ENHANCEMENT OF NATURAL LANGUAGE PROCESSING APPROACH FOR AUTOMATED GENERATION OF OBJECT CONSTRAINT LANGUAGE

SAMIN SALEMI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

SEPTEMBER 2015

Dedication

To my loving parents, Masoum and Mahmoud

And my supportive brother

ACKNOWLEDGEMENT

I would like to express my special appreciation and thanks to my supervisor, Foremost, I would like to express my sincere gratitude to my advisor Prof. Ali Selamat for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

My parents, I don’t think I can find proper words to express my gratitude towards them. It is their encouragement and support from the very beginning of my life that made it possible for me to reach this stage. Sina, my wonderful brother, thank you for always supporting me. Dear Nicholas, I will never forget what you have done for me.

ABSTRACT

Object Constraint Language (OCL) is the most prevalent modeling language to document requirement constraints that are annotated in the Unified Modeling Language. Various researchers have proved that OCL syntax is complex and difficult for some reasons such as its declarative nature. As the measure of ease-of-use factor of a language has a direct relationship with the language's usability, the difficulties in the use of OCL result in the low usability of OCL. There are few research works for OCL generation using some different techniques such as pattern-based and Model-Driven Architecture (MDA)-based. The accuracy of the existing pattern-based work generating OCL specification is low. MDA focuses on software development based on generating models and transforming these models between each other. There are some researches based on MDA to increase the usability of modeling languages. However, only one of the existing works supports OCL. The existing MDA-based work generating OCL specification does not support some OCL elements, such as `collect` and `reject`, and some UML elements such as enumeration. Therefore, this research proposes an MDA-based approach to transform requirement constraints formed in English sentences into OCL specifications using transformation rules. A software tool is developed to validate the proposed approach and compare with the existing works. The comparison shows that the proposed approach solves some limitations of the existing works such as support of some OCL and UML elements, which are not supported by the existing works. The comparison also shows that some accuracy improvement is achieved by the proposed approach in comparison with the existing works.

ABSTRAK

Bahasa Kekangan Objek (OCL) merupakan bahasa pemodelan yang sering digunapakai bagi mendokumenkan kekangan keperluan yang dianotasi ke dalam Bahasa Pemodelan Bersatu (UML). Ramai penyelidik telah membuktikan yang sintaks OCL adalah kompleks dan sukar dan ia berpunca dari pelbagai faktor seperti sifat deklaratif nya. Oleh kerana pengukuran mudah guna bahasa mempunyai perkaitan terus dengan kebolehgunaan bahasa, kesukaran penggunaan OCL menyebabkan kadar kebolehgunaan yang rendah. Terdapat beberapa kajian berkenaan dengan penghasilan OCL menggunakan teknik-teknik berbeza seperti penggunaan berdasarkan corak dan senibina berdasarkan model. Ketepatan kerja sedia ada berdasarkan corak bagi menghasilkan OCL adalah rendah. MDA menumpukan kepada penmbangunan perisian berdasarkan penghasilan model dan transformasi di antara model-model. Terdapat kajian berdasarkan MDA bagi meningkatkan kebolehgunaan bahasa. Bagaimanapun hanya terdapat satu kerja sedia ada yang menyokong OCL. Kerja ini yang berdasarkan MDA bagi menghasilkan spesifikasi OCL bagaimanapun tidak menyokong sesetengah elemen OCL seperti `collect` dan `reject` dan sesetengah elemen UML seperti penghitungan. Oleh itu, kajian ini mencadangkan satu pendekatan berasaskan MDA bagi menukar kekangan keperluan di dalam bahasa Inggeris kepada spesifikasi OCL menggunakan peraturan transformasi. Satu aplikasi dibangunkan untuk mengesahkan pendekatan yang dicadangkan dan perbandingan dibuat dengan kerja sedia ada. Hasil perbandingan menunjukkan pendekatan yang dicadangkan dapat menyelesaikan kekurangan di dalam kerja sedia ada seperti sokongan terhadap sesetengah elemen OCL dan UML yang tidak disokong oleh kerja sedia ada. Perbandingan juga menunjukkan peningkatan ketepatan dapat dicapai menggunakan pendekatan yang dicadangkan berbanding kerja sedia ada.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVATIONS

AI	-	Artificial Intelligence
ATL	-	Atlas Transformation Language
BLAST	-	Berkeley Lazy Abstraction Software Verification Tool
BOTL	-	Bidirectional Object-oriented Transformation language
BPM	-	Business Process Model
CASE	-	Computer-aided software engineering
CBR	-	Case-Based Reasoning
CC	-	Coordinating Conjunction
CIM	-	Computation Independent Model
CIRCE	-	Cooperative Interactive Requirements-Centered Environment
CM-Builder	-	Class Model-Builder
COPACABANA	-	COnstraint PAtterns and Consistency Analysis
CS	-	Conceptual Schema
CWM	-	Common Warehouse Metamodel
DFD	-	Data flow diagram
DM	-	Direct Manipulation
DMG	-	Data Mining Group
DSL	-	Domain Specific Language
DSME	-	Domain Specific Modeling Environment
DSPIM	-	Domain Specific Platform Independent Model
DUPA	-	Declarative Reusable Pattern Approach
EFA	-	Exploratory Factor Analysis
EJB	-	Enterprise Java Bean
EMF	-	Eclipse Modeling Framework
ERM	-	Entity Relationship Model
FN	-	False Negative
FP	-	False Positive

