seen that there are some values but these values are not real examples. Basically an instance is an exampled version of the literals. If we think about the same example that is given for ordinary literals;

Employee(E), Manages(E, D), ins_Manages(E, D2), D<>D2, not (del_Manages(E,D)) -> ⊥

We see that there are literals both ordinary and builtin. These literals can be real example for the instances. For instance for each literal let’s give examples;

Employee(John),

Manages(John,HumanResources),

Ins_Manages(John,Accounting)

not(del_Manages(John,HumanResources))

in the examples what is done is just exemplified the literals which means that instances are real examples of the literals and it can be used as many as possible.

4.6) Detecting Integrity Constraint Violation

There are two important tasks in this project. One of them is checking integrity constraint violation. To check integrity constraint violation some tools are used. In addition to that, a data structure and connections between tools are created. In this section all of them will be explained step by step.
4.6.1) Getting all EDC

To check integrity constraint there are some inputs that have to be ready before. One of them is to obtain all the integrity constraints before starting. To get all of EDC’s there are some development techniques used. Simply to call all of the instances there should be used following 3 coding lines.

    UMLToEDCTranslatorController translator = new UMLToEDCTranslatorController(new File("schema.xmi"));
    translator.translateSchema();
    translator.getAllConstraints()

As it is explained before these 3 lines get the xmi file from converted schema translate it into the logic then get all the constraints. Then it should be inserted all the constraints into EDC. All the EDC’s are stored in an array, and the array iterates each of the EDC one by one.

4.6.2) Obtaining DES Connection

After getting the EDC’s, the program should insert the EDC’s into the DES system. However the DES system is quite complicated for using with Java and EDC. In this section only connection of the DES system will be explained.

DES system is a deductive database system for educational purposes. In the beginning of the project the tutor Ernest Teniente mentioned about the tool and he proposed a integrity constraint violation solution with the using this tool. However the situation was not clear, because DES system is really complicated to integrate and use it with Java. DES is a deductive database system which means that it is a system that can make a deduction by getting input as a rules and facts and the results stored in database. The main syntax is Datalog syntax for implying facts, rules, queries and instances into the database. If a database needs to correlate with logic, DES is one of the best options in the field.

To get connection to DES inside of the development, there should be done some arrangements. DES system is downloadable from the website of Prof. Fernando Saenz Perez. Firstly the tool should be downloaded from there [15]. After downloading the tool, it should be included the development
environment. The tool has some definitions and API explanations inside however what the necessary thing for the development is the DES itself. As it is seen in the following figure. DES system included into the development environment.

To get connection into DES system there are some rules have to be followed. Firstly, we have to be sure that if the program is connected to DES system or not. Java-DES connection is quite simple. In previous sections, it has been mentioned that to get the connection firstly, the files should be inserted into the project folder. This can be seen in Figure-13. When the folders is put into the program directory, it means that the system can be connected directly by using it. However to get the whole connection in Java some methods should be created and called. The first created method is DESController which checks the path of the DES.

```java
public DESController(String path){
    pathToDes = new File(path);
    assert pathToDes.exists():"Path "+path+" does not exists!";
}
```

When the file is located, it should be called in necessary place such this command.

```
DESController desController = new DESController("des/des.exe");
```
Secondly other method which is connect should be called. In this method what is done is just get the path from the user put the path to get the connection. However there is an important issue that in the DES system the predicate warnings are closed. Because in nearly each query insertion the some warnings occur and it affects output syntax of the program. To get rid of this, warnings are closed before getting output results.

```java
public void connect() throws IOException{
    ProcessBuilder pb = new ProcessBuilder(pathToDes.getAbsolutePath());
    Process proc = pb.start();
    inputStream = proc.getInputStream();
    outputStream = new PrintWriter(new BufferedOutputStream(proc.getOutputStream()));
    readHeader();
    this.applyCommand("/undef_pred_warnings off");
}
```

As it is explained, what is need to be done is get connection between system and des.exe. After doing that the tool should be using des system without opening and entering it. To get the connection DES class is created in the development environment. The class has methods for getting connection, getting results from screen, inserting queries and instances. When the program is connected the tool, it is informed in the result windows like following;

![Figure 15 – DES main page](image)

When the screen comes to the result windows, it can be concluded that the system successfully connected to the system and waits for some commands or inputs to process.
4.6.3) EDC Syntax Correction for DES

DES is a quite complex system when it is wanted to be used with another thing. Because inside of the DES there is a builder called Prolog, and it computes everything related with logic and deductive database. For this reason this system has its own syntax order and while inserting the EDC’s into DES system the syntax should be changed according to specified format. Normally an EDC syntax is a following way.

\[ \text{not(Manages(P0,MT0)), ins_Manages(P0,MT0), not(Physician(P0)), not(ins_Physician(P0))} \]

If the following EDC is inserted in that way for sure there would be a syntax error in DES. To correct the syntax there should be developed some methods inside of the development. The following corrections should be made into development phase to insert it into DES System;

4.6.3.1) BuiltIn Literal Syntax Correction

BuiltIn Lateral has already been explained in previous sections. First problem in the syntax is to correct builtin literals in EDC. Normally a BuiltIn syntax in EDC represented like following;

\[ \text{not(Manages(P0,MT0_1)), ins_Manages(P0,MT0_1), Manages(P0,MT0_2), not(del_Manages(P0,MT0_2)), MT0_1<>MT0_2} \]

However while inserting the DES, it returns an error with the syntax. To correct it the syntax changed in following way.

\[ \text{not(Manages(P0,MT0_1)), ins_Manages(P0,MT0_1), Manages(P0,MT0_2), not(del_Manages(P0,MT0_2)), MT0_1\neq MT0_2} \]

A method was created in the developing stage and changed the syntax in specified way. Only problem is to put one “\" in front of the “=”, there should be put four “\\\" in front of the “=”. In the code it is represented like \text{replace("<>", "\\\\=")}. After changing the BuiltIn Literal syntax all the builtin literals works succesfully.
4.6.3.2) Capital Letter Problem

In normal EDC, all the literals have some capital letters, so these letters can’t be recognized in DES. For this reason, all the letters with capital in EDC’s changed with small letters.

Before changing:

not(Manages(P0,MT0_1)), ins_Manages(P0,MT0_1), Manages(P0,MT0_2),
not(del_Manages(P0,MT0_2)), MT0_1\=MT0_2

After changing:

not(manages(P0,MT0_1)), ins_manages(P0,MT0_1), manages(P0,MT0_2),
not(del_manages(P0,MT0_2)), MT0_1\=MT0_2

The literals having capital letters are changed before inserting the des system, and system only recognizes the literals with small letters. On the contrary arity part of the literals which are the part that is in the parenthesis is accepted in capital letter so, there should not be any changed for arity.

4.6.3.3) Final Corrections in EDC

Even there are some syntax changes in EDC, it is still not enough to insert EDC’s into DES system. There are still some problems on that. As a reference let’s take the modified EDC from section 4.6.3.2.

After processing 2 operations now EDC is ready as syntax however there should be add something to be understood as query in DES. Because to insert the DES the EDC has to be like a Prolog query. The last version of EDC has current changes;

- BuiltIn Literal Syntax Corrections
- Change literals from capital letter to small letter

Now there should be one more process for EDC to be ready for inserting into DES. To behave EDC as a query there should be added following letters in each EDC.

query(P0,MT1,MT2) :- not(manages(P0,MT0_1)), ins_manages(P0,MT0_1), manages(P0,MT0_2),
not(del_manages(P0,MT0_2)), MT0_1\=MT0_2

Only change is to add query(P0,MT1,MT2) :- in front of EDC. Because when an EDC inserts into DES, it should be get as a query so to change the EDC type as query, it should be added in front of the
EDC. The challenging part here is put the same arity inside of the query. Because the variables in query should include the variables of the EDC. After processing these 3 sections it can be said that and EDC is ready to insert.

