S. ALIZADEH TABRIZI

ATILIM UNIVERSITY 2021

A SYSTEMATIC MAPPING ON THE QUALITY AND UNDERSTANDABILITY OF UML DIAGRAMS

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

SINA ALIZADEH TABRIZI

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JULY 2021

A SYSTEMATIC MAPPING ON THE QUALITY AND UNDERSTANDABILITY OF UML DIAGRAMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

SINA ALIZADEH TABRIZI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JULY 2021

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Software Engineering, Atılım University**.

_____

Prof. Dr. Ali YAZICI
Head of Department

This is to certify that we have read the thesis **"**A SYSTEMATIC MAPPING ON THE QUALITY AND UNDERSTANDABILITY OF UML DIAGRAMS**"** submitted by **"**SINA ALIZADEH TABRIZI**"** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Asst. Prof. Dr. Damla Topalli Co-Supervisor	Assoc. Prof. Dr. Nergiz Ercil Çağiltay Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Erol Ozçelik Department of Psychology, Çankaya University	_____
Assoc. Prof. Dr. Nergiz Ercil Çağiltay Software Eng. Department, Atılım University	_____
Asst. Prof. Dr. Damla Topalli Computer Eng. Department, Atılım University	_____
Asst. Prof. Dr. Bilge Say Software Eng. Department, Atılım University	_____
Asst. Prof. Dr. Gül Tokdemir Computer Eng. Department, Çankaya University	_____

Date: July 9, 2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Sina Alizadeh Tabrizi

Signature :

ABSTRACT

A SYSTEMATIC MAPPING ON THE QUALITY AND UNDERSTANDABILITY OF UML DIAGRAMS

Alizadeh Tabrizi, Sina
MSc., Department of Software Engineering
Supervisor : Assoc. Prof. Dr. Nergiz Ercil Cagiltay
Co-Supervisor : Asst. Prof. Dr. Damla Topalli

July 2021, 89 pages

Nowadays, the concept of quality is an integral component of any tangible and intangible product and service. In the field of software, the quality is not merely confined to the final product, but it should be applied to all phases of the software development life cycle (SDLC). The Model-driven Software Development (MDSD) necessitates the experts with high skill to fabricate quality modelling artifacts in order to deliver an authentic perception of the desired system. One of the de facto modeling languages is Unified Modelling Language (UML) which is commonly used in the sphere of software engineering to model the system's static and dynamic behaviors, to discern the system structure and to expedite the coding and communicating with the stakeholders on the required functional and non-functional requirements. The primary objective of the present thesis is to present the state-of-the-art on the quality and understandability of UML notation. Therefore, a systematic mapping was applied on 103 journal papers extracted from Web of Science. The results indicated that quality is the main aspect in such a way that other aspects like understandability and maintainability are subsets of this aspect. According to the reviewed studies 13 factors have been found to effective on the quality and understandability, namely, diagram, diagram type, evaluation, guideline, modelling, quality assessment approach, redundancy, refactoring, representations, requirements, software engineer (identified by modelling expertise, knowledge, experience, cognitive abilities, learning style, learning motive, learning strategy) structure and testing.

Keywords: Quality, Understandability, Unified Modelling Language, Systematic Mapping, Model-driven Software Development.

ÖZ

UML DİYAGRAMLARININ KALİTESİ VE ANLAŞILABİLİRLİĞİ ÜZERİNE SİSTEMATİK BİR HARİTALAMA ÇALIŞMASI

Alizadeh Tabrizi, Sina
MSc., Department of Software Engineering
Tez Yöneticisi : Doç. Dr. Nergiz Ercil Cagiltay
Ortak Tez Yöneticisi : Dr. Öğr. Üyesi Damla Topalli