GATE	-	General Architecture for Text Engineering
GOOAL	-	Graphical Object Oriented Analysis Lab
GRAI	-	Graphs with Results and Activities Interrelated
GReAT	-	Graph Rewriting and Transformation
GT	-	Graph transformation
GUI	-	Graphical User Interface
I/O	-	Imperative/Operational
Jamda	-	Java Model Driven Architecture
JAPE	-	Java Annotation Patterns Engine
LHS	-	Left Hand Side
LIDA	-	LInguistic assistant for Domain Analysis
LOLITA	-	Large-scale Object-based Linguistic Interactor
LTL	-	Linear Temporal Logic
MAMT	-	Metamodeling Approach to Model Transformation
MDA	-	Model-Driven Architecture
MDAMT	-	Model Driven Approach to Model Transformation
MDD	-	Model-Driven Development
MOF	-	Meta Object Facility
MOLA	-	MOdel transformation Language
Mopa	-	MOdel PAttern
MT	-	Model Transformation
NER	-	Named Entity Recognition
NIBA	-	Natural Language Requirements Analysis in German
NL	-	Natural Language
NL-OOPS	-	Natural Language-Object Oriented Production System
NLP	-	Natural Language processing
NOESIS	-	Natural Language Oriented Engineering System for Interactive Specifications
OBFS	-	Object-Based Formal Specification
OCL	-	Object Constraint Language
ODM	-	Ontology Definition Metamodel
OMG	-	Object Management Group
OQAPI	-	OCL Query Application Program Interface

PC	-	Preposition Conjunction
PCA	-	Principal Component Analysis
PIM	-	Platform Independent Model
PMML	-	Predictive Mark-up Modeling Language
POS	-	Part Of Speech
PSL	-	Property specification language
PSM	-	Platform Specific Model
PRR	-	Production Rule Representation
QVT	-	Query/View/Transformation
R/D	-	Relational/Declarative
RE	-	Requirements Engineering
RFP	-	Requirement of Request for Proposal
RHS	-	Right Hand Side
RUP	-	Rational Unified Process
SBVR	-	Semantics of Business Vocabulary and Business Rules
SD	-	Structure-Driven
SDLC	-	Software Development Life Cycle
SiTra	-	Simple Transformer
SUGAR	-	Static UML model Generator from Analysis of Requirements
TB	-	Template-Based
TCTL	-	Temporallogik Timed Computation Tree Logic
TGG	-	Triple Graph Grammars
TLG	-	Two-Level Grammar
TN	-	True Negative
TP	-	True Positive
TRADE	-	Techniques for Requirements and Architecture Design
UCDA	-	Use Case Driven Analysis
UML	-	Unified Modelling Language
UMLG	-	UML Generator
USE	-	UML-based Specification Environment
VB	-	Visitor Based
VDM	-	Vienna Development Method
XMI	-	XML Metadata Interchange

XML	-	Extensible Mark-up Language
XSLT	-	Extensible Style Sheet Language Transformation
4W	-	Who, What, Where, When

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1. Overview

The main aim of the Object Management Group (OMG) is to set standards for object-oriented systems. These standards are model-based standards that are used for modeling of processes, systems, and programs. A model is a set of consistent elements with some features and restrictions that these elements represent objects of a real system and relationships between them (Warmer and Kleppe, 2003).

Software designers use modeling languages to document system requirements and constraints. The characteristics of the modeling languages selected by the software designer impact on some factors of software modeling, such as time spent, effort required, number of errors, etc. Thus, we need to select appropriate modeling languages to increase the quality of models (Bobkowska, 2005). One of the significant characteristics of a modeling language is usability. The usability of a modeling language has a significant impact on confidence, effort saving, and speed in model construction using the modeling language (Figl et al., 2009). The quality of models can be increased by improving the usability of the modeling languages used.

The Object Constraint Language (OCL) is the most prevalent modeling language to document requirement constraints, which a software designer is not able to show by Unified Modeling Language (UML). In order to improve the precision of UML models, OCL is needed to express requirement constraints. An OCL specification integrated into a UML model is a Boolean condition that must be true for each instance of the UML model. The low usability of OCL due to its difficult syntax causes the gap between requirement constraints written in natural languages and OCL specifications become bigger and bigger. Thus, the low usability of OCL increases time and effort spent on the design phase of software development significantly and influences on overall development costs (Bajwa, 2012).

The Model-Driven Architecture (MDA) approach defined by OMG changes the software development style from code-oriented to model-oriented (Kardoš and Drozdová, 2010). The philosophy of MDA is to describe each artifact as a model and to transform the models to each other (Jilani et al., 2010). MDA provides a more efficient approach for software development, if the model transformations are done according to their specifications. The model-based software development methodology is transferable due to its platform dependency. The result of its transferability is its reusability (Kardoš and Drozdová, 2010). According to the MDA approach, the current research hypothesizes that a requirement constraint can be formed in an English sentence as a model then the English model can be transformed to an OCL specification as a model.

There is a usability definition made by ISO: "*The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.*" (ISO 9241-11, 1998). When the extent is high, the usability is high. When the extent is low, the usability is low. Usability is composed of some factors such as learnability, efficiency, memorability, errors, and satisfaction. In order to increase the usability of OCL, the current research aims to propose an approach to transform requirement constraints formed in English sentences into OCL specifications. This transformation is based on the philosophy of MDA.

As the foundation of English is not on a formal logic, translation of English to a formal logic such as the formal logic underpinning OCL is too difficult. Thus, a language, which its foundation is on a formal logic, must be used as an intermediate language (Bajwa, 2012). The Semantic of Business Vocabulary and Rules (SBVR) standard proposed by OMG is used to present natural language statements in formal declarative descriptions (OMG, 2013). SBVR representations are easy to process by machines. As the foundation of SBVR is based on higher order logic, transformation of SBVR into other formal languages is simpler than transformation of natural language descriptions into formal languages such as OCL (Afreen et al., 2011).