4.6.4) Instance Syntax

Instances are other important figures while checking the integrity constraints. Before inserting the all EDC’s into DES, instances should be inserted into it. However inserting instances is another challenging issue. There are some rules to insert instances.

- The instance which will be inserted should be small letter.
- There should be known the variable numbers of the instance.
- Variables which will be inserted should be small letter.

Instances generally seem like literals. Because each literal is actually not-parameterized instances. For this reason when inserting instances into DES, the instances could be thought as ordinary literals. The structure of the instance has 2 main body.

- Name of the instance
- Variable names of the instance

The syntax of the instance is: name(variables(variables)*)

It means an instance should be a name and include at least one variable. In addition to this, all instance should be small letter. If a user inserts a instance in that format DES system would accept instances.

4.6.5) Inserting Instances into DES

In DES even the syntax is correct insertion is not easy as it is thought. Firstly there should be open a connection. Then, put a necessary letters in front of the instance then insert variables. If everything is in correct way, the system counts instance as input.

To insert instances into DES there is a special command which is called “/tapi /assert”. While inserting each instance, in the beginning of the instance assert should be written. In the final, the syntax of the instances for DES should be such as;
For the SafeEx to insert the tool following inputs expected from the user;

The program first asks the number of instances, then the user inserts the name of the instance, finally user inserts variable names. If everything entered in expected format an instance would be inserted into DES.

4.6.6) Inserting EDC into DES

In first impression inserting EDC into DES system is easy. Because it can be thought that all the syntax conversions have been made before. However even all the syntax is correct there is another problem for insertion. The name of the problem is unsafe variables. For the first time it is really hard to understand it.
4.6.6.1) EDC Insertion with Unsafe Variables into DES

It is one of the challenging issue for inserting EDC’s into DES is unsafe variables. When program starts to insert an EDC into DES the program check the variables of the instances. In DES negated literals because negated literals have some restrictions. When an EDC wants to insert a query including not values into DES there can be two options.

- The values of the negated literal also have in positive literal.
- The values of the negated literal only unique values inside.

For instance in the following query the problem can be demonstrated more clearly;

```
query(G0, U0) :- not(ownedby(G0,U0)), ins_ownedby(G0,U0), group(G0), del_group(G0)
```

In this example there is only one negated literal which is not(ownedby(G0,U0)) When we insert the query into DES system we would get the following error:

Error: not ownedby(G0,U0) might not be correctly computed because of the unrestricted variables: [G0,U0] !

This means that the negated value does not have any positive value in the EDC. In other words, not(ownedby(G0,U0)) literal is a negated literal and to be safe for these negated literal in this EDC it should have also positive literal of the one that have negated literal. If the EDC could have positive literal of the negated literal the EDC would have computed without any errors.
As it is seen from the Figure-17, with positive value of negated value the query is computed without any error.

However adding positive literal of the negated literal can’t be a solution. Because if a positive literal adds into the EDC the query is corrupted. It successfully computes in DES, but it also affects the meaning of the query. That’s why this solution is rejected by the system itself.

To solve the problem there is another method. Instead of inserting EDC with negated literal, that negated literal could be changed. The algorithm works in the following.
- An EDC is inserted into DES
- Compute EDC in DES
  \[
  /\text{tapi query}(G0, U0) \leftarrow \neg(\text{ownedby}(G0, U0)), \ \text{ins\_ownedby}(G0, U0), \ \text{group}(G0), \ \text{del\_group}(G0)
  \]
- Get computed result from DES
- In Case of error:
  Error: \(\neg \text{ownedby}(G0, U0)\) might not be correctly computed because of the unrestricted variables: \([G0, U0]\)!
- Write another query for unsafe literal
  \[
  /\text{tapi ownedbya} \leftarrow \text{ownedby}(G0, U0)
  \]
As it is seen the variables of the literals is hided in another query. By doing this, the EDC is able to get rid of the problem of unsafe variables.
• Change unsafe variable name with computed query name in EDC

It means that whenever and ownedby(G0,U0) literal is seen in that EDC, the algorithm should work and change it with ownedbya. The new query should seem like;

query(G0, U0) :- not(ownedbya), ins_ownedby(G0,U0), group(G0), del_group(G0)

• Compute changed EDC in DES.

/tapi query(G0, U0) :- not(ownedbya), ins_ownedby(G0,U0), group(G0), del_group(G0)

• Do it recursively until not getting a unsafe variable error for a DES.

By using this algorithm for every EDC all the queries are be able to computed successfully and without and error. This part is the most challenging part of the EDC insertion and with this simple algorithm the solution worked perfectly for the tool.

4.6.7) Detecting Integrity Constraint Violation

After completing all of the syntax corrections, EDC connection, instance insertion and EDC insertion with unsafe variable checking, the program would detect integrity constraint violations. Basically detection is done by DES. DES is very strong tool for integrity constraint detection. When the user inserts the instances as an example the system starts to keep in the deductive database for the checking, the user is able to insert one or more than one instances in the beginning. The instances is inserted into the system one by one. Every instances addresses different literals. However there might be same instances with different variable names too. After completing instance insertion the program starts to check EDC’s one by one. In other words, first program gets the instances then check in one EDC. If there is a violation constraint problem it warns to the user, if not it goes to the other EDC. The following figure demonstrates algorithm more demonstratively;
The most important thing in this process is checking EDC’s one by one. Because all of the EDC’s is inserted in one time, in that case it is not possible to be understood the violated EDC. Because while processing integrity constraint checking, the algorithm checks the instances with EDC and if it is an error warns to the user. If we think about a simple example on this algorithm, assuming that we would like to insert an instance which is \( \text{ins}\_\text{manages(john, team1)} \) and we would like to check each this insertion with each EDC. When the instance insertion is completed, the algorithm starts to insert one EDC. If we think about such an EDC; \( \text{not(manages(X,Y))}, \text{ins}\_\text{manages(X,Y)}, \text{not(member(X,Y))}, \text{not(ins}\_\text{member(X,Y)}) \) it says that if the john wants to manage team1, first he should not manage team1, he should not be member of team1 and he should not be inserted a member of team1. However when \( \text{ins}\_\text{manages} \)
inserted it directly creates a problem on this EDC because of intends to manage team1. When the problem occurs, the program warns the user and says that there is a problem in this EDC and there is an integrity constraint violation.

4.7) Finding Problematic Instance in EDC

Integrity constraint detection is a great milestone for the work. However there are some more important parts. Just detecting the integrity constraint is be able to be a solution. There are some other processes should be done after detection.

When program detects an integrity constraint violation, the second important issue is to find problematic instance in the EDC. Because if we think about an Event Dependency Constraint, it might not be able to find the instance which creates the integrity constraint violation. To find the problematic one the process goes in the following way;

Firstly, the program detects an integrity constraint problem in the EDC. All the processes are explained in Section 4.6 very clearly and step by step. After, detecting the integrity constraint problem, the program starts to find the root of the problem. In other words the program starts to search which instance is caused integrity constraint detection. To find it, there is a clear solution. In the first step, because of having the violation, the program has a result from the query. Because having a result means there is an integrity constraint problem. Let’s assume that we inserted an instance and there is an error in the following EDC:

Instance-> ins_sends(john,hello),

EDC-> query(U0, M0) :- not(sends(U0,M0)), ins_sends(U0,M0), not(user(U0)), not(ins_user(U0)),ins_message(M0)

As it is seen, EDC has ins_sends(U0,M0) literal, it means that in that EDC there is an integrity constraint error because of ins_sends.