Temmuz 2021, 89 sayfa

Günümüzde kalite kavramı, maddi ve manevi her türlü ürün ve hizmetin ayrılmaz bir bileşenidir. Yazılım alanında kalite yalnızca nihai ürünle sınırlı değildir, yazılım geliştirme yaşam döngüsünün (SDLC) tüm aşamalarına uygulanmalıdır. Modele Dayalı Yazılım Geliştirme (MDYG), istenen sistemin otantik bir algısını sağlamak ve kaliteli modelleme yapaylıkları üretmek için yüksek beceriye sahip uzmanlık gerektirir. Birleşik Modelleme Dili (UML), yazılım mühendisliği alanında sistemin statik ve dinamik davranışlarını modellemek, gerekli işlevsel ve işlevsel olmayan gereksinimleri belirlemek, kodlamayı, sistem yapısını, ve sistem üzerindeki paydaşlarla iletişimi hızlandırmak için yaygın olarak kullanılan bir modelleme dilidir. Mevcut tezin temel amacı, UML notasyonu kalitesi ve anlaşılabilirliği konusunda en son teknolojiyi sunmaktır. Bu nedenle Web of Science veritabanından elde edilen 103 dergi makalesi üzerinde sistematik bir haritalama çalışması yapılmıştır. Sonuçlar, incelenen alanda kalitenin ana unsur olduğunu gösterirken, anlaşılabilirlik ve sürdürülebilirlik gibi diğer özellikler kalitenin alt kümeleri olarak değerlendirilmiştir. İncelenen çalışmalara göre, kalite ve anlaşılırlık üzerinde temel olarak 13 faktörün etkili olduğu görülmektedir. Bu faktörler diyagram, diyagram tipi, değerlendirme, kılavuz, modelleme, kalite değerlendirme yaklaşımı, artıklık, yeniden düzenleme, gösterimler, gereksinimler, yazılım mühendisi (modelleme uzmanlığı tarafından

tanımlanan), bilgi, deneyim, bilişsel yetenekler, öğrenme stili, öğrenme güdüsü, öğrenme stratejisi) yapı ve test etme olarak belirtilmiştir.

Anahtar Kelimeler: Kalite, Anlaşılabilirlik, Birleşik Modelleme Dili, Sistematik Haritalama, Modele Dayalı Yazılım Geliştirme.

This Research is dedicated to my parents. For me you sacrificed much, supported unconditionally and loved without limit. Thank you.

ACKNOWLEDGMENTS

I would like to express my special thanks and sincere gratitude to Dr. Nergiz Ercil Cagiltay and Dr. Damla Topalli for their major helps and guidance throughout the research, and for their immense patience.

I shall also thank to the respectful jury members who assigned their valuable time to review the present research.

Furthermore, I thank the members of department of software engineering.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

BPMN	Business Process Model and Notation
UML	Unified Modeling Language
SRS	Software Requirements Specification
MDSD	Model-Driven Software Development
FOOM	Function and Objected Oriented Methodology
OPM	Object Process Methodology
MDD	Model-Driven Development
GDM	Good Decomposition Model
UCD	Use Case Diagram
OO-DFD	Object-Oriented Analysis with Data Flow Diagram
OML	Open Modelling Language
AD	Activity Diagram
EPC	Event Driven Process Chain
VCL	Visual Contract Language
AOI	Areas of interest
LoD	Level of Detail
OCL	Object Constraint Language
WOL	Web Ontology Language
FIRP	Feedback-inclusive rapid prototyping method
OOA	object-oriented analysis
OOLD	Object-oriented logical design
CHOP	Chunking, Ordering and Patterning
SD	Sequence Diagram
FSM	Functional Size Measurement
PBR	Perspective-based Reading
CBR	Checklist-based Reading
VACD	Vertically Aligned Class Diagram
CASE	Computer-Aided Software Engineering