1.2. Problem Background

A kind of knowledge can be presented in various ways. For example, the same knowledge can be written in natural languages and can be depicted using diagrams and specifications, such as UML diagrams and OCL specifications. There is a considerable gap between these presentation types. There are many existing works which use the MDA approach to fill the gap between requirements written in a natural language and object-oriented models by transforming these requirements into the equivalent object-oriented models.

The Large-scale Object-based Linguistic Interactor (LOLITA), which is a computer-aided software engineering (CASE) tool based on natural language processing (NLP), was described by Mich (1996) to analyze natural language texts for extracting objects and associations and generating object models. Overmyer et al. (2001) provide the Linguistic Assistant for Domain Analysis (LIDA), which translates natural language descriptions to UML Class diagrams. 4WL (Who, What, Where, and When Language) is a semi natural language presented by Perez-Gonzalez and Kalita (2002) to transform natural language descriptions into dynamic object models automatically. A technique called Partially Ordered sets of roles (Role Posets) has been proposed by the researchers to automatically translate simple sentences to the 4W language and to produce Class, Object, and Use Case diagrams. Class Model-Builder (CM-Builder) is a CASE tool presented by Harmain and Gaizauskas (2003) to analyze requirements formed in English descriptions using a robust NLP technique and to generate UML Class diagrams interactively or automatically. Grünbacher et al. (2004) introduced an approach called Component Bus System Property (CBSP) to ease capturing relationships between software requirements and architectural models using intermediate models. Li et al. (2005) proposed an algorithm to process natural language texts and to translate them to

UML models with user interaction. Ambriola and Gervasi (2006) presented an environment named Cooperative Interactive Requirement-Centric Environment (CIRCE) to analyze software requirements formed in natural language descriptions and to transform the requirements into static and dynamic diagrams, such as Entity Relationship (ER), UML Class models, and Finite State automata. Natural Language Based Requirements Analysis in German (NIBA) presented by Fliedla et al. (2007) is an approach to analyze requirement descriptions linguistically and to translate them to a Conceptual Schema (CS) such as Activity, Class, or State diagram. Kumar and Sanyal (2008) developed a tool called Static UML model Generator from Analysis of Requirements (SUGAR) to generate UML Class and Use Case diagrams from requirements formed in a natural language. Bajwa et al. (2009) presented an automated system based on NLP to translate natural language requirements to object-oriented models and to create code in some languages. Amdouni et al. (2011) have presented a tool to translate requirements formed in text documents to UML Class diagrams using NLP rules. NL2OCLviaSBVR developed by Bajwa (2012) is an approach, which is the only existing MDA-based work that supports OCL. Wang (2013) proposed the Environment Based Design (EBD), which is a methodology to represent natural language in conceptual models such as UML Use Case, Domain, and Function-Behavior-State (FBS) models.

These works mentioned above have focused on model-driven approaches for transforming natural languages into object-oriented diagrams. However, only one of them supports OCL. In modern software engineering, graphical models such as UML diagrams present a conceptual schema of a software application. As UML diagrams cannot present meaningful time constraints of systems, the diagrams are incomplete (Calegari et al., 2008). In order to improve the precision of these diagrams, textual constraints such as OCL specifications are used to document the constraints of systems. However, there are some problems in using OCL as follow:

i. Low usability of OCL

The utilization of a modeling language refers to the usage extent from the language. When a language has a low utilization with other standards, it means that the language cannot be integrated into the other standards very well. On the other hand, a language, which has a high utilization with other standards, can be used by the other standards easily. In a computer language, symbols are combined according to a set of rules called syntax. The language syntax determines how to make a correct specification using the language. Syntactic complexity is a kind of complexity in languages that the syntactic complexity has effects on some factors of the language, such as speed of learning and extent of usage. Despite the fact that OCL can have a high utilization with other object-oriented standards because of the maturity of its syntax and semantics (Chimiak-Opoka, 2009), it has also been discovered that OCL has a complex syntax (Cabot and Teniente, 2006). OCL statements just describe what it wants to accomplish due to its declarative nature. The declarative nature of OCL is a major reason that makes it difficult to understand (Bajwa, 2012). Thus, OCL is a difficult language especially for users who are new in this language and have little knowledge of OCL (Cabot, 2006; Wahler, 2008; Chimiak-Opoka, 2009; Bajwa, 2012; Störrle, 2013). Thus, software designers struggle with learning and using OCL, in both industry and academia (Störrle, 2013).

The only existing language for annotating UML models is OCL. On the other hand, OCL development is an extremely time-consuming process (Chimiak-Opoka, 2009). In practice, writing requirement constraints in OCL specifications has two steps: first, the constraints are written in a natural language such as English, and then these constraints formed in natural language texts are transformed into OCL specifications by an OCL expert, manually (Bajwa, 2012). Some researches performed in the last few years analyzed the OCL usability. These analyses show that the OCL usability is low (Cabot and Teniente, 2006; Wahler, 2008; Bajwa, 2012; Störrle, 2013) and high time and effort are needed to create and modify OCL specifications for designing large system models (Chimiak-Opoka, 2009). Thus, the low usability of OCL has major effects on economic issues in large systems development (Störrle, 2013). As usability is a key characteristic of a modeling language, the modeling language concepts, such as amount of errors made by users, are concerns with the usability. According to the problems explained in the previous sub-sections, OCL development is an error–prone task (Chimiak-Opoka, 2009). Writing OCL specifications manually may result in erroneous and inconsistent specifications because of the low usability of OCL (Cabot and Teniente, 2006;