There is a integrity constraint violation in ins_sends(john,hello) to understand which constraint has the violation the program does two checkings:
- It checks if the inserted instance is in the EDC: If the EDC has one of the instances that is inserted by the user. If there is, there might be a violation because of the existence.

- It checks the results of the EDC: It means that the program looks the result of the EDC and it matches the result instances with the instance that user insert.

When we come back to our example if it is taken a look more carefully the user has inserted ins_sends(john,hello) and the EDC has this literal too. The program starts to suspect for integrity constraint violation. Later, it computes the EDC. It checks the result of the EDC. In this example, when the EDC is sent to the DES, the result would be the following;

```
query(U0,M0) :-
    not sendsa,
    ins_sends(U0,M0),
    not user(U0),
    not ins_user(U0).
```

$ 'john'
'hello'
$eot !!

There is a integrity constraint violation in ins_sends(john,hello)

The first part of the query is for the EDC, after “$” sign it can be seen the result of the query. As it is seen from the example, there is an integrity constraint violation. Like it is explained, the program first looks for intersection between instance and EDC then checks the result, if there is an intersection
between the result and the variables of the instance. If the program passes from both checks, then it would be said that, there is an integrity constraint violation because of inserted instance. As it is seen, in the example, the program demonstrates the error and it emphasizes which constraint is caused a violation.

In this part the challenging issue is, there are some instances that having same variables with other instances. For instance; in the example ins_sends(john,hello) is inserted as an instance into DES, however user might insert more than one instance. If the user inserts ins_message(hello) now there is an intersection between instances too. In this case the following algorithm works, the program looks to the EDC, instance intersections. If it detects in intersection in the name, the program passes into second step. In the second step it looks to the result. The most important point is that the result should match exactly same as the variable of the instances. For example in the example the result was ‘john’ ‘hello’ and in the instances that user inserts are ins_sends(john,hello) and ins_message(hello) then program starts to check each instances if it is exactly matches or not. When the program checks ins_sends literal, it matches each variable with the results and concludes that the result matches with variables of the instance. When it comes to second one which is ins_message(hello), the first part of the result is not be able to match with it. Because the first part of the result includes john. However the second instance only has hello. By matching results and the variables of the instance the program could warn the user that there is an integrity constraint violation because of that instance or not. By using this rule, now it can be said that after detecting the integrity constraint, the program finds instances which create problems for EDC.

4.8) Integrity Constraint Maintenance

There are 2 main steps of the idea of this thesis. First, detecting integrity constraint violations. Second, fixing integrity constraint violation problem. In this section second one will be explained intensively. The workflow for the integrity constraint maintenance explains in the following diagram;
In this section each part of the algorithm will be explained and some examples will be given.

4.8.1) Converting Event-Dependency Constraint (EDC) to Repair-Generating Dependency (RGD)

Before starting fixing operation, there are some steps that needs to be done by the program. When the program finds an integrity constraint violation, the first step should be done is to convert EDC to RGD. RGD is made up by literals. To convert EDC to RGD there are some rules that should be followed.
First of them is when an integrity constraint violation has to be detected in EDC. This is very important. When user inserts some instance into DES the program starts to check each EDC, if there is an integrity constraint violation or not. When a violation is detected, program starts to make a conversion from EDC to RGD. RGD is a different type of EDC but RGD has crucial property for the program. It has the solution of the integrity constraint violation. In this section, how can the EDC be converted into RGD will be explained. An RGD has two sides. Right and Left part. As we know in EDC, there is only one side. The process of conversion have 4 steps;

- Check all the negated literals in EDC
- Check the literals having negation has insert or delete inside of it
- If there is, move it to the right part of the equation
- Delete negation symbol of the instance where is located in the right hand side of the equation
- If not put in to the left part of the equation.

In the following example, the whole process could be demonstrated more clearly; assuming there is an EDC;

```
query(CG0, M0, P0) :- issendto(M0,CG0), not(del_issendto(M0,CG0)),
message(M0), del_message(M0), not(group(P0))
```

As it is mentioned the first step is to check all negated literals. In this EDC there are two negated literals such as;

```
not(del_issendto(M0,CG0)) and not(group(P0))
```

The second step is to check inside of negated literals. In the inside of it, there should be an insertion or a deletion. When it is checked, it would be concluded that not(del_issendto(M0,CG0)) had a deletion inside of it. Now that one should be sent to right part of the equation in RGD. not(group(P0)) can’t be sent to the right part because it does not have neither insertion nor deletion inside of it. When we get the negated literal with insert or delete, there should be done one more step before putting into right hand side of the equation. While changing the location of negated one, the negation should be deleted. In other words, when not(del_issendto(M0,CG0)) is inserted into right hand side of the
Now is crucial part of the solution. What does this RGD mean? Why the negated insertion/deletion is put into right hand side of the equation. The answer is simple which means that if there is a message, and this message sends to a group conversation in whatsapp, if user wants to delete the message, the program should do following execution; when the message is deleted, also we have to be sure that the message that sends to group should be deleted from the data. Without deleting it from the group it can’t be said that the message is deleted successfully.

According to given example, the definition of RGD can be made. RGD creates a solution for integrity constraint by changing some rules in EDC. When user inserts an instance and that instance violates an integrity constraint in an EDC, if it is possible; EDC should be converted into RGD and it should be put into right hand side of the equation without negation symbol. By doing that the program could find a solution to the EDC’s which have integrity constraints.

4.8.2) Instance Insertion to fix Problem

When there is an integrity constraint problem and RGD is created by using EDC, now the program could go one step further. This step is called instance insertion to fix problem. If an EDC has integrity constraint violation and the program does a simple step. As it is explained in 4.8.1 it converts EDC to RGD. Then it checks right part of RGD. If there is a right part of RGD, program understands that to fix the integrity constraint problem, according the number of literals in the right hand side, one or more than one instances should be inserted into DES. In this section the important thing is deciding the variable of instance that violates the integrity constraint. In other words, the assuming that the program has an integrity constraint violation and the literals which is caused integrity constraint violation is for instance; ins_x(a,b). In the EDC the literal looks like ins_x(U,M). In this case the result should include a and b. When the program has the result as ‘a’, ‘b’, it starts to take a look at in RGD. If there is a literal in the right hand side of the RGD, the user should do the following.

- Take a look at right part of RGD
• Look at the literal that is caused to integrity constraint violation
• For that literal, look at the variable names in EDC
• Insert instances that matches with the variables of EDC

In other words, let’s assume that in the right hand side of RGD there is a literal which is called \text{ins}_y(U). In this case the program should insert that literal to fix integrity constraint violation. However the problem is the literal which will be inserted into EDC should have a related variable with the literal that makes integrity constraint.