MIS	Management Information System
ER	Entity-relationship
HITS	Hyperlink-Induced Topic Search
QAA	quality Assessment Approach
RTES	Real-time Embedded Software
MARTE	Modeling and Analysis of Real-Time and Embedded
CSP	Constraint Satisfaction Problem
ETL	Extract, Transfer and Load
FR	Functional Requirement
WAE	Web Application Extension
DAM	Dependability Modeling and Assessment
CLT	Cognitive Load Theory
CMM	Conceptual Mental Model
SCC	Similarity-based class cohesion (SCC)
AOM	Aspect-oriented modelling (AOM)
LCD	Learner-centered design
PCDL	Post-/frame condition determining language
ER	Entity-relationship

CHAPTER 1

INTRODUCTION

It is deemed that the analysis phase is one of the significant steps of the software development lifecycle, during which an analyst endeavors to model the business process using two prominent notations, namely, Business Process Model and Notation (BPMN) and Unified Modeling Language (UML). The quality of the final products is realized through applying quality assessment to all phases of software development, implying that it is highly important to assess the quality of the constructed models, to be assured that the diagrammatical representations reflect the system in a proper manner, which can assist the implementation team to code a correct system, whose functions can be traceable against to the software requirements specification (SRS).

Indeed, the developers are able to discern the system's behavior, structure and its elements by reviewing the modelling phase artifacts. These are models that are used to communicate with stakeholders, via which the analysis and development teams can document the system both functional and non-functional requirements. The textbooks on modelling merely focus on the diagrams and their description, and the academic training materials are just confined to educate the students with basic notation rules, while the Model-Driven Software Development (MDSD) requires experts with high skill in producing quality designs and models which deliver an accurate and efficient perception of the system.

It is deemed that "software is not only difficult to be created but it is also hard to be understood" [1]. It is not only confined to the software, but it is also true for the models, especially for the design models. One of the first step to solve the understandability problem in the software field was to introduce another artifact, named, documentations. Both software code and the related design models should carry an intent, since the level of the intent expressed by the code influences the simplification of the error discovery and any deviation from or adherence to the functionality that the application should exhibit. The preservation of a comprehensible intent is significant and the source of the intent in the models is the organization people [1]. Hence, the people are crucial for the generating comprehensible software designs. An earlier study report that the agile and lean approaches have an advantage of bringing the stakeholders together to have a face-to-face contact for facilitating and expediting the

communication which is effective in terms of time, cost and resources [2]. According to the study conducted by Kabeli and Shoval, today Unified Modelling Language (UML) is the indispensable modelling language used in software engineering discipline, since UML adopts multiplied diagram types to cover all aspects of the modelled system, while function and objected oriented methodology (FOOM) and object process methodology (OPM) used only limited number of diagram types [3].

Besides its reported importance in the literature, there is a research gap on the state-of-the-art of UML diagrams quality and understandability. Quality of UML diagrams can be defined as the level of absence of defects and errors in a UML mode induced by a conflict between different views or the absence of norms for using the UML. Understandability of the UML diagrams can be defined as the correctness and completeness of the design models facilitating the communication between the stakeholders to discern the system required functional and non-functional needs and the correct reflection of the business process in a real world, which help the user to recognize problem domain information and extend their understandability in problem solving. Understandability is a cognitive process, [and] it is difficult to directly observe it and tests to measure participant's performance were conducted to assess the level of understanding cognitively developed by each participant. A few systematic reviews are available on quality and comprehensibility of other notations diagrams, as well as UML notation, however the modelling notations scope examined by the preceding reviews is wide, so they have not attained to a comprehensive finding on UML modelling language. It is worth noting that by considering the available literature on the topic it can be observed that the border between quality and understandability for UML models is not well specified, as the existing studies try to study the UML diagrams from various aspects, such as layouting, aesthetics, effectiveness, readability, and comprehensibility. Accordingly, this study aims at scrutinizing state-of-the-art for UML diagrams in terms of quality, comprehension, and factors effective on the understandability, by conducting a systematic mapping approach.

In what follows, chapter 2 provides a summary of the related studies, then in chapter 3 research methodology and research questions are elaborated. Afterwards, the answers along with findings synthesis are provided in chapters 4 and 5, respectively. In chapter 6, the major findings and suggestions for prospective studies are presented.