Wahler, 2008; Bajwa, 2012; Störrle, 2013). Thus, creating OCL specifications especially by users with little or no prior knowledge of the OCL syntax and semantics is an extremely risky process (Bajwa, 2012).

ii. Lack of creation of OCL specification in existing OCL tools

Most of the existing OCL tools, such as Dresden OCL Toolkit, IBM OCL Parser, USE, ArgoUML, and Cybernetic OCL Compiler, just perform validation, syntax verification, type checking, and parsing of OCL expressions (Bajwa et al., 2010). They cannot help to create OCL specification.

iii. No support of OCL by most of the existing MDA-based approaches

OCL generation automatically can increase the usability of OCL. There are some MDA-based approaches to improve the usability of modeling languages. However, only one of them supports OCL. On the other hand, There are some OCL generators, such as COPACABANA (Wahler, 2008), NL2OCLviaSBVR (Bajwa, 2012), and OQAPI (Störrle 2013), though, only two of them generate OCL as constraint specifications.

iv. Need of assistance for writing OCL specification

In line with the problems elaborated in the previous sub-sections about the complexity of the OCL syntax and its side effects, writing OCL specifications is a complex task. Thus, there is need of an approach for giving assistance to developers in writing OCL specifications. There are only two approaches to assist developers in writing OCL specifications. The first one is the pattern-based approach proposed by Wahler (2008) for helping to create OCL specifications. Wahler implemented the proposed approach in a tool named COPACABANA. Bajwa (2012) also proposed an MDA-based approach to assist developers in writing OCL specification by a tool called NL2OCLviaSBVR that generates OCL specification automatically.

1.3. Problem Statement

Nowadays, there are few works for generating OCL specifications, such as COPACABANA (Wahler, 2008), NL2OCLviaSBVR (Bajwa, 2012). The two existing works, which help to write OCL specification, have some limitations. For example, users of COPACABANA should extract information from natural language constraints manually. Patterns also should be selected by users manually. Substantial effort is required for adding new patterns into the pattern list. The low accuracy of the tool, which is about 69%, is another limitation of the tool (Bajwa, 2012).

NL2OCLviaSBVR also has some limitations. For example, the tool processes only one input English sentence at a time. It does not support UML enumerations. It also does not support some OCL elements such as `collect()`, `reject()`, and `oclIsTypeOf (T)`. The tool used SiTra, which has been developed by Akehurst et al. (2006) for implementing mapping rules. As SiTra has some limitations, the limitations of SiTra are the limitations of the NL2OCLviaSBVR tool. For example, one of the major limitations of SiTra regards a situation in which there is more than one rule that should map to the same target object. There is no way to determine, using SiTra, which of the rules should construct the target object. The tool's accuracy is about 84% (Bajwa, 2012) which can be improved by solving the limitations.

1.4. Research Questions

The complexity of a language indicates to the resources needed to read and write specifications written using the language. The complexity and difficulty of the OCL syntax explained in the previous section causes some effects such as rising time and effort needed to create OCL specifications and increasing errors in writing OCL specifications. As mentioned in Section 1.1, the ISO definition of usability is: "*The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.*". In modeling languages, the extent is depended on some factors such as learnability, efficiency, memorability, and satisfaction. Thus, the low measure of these factors results in the low usability of the modeling languages. In the current study, the modeling language, which is considered, is OCL. The problems mentioned in the previous section causes the low usability of OCL. Thus, the current study aims to improve the OCL usability. This improvement can be achieved by using MDA. This research aims to use MDA to transform English sentences into OCL specifications. As natural languages such as English are informal descriptions, their translation to formal languages, such as OCL, is very hard. Thus, SBVR business rules are used as intermediate specifications because of their formal foundation. The questions presented below are the research questions corresponding to the objectives that the current research has.

i. How to extract sematic business rules from English sentences?

ii. How to transform business rules into OCL specifications?

iii. How to improve the OCL usability by an MDA-based approach?

1.5. Research Goal

Modeling languages are used to document system requirements and constraints in the design phase of the software development life cycle (SDLC). The characteristics of the modeling languages that are selected by software designers have impacts on some factors of software modeling such as time and effect required. Thus, the simplicity of the design phase has a direct relation with capabilities of the modeling language chosen by software designers. The main goal of the current research is to simplify the design phase of software modeling by proposing an approach to generate OCL specifications automatically. The proposed approach accelerates generating and updating OCL specifications by automating OCL generation by which system designers can transform constraint requirements formed in English sentences into corresponding OCL specifications. Thus, the time required to create OCL specifications is reduced.

1.6. Objectives

In order to achieve the goal mentioned above, the study must answer the research questions presented in Section 1.4. The answer of these research questions solve the problems elaborated in Sections 1.3. There are some objectives to answer the questions. All of these objectives are achieved in the current research. The objectives of this research are as follows:

i. To propose mapping rules for extracting semantic business rules from English sentences.

ii. To propose mapping rules for transforming sematic business rules into OCL specifications.

iii. To propose an MDA-based approach to improve the usability of OCL.

iv. To evaluate the proposed approach in evaluation metrics, such as accuracy, and OCL usability improvement.