In the given example, the variables of problematic constraint is \text{ins}_x(U,M). The name of the variables are U and M. In the right hand side of RGD has \text{ins}_y(U), which means that the name of the variable is U. Now the user understands that, to fix the integrity constraint, he/she has to insert the variable which is related with U. When the user gets an integrity constraint error from \text{ins}_x(U,M), the results are ‘a’, ‘b’. In this example it can be understood that in the beginning of the program user inserted a, b as variable of instance. In this case, the variables matches with the following instance variables. U is related with a and M is related with b. By having this information then if the user investigates right hand side of RGD he/she could see that there is an intersection with U variables in both literal, and we already know that that U actually represents ‘a’ in integrity constraint violation. At the end the user has the following information;

• The variables which is caused to integrity constraint violation
• Solution to fix integrity constraint violation by using RGD
• Variable intersection with right hand side of the RGD and EDC

If the user has these 3 elements, he/she has to enter the literal where is located int right hand side of the RGD with intersected variable. In given example, it should be \text{ins}_y(a). Because \text{ins}_y is the solution of the constraint violation problem and and the intersected variable in both literals is U. When we look at the result of the U in the integrity constraint violation result, it corresponds a. By having these information the necessary solution for the integrity constraint violation should be fixed by inserting \text{ins}_y with a variable.
4.8.3) Validation Solution of Integrity Constraint Violation

Inserting right hand side of the RGD does not mean the problem is fixed. To validate if the error is fixed or not there are 3 steps.

- Insert the related instance into DES
- Run the EDC again
- Check the result if it has been fixed or not

In the 4.8.2 the instance insertion is explained but to validate the solution is correct or not there should be done 2 more things. The first thing is insert the EDC query into DES system again. If the user is inserted right hand side of RGD, now it is time to insert EDC again. Because as it is known that RGD’s are different version of EDC. An EDC explains that in which condition the error happens, in RGD demonstrates that what is the solution of that EDC if there is an integrity constraint violation problem. After inserting instance of the right hand side of the RGD, the program automatically calls the corresponding EDC. That EDC should be same as the EDC which has the integrity constraint violation. The reason of calling that query is, the program has to know that if the solution completed with successfully or not. To understand that the EDC should be checked again with new instance. So, after inserting EDC there is two possibilities. First option is the solution worked and it fixed the problem successfully. Second one is the solution does not work on corresponding EDC. The test results show that %99.9 percent of the solution is worked on corresponding EDC. Actually there is no other option, if the solution does not fix the problem it means there is a problem in schema conversion or there is another possibility that the right hand side of the RGD could be more than 1 value. In this case the error of EDC might be caused in other literal of RGD and everything should be deleted which is related with other insertion of RGD.

4.8.4) Additional Integrity Constraint Problems

When the user inserts an instance, because of having integrity constraint problem in another EDC, that instance could violate another integrity constraint in another EDC. To fix that problem that problematic EDC should be converted into RGD and the right hand side of the RGD should be inserted of the system and the violation should be checked again if it is fixed or not. In the following diagram the process is explained definitively.
The problem occurs because of the new instance that comes from RGD instance. Because when the user inserts a new instance into DES, and that instance fixes the problem of EDC there are some problems occurred in other EDC’s. Assume that we have the following schema for finding the integrity constraint problems;

Figure 21—Fixing additional integrity constraint problem
This is the simple schema example for whatsapp application for mobile phone. In the first step of the program the user enters following instance.

\[ \text{ins\_sends(john,hello)} \]

That means john sends a message and the message is “hello”, this is the first thing that will be inserted into DES system. After processing EDC’s there is a problem occurred one of the EDC.

\[ \text{query(U0, M0) :- not(sends(U0,M0)), ins\_sends(U0,M0), not(user(U0)), not(ins\_user(U0))} \]

The result is:

‘john’
The result says that there is an integrity constraint violation in this EDC and it should be fixed by using RGD. The RGD of following EDC is

\[ \text{not}(\text{sends}(U0,M0)), \text{ins}sends(U0,M0), \text{not}(\text{user}(U0)) \rightarrow \text{ins_user}(U0) \]

This means that when the users sends a message, the user also inserts a user into the database. In this case the user should enter \text{ins_user}(U0) instance. When we take a look at the variable intersection as it can be seen in the example U0 is intersected variable. In the instance U0 is john. So \text{ins_user(john)} instance should be inserted in DES. By entering the instance we fix the corresponding IC violation of EDC. However now we inserted another instance to the DES which is \text{ins_user(john)}, because of that variable now we will have other IC violation errors in EDC's.

When the process continues, the program gets a IC violation in the following EDC.

\[ \text{query}(U0) : \neg \text{user}(U0), \text{ins_user}(U0), \neg \text{profile}(U0), \neg \text{ins_profile}(U0) \]

The result of the query is:

\[ \text{‘john’} \]

The exampled EDC says that when an user inserts a username and the user is not inserted a username and profile name before he should also insert a profile name to be validated person. The following RGD should be such as;

\[ \neg \text{user}(U0), \text{ins_user}(U0), \neg \text{profile}(U0) \rightarrow \text{ins_profile}(U0) \]

This means that the user should insert \text{ins_profile(U0)} into DES system. Other thing that user should look at variable intersection and the intersection is just U0. For \text{ins_user(U0)}, U0 is john. In this case \text{ins_profile(john)} should be inserted into DES. The additional integrity constraints can be fixed in that way.
4.8.4) Iterative Approach to Fix Integrity Constraint (IC) Violation

In previous section 4.8.3 how can additional integrity constraint be fixed is explained. Now there is another problem while fixing integrity constraint violation. When the user finds another integrity constraint violation because of inserting another instance to fix integrity constraint violation in EDC, he/she should insert related instance in right hand side of the RGD. However there is an exception in this solution. The entered instances might affect another EDC or other EDC’s that previously don’t have an IC violation. In other words, the entered instance could affect the previous EDC’s which are passed as successful. To fix this problem there is an algorithm developed. The algorithm has the following steps;

1. Start
2. Detect IC Violation
3. Check if there is IC Problem
   - Yes: Check if more EDC exists in the system
     - Yes: Look at the next EDC
     - No: Check the all results of EDC
6. Check there is IC Violation in all results
   - Yes: Start checking EDC’s from beginning
   - No: Finish
4. Check the instance fixed the problem
   - Yes: Finish
   - No: The violation can’t be fixed

Diagram:
- Start
- Detect IC Violation
- Check if there is IC Problem
  - Yes: Check if more EDC exists in the system
    - Yes: Look at the next EDC
    - No: Check the all results of EDC
  - No: Check there is IC Violation in all results
    - Yes: Start checking EDC’s from beginning
    - No: Finish
- Check the instance fixed the problem
  - Yes: Finish
  - No: The violation can’t be fixed
Passing through all EDC’s in one time is not a solution. Because controlling once could not be a solution solely. When an instance is inserted and that instance violates another EDC but that EDC locates in the previous, the program should iterate all the EDC’s one more time. The most important factor in this section is looking the all results. When the program completes all the EDC’s it looks a String list. That string list holds all results of one iteration in EDC. The array does not check anything such as it includes some special characters or words or not. It directly inserts all the results into it. However at the end of the iteration there is a condition saying that if the result includes an apostrophe the second iteration should continue. The reason of the checking apostrophe is because of output type of result. When an EDC is computed, if there is an IC constraint the output would be like this;

    The Last result28
    query(U0,M0) :-
        not sendsa,
        ins_sends(U0,M0),
        not user(U0),
        not ins_user(U0).

    $'john'$
    'hola'
    $eot !!$

It means that in 28.th EDC there is an integrity constraint error and the result is represented in apostrophe. In the example it can be seen that there are two values with apostrophe which are john and hola. As it is explained previously, the String lists gets all the results without looking anything and at the
end of the iteration program checks if in that string list there is at least one apostrophe(‘) or not. If there is at least one apostrophe program goes another iteration to take a look at the result of the each iteration separately. The program finishes until it finds no violation in the string. When there is no IC violation the program prints completed in the screen, and finishes all the work completely.