CHAPTER 2

BACKGROUND

In the preceding chapter, the preliminary information regarding the problem definition, study objectives, novelty of the present research and significance of the study were provided. However, prior to the present study, several studies have been carried out on the UML diagrams understandability, visualization, quality, current practices for high quality, empirical studies on UML notation, empirical research on model-driven development using UML, consistency management, maintainability, complexity metrics by applying systematic literature review approach or other research methodology which are elaborated in subsections 2.1 and 2.2. The presented chapter is devoted to exploring and explain the studies with a same or at least similar topic, and revealing their differences and in some cases, their superiorities, and deficiencies, particularly in terms of the research design and methodology. To this end, the available literature on the systematic review of the papers on UML modelling language are critically reviewed and this chapter is merely confined to present the pros and cons.

2.1. Preceding miscellaneous studies

Kabeli and Shoval [3] have compared two modelling languages namely, functional and objected-oriented methodology (FOOM) and object process methodology (OPM) in terms of comprehension and quality [3]. They conducted two controlled experiments, considering two perspectives, including users' perspectives and analysts' perspective [3]. For the users, they focused on the comprehension, while for the analysts they focused on the quality, in terms of correctness [3]. They observed that FOOM specifications are more understandable [3]. They carried out systematic literature review, during which 43 articles were considered and the complexity metrics for process diagrams were derived [3]. In this part, the studies, which have experimentally and theoretically studied the UML notations from different perspectives, are reviewed. An earlier study states that for abstract representing and communication of the domain models understating between the stakeholders, conceptual models are used as the useful means [4]. Indeed, they state that the conceptual models is consisted of modelling constructs in the form of graphical

symbols and text [4]. The effectiveness of the modelling language has been the main consideration of the previous studies [5]. In that study, the authors attempted to study the cognitive process during the perception of the conceptual models by performing two experimental studies to reveal the effective role of the eye tracker in understanding the cognitive process and the parts, which are focused by the engineers [5]. In that study, a combination of eye tracker and verbal protocol analysis was used to comprehend how the visual association between the segments of the conceptual modelling scripts can be a proper metric and measure of the cognitive integration during the problem solving [5]. Their findings showed that the scripts developed based on the different rules trigger different cognitive processes [5].

2.2. Preceding systematical literature review studies

One of the comparable study to the present research was conducted by [6] who sought to conduct a systematic literature review on process models for listing the factors effective on the understandability. They searched for the studies between 1995 and 2015 and the selected 45 publications for full review. Similar to the present thesis, [6] they also used MS excel file for storing the publication information. Their SLR method is consisted of six main steps, defining the search problem, defining the objective and questions, conducting a pilot study, defining a search string, identifying the data sources and defining inclusion and exclusion criteria. The main difference between our study and theirs' is related to the search string and they have just focused on process diagrams of UML notation, while we have studied the papers focused on both conceptual models and process models. Moreover, their search string included only four key terms as below:

((Understandability OR comprehension) AND ("process model" OR UML))

In addition, they selected six major electric sources, which increase the risk of duplication and the effort needed for excluding the repeated papers. It should be also noted that they only analyzed the empirical studies, while we have reviewed both theoretical and experimental papers.

The other study carried out by [7] was a systematic literature review, but they focused on software visualization evaluation. They reviewed 387 full papers indexed in SOFTVIS/VISSOFT conference. Their main objectives in answering the research questions were to identify the characteristics of evaluation that verify the effectiveness of the software visualization approaches and the suitability of the available evaluations

approaches for checking the effectiveness of the software visualization. They followed the guideline for the systematic literature review proposed by Kitchenham [8]. Unfortunately, their inclusion and exclusion criteria were so narrow compared to the inclusion and exclusion criteria of the presented study. They extracted 178 papers for full review among 387 papers. They have not clearly defined according to which criteria they have selected their corpus. Another main difference of their study was related to the fact that they assessed the quality of the papers for review and classified their corpus into five main categories, including evaluations, design studies, systems, techniques, and models. Seemingly, in their search string there were no Boolean operations, like AND or OR, and they have not presented the structure of the search string they adopted, while they presented a list of key terms that they have searched as below:

> *{evaluation, survey, [case|user] stud[y|ies], [application usage | analysis] example[s], use case[s], application scenario[s], [controlled | user] experiment, demonstration, user scenario[s], example of use, usage scenario[s], example scenario[s], demonstrative result[s]}*

Mohaghegghi et al. [9] carried out a similar study by performing a SLR on the studies whose main concern was the quality of models to gain an insight into what is the meaning of quality for the models and what are the methods to enhance it. Their corpora were consisted of 40 studies and found that the models quality can be studied in terms of six aspects. Similar to the present study, they followed the SLR guideline presented by Kitchenham et al. [8]. One of the major superiority of the present study in comparison with Mohagheghi et al. [9], is the time period of our study, as we have reviewed the available papers for 20 years, while their corpora were limited to the period of 2000 till 2007. One other drawback of their study is the selection of almost 12 sources, which cover both journal papers and conferences papers, while at the same time increases the risk of duplication and the need for extra effort to exclude the replications. It should be also noted that they have not clearly presented their inclusion and exclusion criteria, as well as their search string and the SLR steps that they have taken.

El-Attar and Miller [10] studied the quality of use case models by applying SLR. They divide the quality of use case models into five major categories, namely consistency, correctness, and completeness, fault-free, analytical and understandability, which is not in line with the factors and quality attributes identified in the present thesis. They

extracted their corpus from 11 electronic databases, which may lead to an increase in the number of duplicate papers, as most of the Scopus journals with quartile of Q_2 or Q_1 are also journals with impact factor in World of Science and their associated papers are indexed in two databases at the same time. Their search string lacked two important key terms of comprehensibility and understandability:

> *Use Case; Use Cases; Use Case AND (Actor OR Model OR Description OR Text OR Textual OR Diagram OR Authoring OR Practices OR Quality OR Inspection OR Heuristics OR Guidelines OR Anti-patterns OR OR Modeling OR Mistake OR Problem OR Drawback OR Recommendation OR Suggestion OR Warning OR Rule OR Syntax OR Metamodel OR Appropriate OR Pitfalls OR Proper OR Patterns OR Blueprints OR UML OR OMG).*

Similar to the present thesis, El-Attar and Miller [10] have clearly specified the applied inclusion and exclusion criteria. It should be noted that their corpus was consisted of 92 papers obtained from various sources, including sources—journals (14), conferences (41), workshops (16), books (20) and official syntax reference (1).

The main advantage of their study over the present research was related to the procedure they adopted for quality assessment of their corpus, by considering following screening criterion [10]

> **c1.** Would the application of a recommended practice lead to a syntactically correct UC model?
>
> **c2.** Is the paper based upon research?
>
> **c3.** Is the reported practice clearly identified and presented?
>
> **c4.** Is the reported recommended practice validated in practice or merely a ‘‘lessons learned’’ report based on expert opinion?
>
> **c5.** Has the recommended practice been empirically validated?
>
> **c6.** Is the reported practice of value to future research?

In the study by Budgen et al. [11], the authors adopted SLR to reveal that how widely the notation of UML and their advantages have been surveyed empirically and to also classify the found studies in terms of the aspects of the UML that they have studied. Their study period was so narrow, and it was between 2000 to 2008. They have clearly declared both inclusion and exclusion criteria, while they have used four electronic data bases, namely, IEEE, ACM, ScienceDirect and EMBASE. It is worth noting that in their search string they have tried to cover both British and American spelling like

the term *Modelling* and *Modeling*. They have also used the process of snowballing to extract more papers.