1.7. Research Scopes

The problems and resources, which have been used, determine the scope of the current study. Five main concepts form the focus in the current study. The concepts are explained below:

i. UML models

OCL is a standard sub-language of UML to present requirement constraints that cannot be generated by UML. As OCL does not represent temporal aspects directly, it is not appropriate for UML models that are based on temporal logic such as Activity models. Thus, the current research focuses on UML Class models.

ii. Model transformation technique

OMG focuses on changing SDLC from code-oriented to model-oriented by the MDA approach. The model transformation technique that is the philosophy of MDA describes each artifact as a model and transforms the models to each other (Jilani et al., 2010). The current research uses the model transformation technique in the proposed approach by which English sentences that can be transformed into OCL specifications.

iii. Validation

In order to validate the proposed approach is implemented in a java tool. The tool must be compared with the existing tools. As only the execution file of the existing tools is available, these tools must be tested by the black box testing technique and compared with each other. Black-box testing peers into the functionality of an application not into the internal structure of the application. Accuracy, Precision and Recall are the most common metrics to evaluate an information retrieval system. F-measure is another evaluation metric that shows the harmonic mean of precision and recall. The proposed approach is evaluated and compared with the existing works using the evaluation metrics.

iv. Statistical techniques for usability testing

As OCL usability improvement is a major goal of the current research, it must be confirmed whether there is any usability improvement or not. ISO 9241-11 suggests that measures of usability should cover three factors involving: efficiency, effectiveness, and satisfaction (Damljanovic and Bontcheva, 2009). Thus, a survey is designed to measure the OCL usability using these three factors. The efficiency factor is measured using effort-saving and time-saving variables. The effectiveness factor is measured using writability and confidence variables. The satisfaction factor is measured using ease-of-use and comfort variables. Two states are considered in the OCL usability measurement. In the first state, users write OCL specifications manually. In the second state, users generate OCL specifications using the tool implementing the proposed approach. Users' responses are analyzed using the Exploratory Factor Analysis (EFA) to eliminate the variables, which do not have any significant impact on any factor. The analysis also eliminates variables, which have cross-loading. Variables that have cross-loading have significant impacts on more than one factor. Thereafter, the amount of improvement in these variables is measured by a comparison between the two states.

1.8. Research Motivations

Research justification refers to the rationale for the research. The justification of a research is the reason that why the research is being conducted. Thus, this section explains about the reasons that motivate the current research. In order to explain about the motivation of the current research, the importance of two issues must be explained as follow:

i. Importance of OCL

The UML graphical modeling language is not able to show requirement constraints. OCL is the most prevalent language to document requirement constraints. Thus, in order to improve the precision of UML models, OCL is needed to express requirement constraints. UML models are completed by OCL specifications that express additional information about the object-oriented artifacts. UML models without constraint specifications cannot express all information. Thus, these models must be integrated into OCL specifications that express the constraints of the system being modeled. Combining UML and OCL increases the maturity of the system modeling process. Despite the fact that OCL has been introduced more than ten years ago, the adaptability of OCL is still lower than other languages. In the software modeling community, researchers must try to increase the adaptability and acceptability of OCL.

ii. Importance of modeling language characteristics

In the software development process, models have major roles in generation, integration, and maintenance of models. In complex systems, complex diagrams and constraint specifications are integrated to document processes and system requirements. Good modeling languages are required to achieve high quality models (Bobkowska, 2005). In software development tasks, modeling languages have effects on some characteristics such as difficulty level, time required for modeling and searching information, usability, number of errors, and level of automation (Figl et al., 2009).

Usability is a key characteristic of a modeling language. The usability characteristic depends on some factors such as ease of learning, ease of use, efficiency, time needed to perform a task, productivity, amount of errors made by users, memorability, and user satisfaction. In constructing models using a specific modeling language, usability has a direct relationship with the effort and time required in generation of the models. Therefore, improvement of the OCL usability is a real need that must be considered by researchers, because the improvement can have significant effects on the design phase of software modelling.

1.9. Research Outcomes

There are some problems, which are motivations for embarking on the current research. Mainly, these problems are related to the complexity of the OCL syntax that causes some other problems such as high time and effort required for using OCL and high risk in writing OCL specifications. The main goal of the current research is to simplify the design phase of software modeling by proposing an MDA-based approach to improve the usability of OCL. Two sets of mapping rules are proposed in the MDA-based approach. The first set of mapping rules is used for extracting semantic business rules from English sentences. The second set of rules is used for transforming the sematic business rules into OCL specifications. Finally, the proposed approach is evaluated in metrics, such as accuracy, and OCL usability improvement.

1.10. Research Outline

The outline of a research is a general plan of the materials presented in the research. This section presents the order of main parts of the current study, their tasks, their importance, and the relationship between them. The current research is organized in seven chapters as follow:

Chapter 1, introduction, gives an overview on structure and nature of the study. This chapter elaborates the study background, and the research problems are discussed. Thereafter, the research questions, the main goal and the objectives by which the research goal is achieved are explained. Finally, the chapter justifies the importance of the current study.

Chapter 2, literature review, investigates the prior studies about the area of the current study. The only existing work on OCL generation from English texts is deeply analyzed and its limitations are identified.

Chapter 3, research methodology, describes the approaches and concepts used in the current study. The chapter explains the procedure of the current research step-by-step in detail. Furthermore, the chapter shows how the proposed approach called En2OCL achieves the objectives.

Chapter 4, the proposed En2OCL approach for OCL generation, explains how the MDA approach is used to improve the OCL usability. Thereafter, the proposed approach by which English sentences are transformed into OCL specifications using transformation rules is elaborated.

Chapter 5, evaluation of the proposed approach, presents the validation of the proposed approach. Furthermore, some analyses are presented in this chapter to determine the impact of the proposed approach on the OCL usability.

Chapter 6, discussion, compares the proposed approach with the existing works. Evaluation metrics are measured for the software tool implementing the proposed approach and the existing tool to compare them together. The impacts of the proposed approach and the existing tool on the OCL usability are measured and compared.

Chapter 7, conclusion and future works, elaborates the research contributions, limitations, and future works.

REFERENCES

Afreen, H., and Bajwa, I. S. (2011). Generating UML Class Models from SBVR Software Requirements Specifications. *In Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011)*. Gent, Belgium: 23-32.