4.9) Development Process

The implementation of the SafeEx tool is a software development project. According to the best development techniques in software engineering field, it is important for the project to follow some development process model, or to be more general software development methodology which is a framework used to structure, plan, and control the process of the development. Following a known methodology facilitates the planning, makes the process more transparent, which is very important when more than one person is involved in the project, makes the project results more predictable, and assists in ensuring the quality of the resulting product. There is a variety of software development methodologies, each having its strong and weak points. When choosing a methodology, the characteristics of the concrete project must be considered in order to find the one which would suit best in the given case.

Roughly, the approaches can be divided into two groups by the process model “waterfall" and iterative model. “Waterfall" methodology was one of the first methodologies of software development, and it was popular during several decades. The main idea is that the development consists of several sequential phases, usually the following: requirements definition, design, implementation, testing, and maintenance. It assumes that all the requirements can be known and understood at the beginning of the development, and that they do not change later on. While this might work for some kinds of projects, the experience of the last decade shows that it doesn't give very good results [16]. Another group of methodologies, opposing the “waterfall" model, use so called “iterative approach". Its main principle is that the development is elaborated in cycles (iterations), each iteration includes the phases, similar to the ones of the "waterfall" model which is requirements definition or refinement, design, testing, implementation, testing, but the difference is that these phases repeat several times, in each iteration. One of the examples of well-known frameworks based on iterative approach is the Rational Unified Process (RUP). It defines not only the process model, but also artifacts and roles of the software development project. RUP is a complex and elaborated methodology, and suits well for big
organizations, however for small projects it is usually too heavyweight. An alternative to these approaches is agile software development, which strictly speaking is not a methodology, but rather a philosophy, as it doesn't define exact methods and practices, but describes the general principles.

4.9.1) Agile Software Development

The main principles of the Agile software development were published in 2001 as the Manifesto for Agile Software development [17]. According to the Manifesto the value is given to:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

This means that while the things stated on the right are important, the ones from the left side of each assertion have the preference.

Figure-19 shows the illustration of the agile process method. It can be seen that while there are several levels of the process: strategy level, release level, iteration, daily work and all of them are cyclic, and independently of the level, the result is always working software.
There is a number of methodologies which follow these principles, thus they can be called “agile methodologies”. The most well-known and widely used methodologies are Scrum and eXtreme programming (XP).

4.9.2) Chosen Process Model for the Project

For the development of the current project it was decided to follow the agile principles. It suits well for the purpose of this project, as, being a research project, it is characterized by uncertainty and high probability of changes in the requirements, or design, or the technologies used. Agile provides necessary flexibility to face these challenges, and being “lightweight" approach, it allows to reduce the overhead of unnecessary documentation and complex processes.

The main principles of Agile development fit perfectly for the project of such kind. Individuals and interactions in this case have indeed more value than processes and tools, as, having a small number of people involved in the project, complex processes can just complicate the communication. The
preference for working software over comprehensive documentation is also suitable for this master project. The main objective of the project is the implementation of the designed method, and demonstration that the method works as expected. The documentation is very important too in this case, firstly for the project report (this document), and secondly, for the people who will use the developed software and also those who will continue working on it by adding new features. However, this documentation is not as important as the software itself. Contract negotiation is not applied in this case, as there is no real customer, which we are related to with contractual commitment. The tutors act as customers, defining the requirements and accepting the work done. And indeed the collaboration with them is a very important part of the project, to ensure that the software will be useful. Finally, the more value given to responding to change in comparison to following a plan also applies for the current project, because, as it was mentioned before, due to the research character of the project, it should be open for changes at any stage and allow to step back and try a different way in design, implementation or the used technologies.

After analysing Scrum [18] and XP [19] agile methodologies, I came to a conclusion that while both approaches can be applied to the current project, neither of them can be used in their entirety, and some adaptation is required. I found extreme programming methodology to be more easily adaptable.

Scrum uses quite specific roles for the team members, such as Scrum master (a role responsible for ensuring that the project is carried through according to the Scrum practices) and Product owner (responsible for maximizing the value of the product and the work of the development team), for which it was difficult to select one person. And also Scrum method has specific events, such as Daily Scrum (a short meeting of the development team the purpose of which is to synchronize activities and plan the work for the next 24 hours. This practice isn’t well adapted to the project, as while there is just one main developer, daily meeting are not applied. In cases when collaboration with the advisers or other developers is needed, the meeting is agreed beforehand.

Extreme programming is similar in many aspects to Scrum. It also values the customer satisfaction, small but frequent releases, iterative development and readiness to changes. According the Scrum the project processes in the following way;

- The schedule of the upcoming released is defined in the release planning.
• Spikes are the solutions to exact architectural, design or technical problems, which help to estimate the difficulty of the features implementation and plan the work accordingly.
• After the release plan has been defined, the project enters in the phase of elaboration, where all the work is done in multiple iterations.
• Plan of each iteration. If some tasks can’t be finished in one iteration, they can be moved to another one.
• When it is needed for some new feature (User story) appears, this is also taken into account in the release plan.

When we analyse extreme programming [19], there are some properties that can’t be applied during the project development and evaluation. For example, daily stand up meetings (analog of Daily Scrums), for the reason described above, paired programming and some others. But these practices can be omitted, while conserving the other rules, and such tailored methodology covers all the planning needs of the development part of this master thesis.

4.9) SafeEx Tool Iterations

According to the process model, described in Figure-19, the development is divided in several iterations, which form the release plan. Some of the requirements were identified prior to the start of the development, others have been appearing in the course of the project. The iterations are enumerated, and the name of the iteration reflects the main feature or the objective of the iteration.

4.9.1) Iteration I: Familiarize with Conceptual Schema, UML/OCL and Integrity Constraint

Before starting the implementation it is necessary to examine carefully and understand the Conceptual Schema. Also it is useful to familiarize with other works on the similar topics to get a broader view of the previous attempts of solving raised issues. To familiarize conceptual schema and UML/OCL diagrams some papers are read. It is really needed before starting the project. Because some steps might be forgotten and without knowing what the OCL/UML and conceptual schema are, the project can’t be completed successfully.
In these iteration general purpose is to design requirements of the project and some basic steps to prepare coding. There are not too much work needs to be done in first iteration.

4.9.2) Iteration II: Perform the Integration of Schema Conversion

The development of the software implementing the schema conversion includes system integration, testing and validating. Implementation part is not included because it had already been done by Xavier Oriol. However there are some steps that needs to be done for instance checking the development.

- Reusing schema conversion integration
- Development of schema conversion integration
- Testing schema conversion integration

In this iteration there are some requirements are checked while in development process. These are;

- The system should convert UML/OCL conceptual schema to XMI
- The system should convert XMI to Logic

This iteration is actually real beginning of the development part, the real development is started with in this iteration.

4.9.3) Iteration III: Perform the implementation of Repair-Generating Dependency (RGD)

Before putting some instances into the system should get all the of the EDC. However getting all the EDC’s solely is not the option. In addition to this, all the EDC’s should be converted into RGD. In this iteration the completed processes are in the following;

- The system gets all converted UML/OCL Schema Diagram
- The System translates schema diagram into EDC
- The system converts EDC to RGD

There were 3 initial expectations in the beginning of the iteration. However in the middle of iteration, it was realized that there should be added some changings into EDC to get successful conversion into RGD. For this reason following tasks are added into iteration;

- There should be a data structure to convert EDC to RGD
After adding that additional task, it can be said that the expected-time of the iteration finished in time however in a little bit hurry.