Haug et al. [12] carried out a systematical review on 21 empirically studies on the model-driven development (MDD) with UML, published between 2000 to 2005 in the prominent journals and conference proceedings, the main aim was centered on obtaining an overview on the state-of-the-art for the empirical research on MDD with UML. In that study, Kitchenham's guideline for systematical review was adopted. They believe that the systematic review consisted of several discrete activities as below:

- Plan a review
- Conduct the review
- Report the review

During the planning two main tasks, namely, identification of the need for reviewing and establishment of review protocol are accomplished.

In the study done by Haug et al. [12], the enquired search string is not presented. They have just provided the key terms used during the search for publications in the journals and proceedings. These keywords were: UML, unified modelling language, use cases, diagram, analysis and design and other words related to UML diagrams or other UML constructs. Moreover, in the inclusion criteria, the language of the publications is filtered, and they have included the key terms as the inclusion and exclusion criteria.

In a study by Lucas et al. [13], a systematic review was applied on 42 studies in terms of UML models in terms of consistency management. In that study, the authors sought to discover the various model's consistency conceptions, proposals, problems and the proposed solutions. That study corpora were extracted from IEEE Computer Society, ACM Digital Library, Google Scholar, Science Direct and the Scopus Database. As the search string, they used following string:

> *(''management" AND ''model" AND (''inconsistency" OR ''consistency")) OR (''model" AND (''inconsistency" OR ''consistency"))*

Seemingly, the search string could have included more key terms to cover all studies related to the consistency and inconsistency management, like maintenance, artifact, designs etc. They point out that some synonyms like diagram, view and concern were used, but they have not provided the full search string for these synonym key terms. In the inclusion and exclusion criteria, the time for which the papers were extracted from the scientific databases is not declared. Furthermore, in the exclusion criteria, it is

stated that the short papers were excluded, without giving a specific definition for the short papers and they have not provided the timeline for the research, that is when the study was commenced and how long did it take to extract and review the papers.

Fernández-Sáez et al. [14] also conducted a SLR study in which they focused on the maintenance of UML diagrams. A systematical mapping was conducted to gain some knowledge on the empirical evident related to the adoption of UML diagrams for maintaining of the source code and the maintenance of the diagrams themselves. In that study, 38 papers were reviewed. For Fernández-Sáez et al. [14], the maintenance refers to the modification of a software product after its delivery to the end users for applying corrections on the faults in order to enhance the performance and other features or include other functionalities. It is deemed that two important tasks are included in the maintenance, comprehending the software artifact, that is, to modify a program, the programmers need to discern the functional requirements. The second task is modification of the software artifact, which means that for incorporating the necessary changes, a maintenance engineer needs to create, modify, and validate the data structures, interfaces and documentation. In fact, according to ISO 25000, these two tasks are sub-characteristics of the maintainability. They have searched terms "UML", "maintenance" and "Empirical" as the main search terms along with terms synonyms, like unified modelling language, Maintainability, Modularity, Reusability, Analyzability, Changeability, Evolution, Evolvability, Modification, Stability, Testability, Comprehensibility, Comprehension, Understandability, Understanding, Misinterpretation, Experiment, Survey, Case study, Action research. They applied the search on six digital libraries, including SCOPUS database, Science@Direct with the subject Computer Science, Wiley InterScience with the subject of Computer Science, IEEExplore, ACM Digital Library and SPRINGER databases. The search period was set to 1997 to 2010. They reviewed publications including journals, conferences, and workshops papers. As they have selected multiple databases, the risk of finding the duplicate papers was high. Moreover, they have not presented the search string.

Polančič and Cegnar et al. [15] carried out SLR, during which 43 articles were considered and the complexity metrics for process diagrams were derived.

CHAPTER 3

METHODOLOGY

3.1. Methodology

In order to achieve an in-depth perception on the topic, a systematic mapping method is employed during which only journal articles are investigated. Boland, Cherry and Dickson [16] define systematic review as a literature review used for identifying, evaluating and synthesizing the most proper contents from which the answer of research questions can be derived. In fact, it is a gold standard with a main focus on synthesizing the findings of the preceding studies studied the related topic. Using this method, scholars can gain an insight into the current state of knowledge on a specific topic, which may be used for answering questions, like "what are the most common applications of genetic or evolutionary algorithms in hydraulic engineering discipline?" or "Which algorithms are used for analyzing the mental load of the analyst during diagram review task".