Afreen, H., Bajwa, I. S., and Bordbar, B. (2011). SBVR2UML: A Challenging Transformation. In Proceedings of the Frontiers of Information Technology (FIT). December 19-21. Islamabad, Pakistan: 33-38.

Agrawal, A., Karsai, G., and Shi, F. (2003). Graph transformations on domain-specific models. *Institute for Software Integrated Systems, Vanderbilt University, 2015 Terrace Place, Nashville, TN 37203, Technical Report ISIS-03-403*.

Akehurst, D. H., Boardbar, B., Evans, M., Howells, W. G. J., & McDonald-Maier, K. D. (2006). SiTra: Simple Transformations in Java. 9th International Conference on Model Driven Engineering Languages and Systems (LNCS). 351-364.

Akehurst, D. H., and S.Kent. (2002). A Relational Approach to Defining Transformations in a Metamodel. *Proceedings of the 5th International Conference on The Unified Modeling Language (UML '02)*. Dresden, Germany: 243-258.

Ambriola, V., and Gervasi, V. (2006). On the Systematic Analysis of Natural Language Requirements with CIRCE. *Automated Software Engineering.* 13(1), 107-167.

Amdouni, S., Karaa, W. B. A., and Bouabid, S. (2011). Semantic annotation of requirements for automatic UML Class diagram generation. *The Computing Research Repository (CoRR).* abs/1107.3297.

Appukuttan, B. K., Clark, T., Reddy, S., Tratt, L., and Venkatesh, R. (2003). A model driven approach to model transformation. Workshop on Model Driven Architecture: Foundations and Applications (MDAFA'2003). June 2003. Holland.

Aruna, S., Nandakishore, L. V. (2014). Ensemble Neural Network Algorithm for Detecting Cardiac Arrhythmia. Proceedings of Artificial Intelligence and Evolutionary Algorithms in Engineering Systems (ICAEES 2014), 1, 27-35.

Bajwa, I. S. (2012). A natural language processing approach to generate SBVR and OCL. University of Birmingham.

Bajwa, I. S., Bordbar, B., and Lee, M. G. (2010). OCL Constraints Generation from Natural Language Specification. 14th IEEE International Enterprise Distributed Object Computing Conference (EDOC). Vitoria, Brazil: 204-213.

Bajwa, I. S., Bordbar, B., and Lee, M. G. (2013). SBVR vs. OCL: A Comparative Analysis of Standards. The Computing Research Repository (CoRR). abs/1304.7346.

Bajwa, I. S., Bordbar, B., & Lee, M. (2012). NL2Alloy: A Tool to Generate Alloy from NL Constraints. Digital Information Management 10(6), 365-372.

Bajwa, I. S., and Choudhary, M. A. (2011). From Natural Language Software Specifications to UML Class Models. In 13th International Conference Enterprise Information Systems (ICEIS 2011) (Vol. 102, pp. 224-237): Springer.

Bajwa, I. S., and Lee, M. G. (2011). Transformation Rules for Translating Business Rules to OCL Constraints. In Proceedings of the European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA). Birmingham, UK: 132-143.

Bajwa, I. S., Lee, M. G., and Bordbar, B. (2011). SBVR Business Rules Generation from Natural Language Specification. In AAAI Spring Symposium: AI for Business Agility (pp. 2-8). Stanford, California, USA: AAAI.

Bajwa, I. S., Mumtaz, S., and Samad, A. (2009). Object Oriented Software Modeling using NLP Based Knowledge Extraction. *European Journal of Scientific Research.* 35(1), 22-33.

Biehl, M. (2010). Literature Study on Model Transformations. Royal Institute of Technology, Tech. Rep. ISRN/KTH/MMK.

Bobkowska, A. (2005). Modeling Pragmatics for Visual Modeling Language Evaluation. Proceedings of the Forth International Workshop on Task Models and Diagrams for User Interface Design (TAMODIA 2005). September 26-27, 2005. Gdansk, Poland: 75-78.

Braun, P., and Marschall, F. (2003). The Bi-directional Object- Oriented Transformation Language: Technische Universität München (TUMI0307) o. Document Number)

Bryant, B. R., Lee, B., Cao, F., Zhao, W., Burt, C., Raje, R. et al. (2003). From natural language requirements to executable models of software components. *Proceedings of Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation*. September 24-26, 2003. Chicago, Illinois: 51-58.

Cabot, J. (2006). Ambiguity issues in OCL postconditions. *Proceedings of OCL for (Meta-) Models in Multiple Application Domain (MODELS'06), Technical Report*.

Cabot, J., Clarisó, R., Guerra, E., & Lara, J. d. (2008). An Invariant-Based Method for the Analysis of Declarative Model-to-Model Transformations. 11th International Conference of Model Driven Engineering Languages and Systems (MoDELS 2008). September 28 - October 3. Toulouse, France: 37-52.

Cabot, J., and Teniente, E. (2006). A metric for measuring the complexity of OCL expressions. Workshop on Model Size Metrics (MODELS'06). October 1–6, 2006. Genova, Italy.

Calegari, D., Cengarle, M. V., & Szasz, N. (2008). UML 2.0 Interactions with OCL/RT Constraints. Forum on specification and Design Languages (FDL 2008). September 23-25. Stuttgart, Germany: 167-172

Cam, P. A., Nguyen, M. L., & Shimazu, A. (2011). Study on extracting conceptual structures from legal texts. Japan Advanced Institute of Science and Technology.

Castro, V. D., Marcos, E., and Vara, J. V. (2011). Applying CIM-to-PIM model transformations for the service-oriented development of information systems. *Information and Software Technology*. 53(1), 87–105.