4.9.4) Iteration IV: Perform the implementation of Integrity Constraint Violation Checking

This is one of the most important iterations in the project. Because IC violation checking is really tough and big issue inside of the project. To develop this one the following tasks inserted in the beginning of the project;

- Obtain all Event Dependency Constraint from the xmi file
- Create necessary data structure for DES system
- Create necessary data structure for Instances
- Create necessary data structure for EDC
- Obtain DES connection
- Insert instances into DES
- Insert EDC’s into DES
- Detect IC Violation in

However while iterating the fourth level, it was concluded that there are some additional properties that need to be added into iteration

- Syntax correction of EDC’s to insert into DES
- Syntax correction of instance to insert into DES
- Unsafe variable correction in DES

These additional 3 methods was occurred after starting the iteration, in the development process when EDC’s and instances tried to insert into DES, it was understood that there should be a special syntax for the insertion. That’s why first two additional tasks are added because of syntax problems. The third problem was added because of DES again. In the DES system there is a problem for a type of lateral which is called unsafe. In section 4.6.6.1 it is explained clearly. So to fix that problem necessary changings were made in the development. However the problem is some tasks were not finished in this iteration. For instance syntax corrections were made in this iteration but the following tasks delayed and forwarded into fifth iteration.
- Insert instances into DES
- Insert EDC’s into DES
- Detect IC Violation

4.9.5) Iteration V: Perform the Implementation of Integrity Constraint Maintenance

Because of adding 3 tasks from previous iteration, this iteration includes more than expected tasks. That’s why it took more time than supposed. This iteration is also one of the most crucial one for the project. Because once program detects an IC violation, program has to fix that violation by using some methodologies. To complete IC violation constraints following tasks were used in this iteration. Firstly the tasks came from previous iteration had to be resolved.

- Insert instances into DES
- Insert EDC’s into DES
- Detect IC Violation

After resolving these 3 issues necessary tasks were started to process.

- Instance insertion to fix problem
- Validate Solution of IC Violation
- Additional IC Problems
- Iterative Approach to Fix IC Violation

This iteration also took very long time to finish it. Because there were more than 5 tasks to complete it and because of first 3 tasks, the expected finishing time of the iteration has not reached successfully.

In this iteration also some requirements completed:

- The system should be available for inserting instances
- The system should be available to detect IC violation problems
- The system should propose solution for the IC violation problem
- The system should solve IC violation
In this iteration it can be said that all the requirements have been satisfied successfully. However the expected finishing time passed 1 week and it was completed at the end of the August.

4.9.6) Iteration VI: Perform GUI

In this iteration a GUI created to demonstrate problems and solution more beautifully. However, the time was not enough to complete all GUI that was expected in the beginning of the project.

5. Project Planning and Cost

5.1) Initial Planning

As discussed in previous sections agile approach was chosen for the elaboration of the project, which means that the development process is iterative and incremental. The development is split into several iterations, and new features are added to the software incrementally. In order to make the planning of the project the key milestones were defined in such a way that each iteration of the software adds some important features. In the beginning of each iteration the analysis is performed and requirements are defined in more detail. The initial planning of the project is shown in Figure-25.
Figure 25 – Initial Planning

<table>
<thead>
<tr>
<th>Task</th>
<th>Begin</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of the Project</td>
<td>1/1/14</td>
<td>2/1/14</td>
<td>1 month</td>
</tr>
<tr>
<td>Analysis of the Pr.</td>
<td>2/2/14</td>
<td>2/9/14</td>
<td>8 days</td>
</tr>
<tr>
<td>Implementation</td>
<td>2/11/14</td>
<td>2/19/14</td>
<td>124 days</td>
</tr>
<tr>
<td>First Cycle</td>
<td>2/20/14</td>
<td>2/21/14</td>
<td>23 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>2/21/14</td>
<td>2/22/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Initial Design Eff.</td>
<td>2/23/14</td>
<td>2/24/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Implementation</td>
<td>2/25/14</td>
<td>3/2/14</td>
<td>10 days</td>
</tr>
<tr>
<td>Testing</td>
<td>3/4/14</td>
<td>3/6/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Experiment</td>
<td>3/6/14</td>
<td>3/8/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Assessment</td>
<td>3/8/14</td>
<td>3/10/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Second Cycle</td>
<td>3/11/14</td>
<td>3/17/14</td>
<td>25 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>3/12/14</td>
<td>3/13/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Design Improve</td>
<td>3/14/14</td>
<td>3/18/14</td>
<td>4 days</td>
</tr>
<tr>
<td>Implementation</td>
<td>3/19/14</td>
<td>3/21/14</td>
<td>24 days</td>
</tr>
<tr>
<td>Testing and Deo.</td>
<td>3/22/14</td>
<td>3/26/14</td>
<td>12 days</td>
</tr>
<tr>
<td>First Cycle</td>
<td>4/2/14</td>
<td>4/5/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>4/6/14</td>
<td>4/8/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Testing and Deo.</td>
<td>4/9/14</td>
<td>4/11/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Second Cycle</td>
<td>4/12/14</td>
<td>4/14/14</td>
<td>13 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>4/15/14</td>
<td>4/17/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Testing and Deo.</td>
<td>4/18/14</td>
<td>4/20/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Third Cycle</td>
<td>4/21/14</td>
<td>4/23/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>4/24/14</td>
<td>4/26/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Final Cycle</td>
<td>4/27/14</td>
<td>5/2/14</td>
<td>16 days</td>
</tr>
<tr>
<td>Analytic and Pla.</td>
<td>5/2/14</td>
<td>5/4/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Implementation</td>
<td>5/5/14</td>
<td>5/11/14</td>
<td>10 days</td>
</tr>
<tr>
<td>Testing and Deo.</td>
<td>5/12/14</td>
<td>5/14/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Preliminary Report</td>
<td>5/15/14</td>
<td>5/16/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Documentation</td>
<td>5/17/14</td>
<td>5/19/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Preparatory Work</td>
<td>5/20/14</td>
<td>5/22/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Preliminary Report</td>
<td>5/23/14</td>
<td>5/24/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Final Report</td>
<td>5/25/14</td>
<td>5/27/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Finalization</td>
<td>5/28/14</td>
<td>5/30/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Preparation of the Final Report</td>
<td>6/1/14</td>
<td>6/3/14</td>
<td>2 days</td>
</tr>
<tr>
<td>Final Report</td>
<td>6/4/14</td>
<td>6/6/14</td>
<td>2 days</td>
</tr>
</tbody>
</table>
5.2) Final Planning

During the process of the elaboration the SafeEx tool system properties, the requirements sometimes were changed or redefined, but the development mostly followed the original plan consisting of 6 iterations. A detailed description of what was developed at each iteration can be found in 4.9. However, the time length of the project tremendously increased.

The biggest reason was time of the project. The development of project was started in the beginning of June. Because previously I was doing my master thesis at a company and in March the company decided to close the project. In this case I could not do my thesis and related project in that company. After asking many lecturers for doing a possible master thesis for graduating, I found SafeEx tool project. It was the most suitable project for me to finish in a limited time. However the time was not enough to finish everything in 3 months including development, testing, interface, documentation. That’s why, most of time my initial plan was corrupted because of not having enough time to complete each iteration.

Second reason was, during April, May, June and August I was working at a company as a part time worker, so I had very limited time to study for thesis. This also caused me to not complete everything in a very short time. However I believe that, at least I completed some expected parts in the project, and prepared a detailed document which explains everything about project in a very detailed way.