Chimiak-Opoka, J. (2009). OCLLib, OCLUnit, OCLDoc: Pragmatic Extensions for the Object Constraint Language. 12th International Conference on Model Driven Engineering Languages and Systems (MoDELS'09). Oct 4-9. Denver, Colorado, USA: 665–669.

Czarnecki, K., and Helsen, S. (2006). Feature-based Survey Of Model Transformation Approaches. *IBM Systems Journal*. 45(3), 621-646

Czarnecki, K., and Helson, S. (2003). Classification of Model Transformation Approaches. In proceedings of the Workshop on Generative Techniques in the Context of Model-Driven Architecture (OOPSLA 2003). 1-17.

Damljanovic, D., and Bontcheva, K. (2009). Towards Enhanced Usability of Natural Language Interfaces to Knowledge Bases. Web 2.0 and Semantic Web 2009. 105-133.

Dayan, D., Kaplinsky, R., Wiesen, A., & Bloch, S. (2007). AMDA: Matching the Model-Driven-Architecture's Goals Using Extended Automata as a Common Model for Design and Execution. IEEE International Conference on Software-Science, Technology & Engineering (SwSTE 2007). Herzlia, Israel: 1-13.

Dennis, G., Seater, R., Rayside, D., & Jackson, D. (2004). Automating Commutativity Analysis at the Design Level. Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis (ISSTA). July 11-14. Boston, Massachusetts, USA: 165-174.

Duddy, K., Gerber, A., Lawley, M., Raymond, K., and Steel, J. (2004). Model transformation: a declarative, reusable patterns approach. in Proceedings of Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003) 16-19 September 2003. Brisbane, Australien: 174-185.

Duffy, D. (1995). From Chaos to Classes: Object-Oriented Software Development in C++. London ; New York : McGraw-Hill.

Dzidek, W. (2003). Using aspect oriented programming to instrument OCL contracts in Java. Carletoon University, Ottawa, Ontario, Canada.

Figl, K., Mendling, J., and Strembeck, M. (2009). Towards a Usability Assessment of Process Modeling Languages. Proceedings of the 8th Workshop Geschäfsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2009). 26-27 November 2009. Berlin, Germany: 118-136.

Fliedla, G., Kopa, C., Mayra, H. C., Salbrechtera, A., Vöhringera, J., Webera, G. et al. (2007). Deriving static and dynamic concepts from software requirements using sophisticated tagging. *Data and Knowledge Engineering.* 61(3), 433-448.

Grünbacher, P., Egyed, A., and Medvidovic, N. (2004). Reconciling software requirements and architectures with intermediate models. *Software and System Modeling.* 3(3), 235-253.

Harmain, H. M., and Gaizauskas, R. J. (2003). CM-builder: A natural language-based CASE tool for object-oriented analysis. *Automated Software Engineering.* 10(2), 157-181.

Hatebur, D., & Heisel, M. (2010). Making Pattern- and Model-Based Software Development more Rigorous. Proceedings of the 12th international conference on Formal engineering methods and software engineering (ICFEM 2010). November 17-19. Shanghai, China: 253-269.

Huber, P. (2008). The Model Transformation Language Jungle - An Evaluation and Extension of Existing Approaches. Vienna University of Technology.

Ilieva, M. G., and Ormandjieva, O. (2006). Models Derived from Automatically Analyzed Textual User Requirements. *Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications (SERA 2006)*. 9-11 August 2006. Seattle, Washington, USA: 13-21.

Jilani, A. A. A., and usman, M. (2010). Model Transformations in Model Driven Architecture: A Survey". *IEEE 2nd International Conference on Education Technology and Computer (ICETC)*. Shanghai, China.

Jolliffe, I. T. (2002). Principal Component Analysis. New York: Springer.

Jouault, F., Allilaire, F., Bézivin, J., and Kurtev, I. (2008). ATL: A model transformation tool. Science of Computer Programming. 72(1-2), 31–39.

Judson, S. R., Carver, D. L., and France, R. B. (2003). A Metamodeling Approach to Model Transformation. *Object Oriented Programming, Systems, Languages, and Applications (OOPSLA '03)*. October '03. Anaheim, California, USA: 326 – 327.

Kalnins, A., Barzdins, J., and E. Celms. (2004). Basics of Model Transformation Language MOLA. *in Proceedings of Model-Driven Architecture: Foundations and Applications (MDAFA 2004)* Twente, The Netherlands / Linköping, Sweden: 14—28.

Kardoš, M., and Drozdová, M. (2010). Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA). Information and Organizational Sciences (JIOS). 34(1), 89-99.

Ketfi, & Belkhatir. (2005). Model-driven framework for dynamic deployment and reconfiguration of component-based software systems. Metainformatics International Symposium (MIS) Esbjerg, Denmark.

Kleppe, A., Warmer, J., & Bast, W. (2003). MDA Explained: The Model Driven Architecture™: Practice and Promise. Addison-Wesley Professional.

Kumar, D. D., and Sanyal, R. (2008). Static UML Model Generator from Analysis of Requirements (SUGAR). *Advanced Software Engineering Applications (ASEA)*. December 13-15. Hainan Island, China: 77-84.

Lano, K., Kolahdouz-Rahimi, S., and Poernomo, I. (2012). Comparative Evaluation of Model Transformation Specification Approaches. Software and Informatics. 6(2), 233-269.

Li, K., Dewar, R. G., and Pooley, R. J. (2005). Object-Oriented Analysis Using Natural Language Processing. *Linguistic Analysis*.

Manterea, T., and Alander, J. T. (2005). Evolutionary software engineering. Journal of Applied Soft Computing. 5(3), 315–331.

Mens, T., & Gorp, P. V. (2005). A Taxonomy of Model Transformation. Proc. Int'l Workshop on Graph and Model Transformation (GraMoT 2005). Tallinn, Estonia.

Mich, L. (1996). NL-OOPs: From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA. *Natural Language Engineering*. 2(2), 161-187.

Nikseresht, A., & Ziarati, K. (2011). MDA Based Framework for the Development of Smart Card Based Application. International MultiConference of Engineering and Computer Science (IMEC 2011). March 16-18. Hog Kong: 263-268.

Njonko, P. B. F., and El Abed, W. (2012). From natural language business requirements to executable models via SBVR in Proceeding of the International Conference on Systems and Informatics (ICSAI 2012). 19-20 May. Yantai 2453-2457

OMG. (2013). Meta Object Facility (MOF) Core Specification, v2.4.1

OMG. (2012). OMG Object Constraint Language (OCL). OMG Document Number: formal/2012-01-01.

OMG. (2013). Semantics of Business Vocabulary and Business Rules (SBVR), v1.2.

Osis, J., Asnina, E., & A., G. (2008). Computation Independent Representation of the Problem Domain in MDA. Software Eng. 2(1), 19-46.

Overmyer, S. P., Lavoie, B., and Rambow, O. (2001). Conceptual Modeling through Linguistic Analysis Using LIDA. *Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001)*. 2002-12-17. Toronto, Ontario, Canada: 401-410.

Perez-Gonzalez, H. G., and Kalita, J. K. (2002). *GOOAL: A Graphic Object Oriented Analysis Laboratory*. In *Companion of the 17th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications* (pp. 38-39). Seattle, Washington, USA: ACM.

Raj, A., Prabhakar, T. V., and Hendryx, S. (2008). Transformation of SBVR Business Design to UML Models. *Proceedings of the 1st India software engineering conference (ISEC 2008)*. ACM New York, NY, USA: 29–38.

Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2007). CIM to PIM Transformation: A Reality. *International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2007)*. Beijing, China: 1239-1249.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1992). Object-oriented modeling and design. USA: Pearson Education.

Scheidgen, M. (2006). Model Patterns for Model Transformations in Model Driven Development. Proceedings of the Joint Meeting of The Fourth Workshop on Model-Based Development of Computrer-Based Systems and The Third International Workshop on Model-based Methodologies for Pervasive and Embedded Software, MBD/MOMPES 2006. March 30. Potsdam, Germany: 149-158.

Schürr, A. (1994). A visual language and environment for programming with graph rewrite systems. *RWTH Aachen, Fachgruppe Informatik, , Technical Report AIB 94-11.*

Seco, N., Gomes, P., and Pereira, F. C. (2004). Using CBR for Semantic Analysis of Software Specifications. *Advances in Case-Based Reasoning, 7th European Conference (ECCBR 2004)*. Madrid, Spain: 778-792.

Selway, M., Grossmann, G., Mayer, W., & Stumptner, M. (2013). Formalising Natural Language Specifications using a Cognitive Linguistics/Configuration

Based Approach. 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2013). Vancouver, BC, Canada: 59-68.

Sharaff, A. (2013). A Methodology for Validation of OCL Constraints Using Coloured Petri Nets. International Journal of Scientific & Engineering Research (IJSER). 4(1).

Sharma, M., & Vishwakarma, R. G. Formalization & data abstraction during use case modeling in object oriented analysis & design. 3th International Conference on Computer Science, Engineering & Applications (ICCSEA 2013). May 24-26. Delhi, India: 67–75.

Standardization, I. O. f. (1998). Ergonomic requirements for office work with visual display terminals (VDTs). Guidance on usability.

Störrle, H. (2013). Improving the Usability of OCL as an Ad-hoc Model Querying Language. In 13th International Workshop on OCL, Model Constraint and Query Languages (OCL@MoDELS) (Vol. 1092, pp. 83-92). United States, Miami, Florida: CEUR-WS.org.

Taentzer, G., Arendt, T., Ermel, C., & Heckel, R. (2012). Towards refactoring of rule-based, in-place model transformation systems. Proceedings of the First Workshop on the Analysis of Model Transformations. 41-46.

Tratt, L. (2006). The MT model transformation language. *in Proceedings of the ACM Symposium on Applied Computing (SAC 2006)*. April 23-27, 2006. Dijon, France: 1296-1303.

Vela, B., Fernández-Medina, E., Marcos, E., & Piattini, M. (2006). Model driven development of secure XML databases. Special Interest Group on Management of Data (SIGMOD). 35(3), 22-27.

Vlist, E. V. D. (2003). Relax Ng (Simplification and Restrictions). Oreilly & Associates Inc.

Wahler, M. (2008). Using Patterns to Develop Consistent Design Constraints. ETH Zurich, Switzerland.

Wang, M. (2013). Requirements Modeling: from Natural Language to Conceptual Models Using Recursive Object Model (ROM) Analysis. Concordia University, Montreal, Quebec, Canada.

Warmer, J., & Kleppe, A. (1999). Object Constraint Language: Precise Modeling with UML. Addison Wesley.

Warmer, J., and Kleppe, A. (2003). The Object Constraint Language: Getting Your Models Ready for MDA. Boston, MA, USA: Addison-Wesley Longman.

Wilke, C., Thiele, M., & Wende, C. (2010). Extending Variability for OCL Interpretation. 13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010). October 3-8. Oslo, Norway.

Yue, T., Briand, L. C., and Labiche, Y. (2011). A Systematic Review of Transformation Approaches between User Requirements and Analysis Models. Springer Requirements Engineering. 16(2), 75-99.