5.3) Project Cost Estimation

In this section the cost of the project is estimated. Despite the length of the project turned out to be less than in the initial planning, total dedication of hours was approximately same. Because, to complete the project in a very limited time the working hours have been increased significantly. For that reason the estimation of cost will be performed using the initial planning, with the length of the project of 4 months.

The cost of the project in terms of expenses is calculated by using the following aspects;
5.3.1) Hardware Resources

As a hardware requirement, what is needed for the project is just a computer. For developing the project I used my personal laptop. One laptop is enough to do everything that’s why I can say that there is no expense in hardware resources.

5.3.2) Software Resources

During the developing of the project there are some software that has been used:

Eclipse IDE, Microsoft Word, git, DES, ArgoUML.

ArgoUML was used to convert UML/OCL conceptual schema into xmi file. It was open source software, that’s why for the ArgoUML there is no expense. DES is also open source software created by UCM professors. This one also has no expense as for the development phase. To develop software in Java, Eclipse IDE was used, and this one is also open source and has no charge.

The non-free software which was used in the project included Microsoft Word, which also required Microsoft Windows operating system installed. This could lead to significant expenses, but in my case being a student of UPC I could get this software free of charge for the period of the development of the thesis.

5.3.3) Office Resources

This category covers the equipment of the working place, which includes a desk and a chair, electricity and internet connection. Because of being just one person in the project, there is no chance to rent a room or office in somewhere else. However according to my experiences of living cost in Barcelona. The calculation of the each would be;

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Electric</td>
<td>25€</td>
</tr>
<tr>
<td>Internet</td>
<td>20€</td>
</tr>
<tr>
<td>Printing</td>
<td>5€</td>
</tr>
<tr>
<td>Chair and Desk</td>
<td>100€/4 = 25€ per month</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>75 €</strong></td>
</tr>
</tbody>
</table>

*Figure 26 – Office resources*

If we think about 4 months the equation would be: 300€
5.3.4) Human Resources

In order to estimate the costs of human resources the initial estimation of the length of the project is taken as a base.

<table>
<thead>
<tr>
<th>TASK</th>
<th>Task Description</th>
<th>Estimated Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>State of the Art of SafeEX tool</td>
<td>20 hours</td>
</tr>
<tr>
<td>2</td>
<td>Analysis of current development</td>
<td>28 hours</td>
</tr>
<tr>
<td>3</td>
<td>Study of the available environment tools to develop the project.</td>
<td>8 hours</td>
</tr>
<tr>
<td>4</td>
<td>Project environment and tools learning, installation and configuration.</td>
<td>80 hours</td>
</tr>
<tr>
<td>5</td>
<td>SafeEX Tool Design</td>
<td>48 hours</td>
</tr>
<tr>
<td>6</td>
<td>SafeEX Tool Implementation</td>
<td>150 hours</td>
</tr>
<tr>
<td>7</td>
<td>SafeEX Tool Testing</td>
<td>30 hours</td>
</tr>
<tr>
<td>8</td>
<td>Final Report Elaboration</td>
<td>76 hours</td>
</tr>
<tr>
<td></td>
<td><strong>TOTAL</strong></td>
<td><strong>410 hours</strong></td>
</tr>
</tbody>
</table>

*Figure 27 – Human resources*

If we hire a person, and propose a wage 9€/hour the following figure would explains total cost;

<table>
<thead>
<tr>
<th>Resources</th>
<th>Rate</th>
<th>Time/Period</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office Resources</td>
<td>75€/month</td>
<td>4 months</td>
<td>300€</td>
</tr>
<tr>
<td>Human Resources</td>
<td>9€/hour</td>
<td>410 hours</td>
<td>3690€</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>3990€</strong></td>
</tr>
</tbody>
</table>

*Figure 28 – Total cost*
6. Conclusions

6.1) Project Results

From the point of view of the results of the project it could be considered as adequate. Because in a very limited time I had to complete everything included documentation, development and API. Even though, I did not complete everything successfully the program works and detects the IC violation and corrects most of them in a successful way.

- Before developing that project there is no way to detect IC Violations given Event Dependency Constraint, it can be figured out manually however it took lots of time to detect with without using a computation. By using the tool now a user could insert a converted XMI and could insert check in the following UML/OCL schema in its Event Dependency Constraint Diagrams there is at least a IC violation or not.
- Secondly this tool solves the IC violation problem. If an instance causes an integrity constraint problem, the program automatically detects the problem and warns to the user. It shows to the user how the IC violation could be fixed, when user enter the instance that is suggested by program, the system automatically fixes the error.

Because of having limited time, some problems could not be fixed. For instance, if an RGD has two variables in the right hand side, and if the user tries to enter one of them, if the program finds that that is the wrong one, the process can’t come back in the initial situation. Because in DES tool there is no comeback feature. However in general, it can be said that, the program works and it detects all IC violations and mostly it solves IC Violation problems.

6.2) Personal Evaluation

The general evaluation myself in this project is average. It was my first project that I have completely designed and completed. Because in my bachelor degree, there were 3 people in final project, so for me this is the first time in my life.

I was quite unlucky in the beginning of the project, because normally a project should start in February however while doing my thesis in a company, the company decided to close the whole project
and I could not realize my master thesis in that company. I started to search a thesis in April and finally I found this beautiful project in the middle of May. I started everything in June, however because of not having enough time for completing everything I was not quite satisfied what I was done in the project.

I was quite lucky, because in this master I have specialized in Software Engineering field, and this project is also about Software Engineering. Also thanks to the MIT master, I already taken Software Engineering I and Software Engineering II courses. By taking that, I had enough knowledge about UML and OCL. The only problem is I don’t know so much things about deductive database and its software such as Prolog and Datalog Educational System but during the project I got used to these software too.

Thanks to the project, I understand how to detect integrity constraint violation, how to solve integrity constraint violation and I learned different tools such as DES. I also learned small problems could make some big errors in everything. A small problem could cost your weeks in the project. This was another good thing in the project that I learned. I also improved my problem solving skills by looking the solution way in the document or in an academic paper or in journal.

I am really happy to work with very helpful lecturer and my thesis director Ernest Teniente, also I am very delighted to collaborate Xavier Oriol who is a PhD student in UPC.

6.3) Future Work

During the project the tool has solved significant errors which is caused by IC violation. However, it is not possible to say that everything is completed during 4 months of master thesis period. Here are some of them;

- Building an IC violation without using DES:

  In this project to detect IC violations mainly DES is used, however after processing many steps, there is some deficiencies detected in DES. Because when there can be two solution possibility to fix IC violation in given EDC, the user choses one of them. The chosen instance is inserted into DES, however if the chosen solution way is wrong then the everything should come back to beginning of the problem occurance. This causes a real problem in DES. Because in thes system when a instances is inserted there is no way to delete that instance from DES. To fix the
problem, there could be created a new IC violation detection system by developers.
The good way of developing a new system is there is no synax restriction while
inserting instances and Event Dependent Constraints. Only problem while
developing system is time. Because probably to develop such kind a program takes
much longer time comparing to DES. That’s why if there is enough time to develop
such kind of system, there could be good option to use it in the program. In other
problem in DES is, it is such a huge program and for the development phase of the
project some properties are not needed. However because of using whole system
these unnecessary properties can’t be discarded while developing the tool.

- Better GUI:

Because of having very limited time for designing and developing the project. The
GUI was not designed very user friendly. In future work the GUI could be improved.
There are some problems in GUI also. For instance while inserting a value, the user
should know instance types. These instance types does not be shown in GUI. To use
the program very effectively the GUI could be improved.

- Adding new properties to tool:

This tool is a very basic one for detecting integrity constraint and finding some
solutions according to these problem. However some new properties could be
added into the system.
Bibliography


Acronyms

GUI
Graphical User Interface

IDE
Integrated Development Environment

XML
Extensible Markup Language

XMI
XML Metadata Interchange

DES
Datalog Educational System

IC
Integrity Constraint

EDC
Event Dependency Constraints

RGD
Repair Generating Dependency

RDBMS
Relational Database Management System

UML
Unified Modeling Language

OCL
Object Constraint Language

SQL

Structured Query Language
Appendix

Appendix-1

Converted Schema example. Related UML/OCL schema is in Figure-22:

```xml
<?xml version='1.0' encoding='UTF-8' ?>

    
    <XMI.header>
        <XMI.documentation>
            <XMI.exporter>ArgoUML (using Netbeans XMI Writer version 1.0)</XMI.exporter>
            <XMI.exporterVersion>0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100 (Mon, 11 Jan 2010) $</XMI.exporterVersion>
        </XMI.documentation>
    </XMI.header>

    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>

    <XML.content>
        <UML:Model xmi.id='-'109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000865'
            name = 'untitledModel' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
            isAbstract = 'false'>
            
            <UML:Namespace.ownedElement>
                <UML:Class xmi.id='-'109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000866'
                    name = 'User' visibility = 'public' isSpecification = 'false' isRoot = 'false'
                    isLeaf = 'false' isAbstract = 'false' isActive = 'false'>
                    
                    <UML:GeneralizableElement.generalization>
```

83
<UML:Generalization xmi.idref='109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008AC'/>
</UML:GeneralizableElement.generalization>
</UML:Class>

<UML:Class xmi.id='109-83-72--5-e7c62dc:146b422d1df:8000:0000000000000867'
name='Message' visibility='public' isSpecification='false' isRoot='false'
isLeaf='false' isAbstract='false' isActive='false'/>

<UML:Association xmi.id='109-83-72--5-e7c62dc:146b422d1df:8000:0000000000000869'
name='Sends' isSpecification='false' isRoot='false' isLeaf='false'
isAbstract='false'>

<UML:Association.connection>

<UML:AssociationEnd xmi.id='109-83-72--5-e7c62dc:146b422d1df:8000:000000000000086A'
name='sender' visibility='public' isSpecification='false' isNavigable='true'
ordering='unordered' aggregation='none' targetScope='instance' changeability='changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id='109-83-72--5--1d0c5f2a:14762170b73:-8000:00000000000009B3'>

<UML:Multiplicity.range>

<UML:MultiplicityRange xmi.id='109-83-72--5--1d0c5f2a:14762170b73:-8000:00000000000009B2'
lower='0' upper='1'/>

</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>

<UML:AssociationEnd.participant>
  <UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000866'/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000086B'
  name = 'msg' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
  ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>
  <UML:AssociationEnd.multiplicity>
    <UML:Multiplicity xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008BF'>
      <UML:Multiplicity.range>
        <UML:MultiplicityRange xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008BE'
          lower = '0' upper = '-1'/>
      </UML:Multiplicity.range>
    </UML:Multiplicity>
  </UML:AssociationEnd.multiplicity>
  </UML:AssociationEnd.participant>
</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000867'/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>

<UML:Class xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000878' name = 'Text' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'/>

<UML:Class xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000087C' name = 'Time' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'/>

<UML:Class xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000087D' name = 'Group' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'>

<UML:GeneralizableElement.generalization>

<UML:Generalization xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008AB'/>

<UML:Generalization xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008AF'/>

</UML:GeneralizableElement.generalization>

</UML:Class>

<UML:Association xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000087F' name = 'MemberOf' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
<UML:Association.connection>
    <UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000080'
        name = 'group' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
        ordering = 'unordered' aggregation = 'aggregate' targetScope = 'instance'
        changeability = 'changeable'>
        <UML:AssociationEnd.multiplicity>
            <UML:Multiplicity xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000000C9'>
                <UML:Multiplicity.range>
                    <UML:MultiplicityRange xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000000C8'>
                        lower = '0' upper = '-1'/>
                </UML:Multiplicity.range>
            </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
        <UML:AssociationEnd.participant>
            <UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000008D'/>
        </UML:AssociationEnd.participant>
    </UML:AssociationEnd>

    <UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000081'
        name = 'userG' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
        ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability =
        'changeable'>
</UML:AssociationEnd>
<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = '-109-83-72--5--1d0c5f2a:14762170b73::8000:00000000000009B9'>

<UML:Multiplicity.range>

<UML:MultiplicityRange xmi.id = '-109-83-72--5--1d0c5f2a:14762170b73::8000:00000000000009B8'>
lower = '0' upper = '-1'/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>

<UML:AssociationEnd.participant>

<UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df::8000:0000000000000866'/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>

<UML:Association xmi.id = '-109-83-72--5-e7c62dc:146b422d1df::8000:0000000000000894'
name = 'OwnedBy' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
isAbstract = 'false'>

<UML:Association.connection>

<UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df::8000:0000000000000895'
name = 'owned' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008C7'>

<UML:Multiplicity.range>

<UML:MultiplicityRange xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008C6'

lower = '0' upper = '-1'/>

</UML:Multiplicity.range>

</UML:Multiplicity>

</UML:AssociationEnd.multiplicity>

<UML:AssociationEnd.participant>

<UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:000000000000087D'/>

</UML:AssociationEnd.participant>

</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000896'

name = 'owner' visibility = 'public' isSpecification = 'false' isNavigable = 'true'

ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = '-109-83-72--5--1d0c5f2a:14762170b73:-8000:00000000000009BB'>

</UML:Multiplicity>
name = 'msg' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008BD'>

<UML:Multiplicity.range>

<UML:MultiplicityRange xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008BC'>

lower = '0' upper = '-1'/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>

<UML:AssociationEnd.participant>

<UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:0000000000000867'/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008B7'
name = 'receiver' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = '-109-83-72--5-1d0c5f2a:14762170b73:-8000:00000000000009B7'
name = 'Pair' visibility = 'public' isSpecification = 'false' isRoot = 'false'
isLeaf = 'false' isAbstract = 'false' isActive = 'false'>

<UML:GeneralizableElement.generalization>

<UML:Generalization xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008D9'/>

</UML:GeneralizableElement.generalization>
</UML:Class>

<UML:Generalization xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008D9'
isSpecification = 'false'>

<UML:ModelElement.comment>

<UML:Comment xmi.idref = '-109-83-72--5--69830004:146b4868b65:-8000:00000000000009AE'/>

</UML:ModelElement.comment>

<UML:Generalization.child>

<UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008D8'/>

</UML:Generalization.child>

<UML:Generalization.parent>

<UML:Class xmi.idref = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008AE'/>

</UML:Generalization.parent>

</UML:Generalization>

<UML:Association xmi.id = '-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008DA'
name = '' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
name = 'pair' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>

<UML:Multiplicity xmi.id = "-109-83-72--5--69830004:146b4868b65:-8000:00000000000009B5">

<UML:Multiplicity.range>

<UML:MultiplicityRange xmi.id = "-109-83-72--5--69830004:146b4868b65:-8000:00000000000009B4">
lower = '0' upper = '-1'/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>

<UML:AssociationEndparticipant>

<UML:Class xmi.idref = "-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008D8"/>
</UML:AssociationEndparticipant>
</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = "-109-83-72--5-e7c62dc:146b422d1df:-8000:00000000000008DF"
name = 'userP' visibility = 'public' isSpecification = 'false' isNavigable = 'true'
ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'>

<UML:AssociationEnd.multiplicity>