It is essential to utilize a rigorous and accountable methods in a research [17]. That is, systematic review as a secondary research gathers the findings of the primary studies [17]. In a SLR, the focus may be on reviewing the statistical data to find the questions' answer or qualitative data for devising new theories and concepts, and the role of systematic review as an important components for evidence-informed decision making is undeniable [17].

3.2. Research questions

The research questions of the present review study are as below:

RQ1. Which factors are studied for the quality and understandability of UML diagrams during 2000-2020?

As the quality and understandability of UML diagrams are realized by full filling more than one quality attributes, it is important to identify the factors which have an impact on the quality and understandability of UML diagrams. To this end, RQ1 is the main question of the study and the answer to this question should be noted as the primary contribution of the thesis to the field.

RQ2. From which perspectives the quality and understandability of UML diagrams are studied during 2000 to 2020?

The main difference between the systematic literature review and systematic mapping lies in the fact that SLR is more question-centered, while one of the main objectives of mapping study is to gain information on the state-of-the-art and reveal the studied perspectives for a specific field. In this regard, RQ2 question is designed to identify the studies perspectives for UML diagrams.

RQ3. Is understandability a subset of quality in UML diagrams?

In the literature, the border between two aspects of UML diagrams, namely quality and understandability is not well-defined and sometimes merely understandability of UML models may be noted as the quality model. By formulating this question, it was aimed to find out the relationship between the understandability and quality in UML diagrams.

3.3. Research design

To provide the responses for the raised questions, authors applied systematic mapping on the journal papers extracted from Web of Science. The search engine of Web of Science came up with 3.236 articles consisted of both journal and conference papers during 7 July 2020 until 19 August 2020. The search results were in terms of paper titles by applying inclusion and exclusion criteria and DOI links of the related studies were saved and used for downloading the papers full text. From 2 September 2020 until 25 September 2020, by applying systematic mapping phases elaborated in the Figure 1, on the downloaded articles, the number of articles were mitigated from 396 to 103. It should be noted that there were 2 replications, 13 unrelated papers, and 76 semi-related papers and 9 papers on systematical literature review, which are excluded from the corpora and some of them are cited in introduction, literature review and discussion sections (see Figure 1).

The steps followed for systematic mapping can be summarized as below:

Step 1: creating search string

Step 2: searching the string created in step 1, using Web of Science

Step 3: screening the title and abstract of found papers

Step 4: applying inclusion and exclusion criteria

Step 5: applying systematic mapping on abstract Sections

Step 6: reading introduction, results, and conclusion for 70 articles

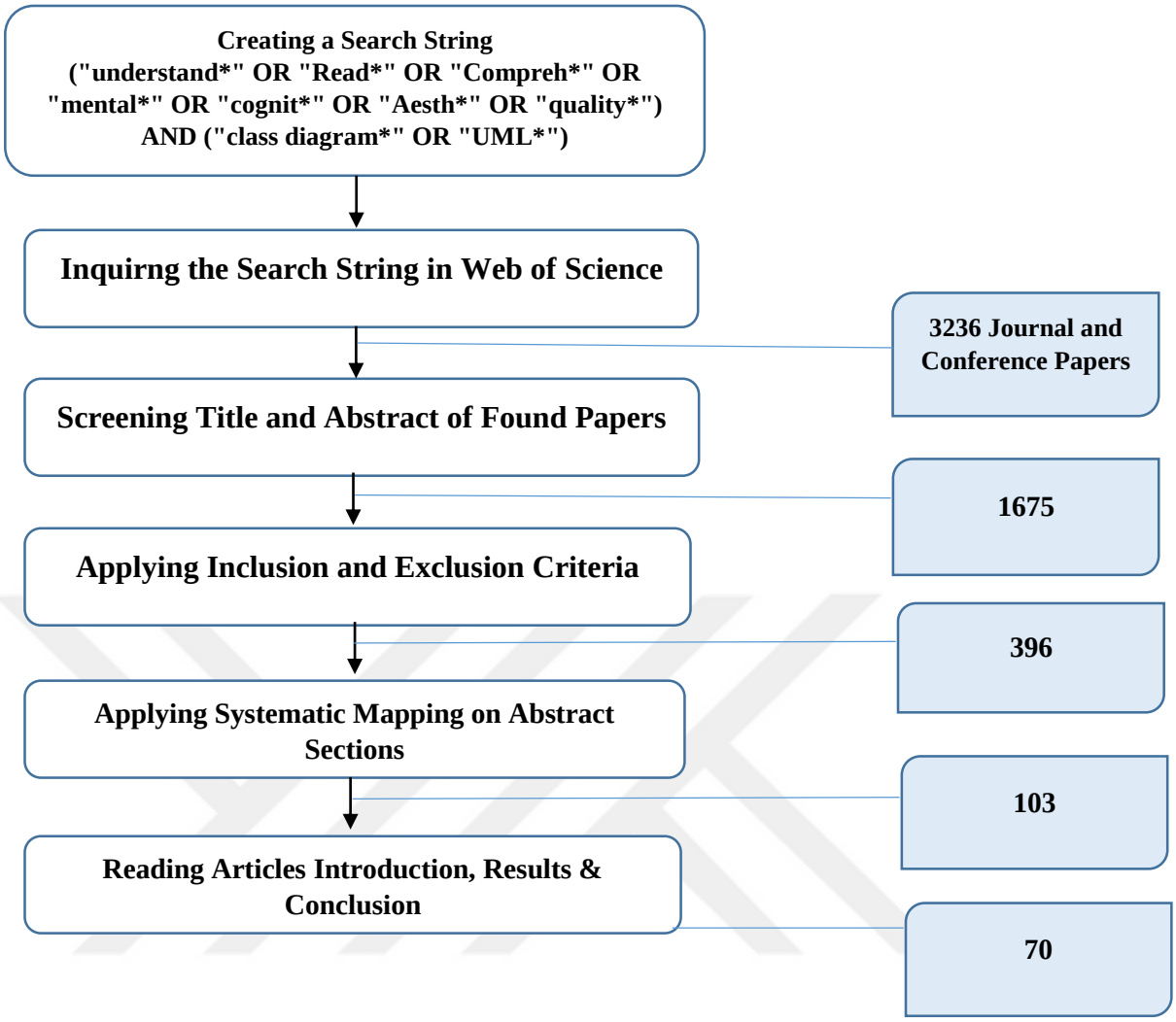

Figure 3.1 Systematic mapping steps used in this study

Inclusion Criteria:

1) Publication year after 2000
2) Article written in English
3) Publications that investigate the understandability of UML diagrams
4) Publications that investigate both conceptual models and process models (UML)

Exclusion Criteria:

1) Articles without journal, issue, etc. information
2) Articles do not mention their novelty
3) Articles published in conferences are excluded

The inclusion and exclusion criteria were chosen in a manner that the majority of the related studies to be found and also to narrow the study scope in terms of time

period, moreover to increase the feasibility of the research within 1 year, and due to these considerations, the conferences papers were excluded, since the number of the conference papers on the study topic was extremely high and inclusion of the conference paper might have led to fiasco in accomplishing the research. In addition, the papers available in other languages, except for English were completely excluded to maximize the feasibility of the research in a timely manner. To encompass all UML diagrams, the studies investigated both process and conceptual models were considered. The articles with missing meta-data, like journal title, issue or volume number were excluded to merely focus on the authentic papers. Moreover, the studies which have novelty in terms of the adopted methodology or findings were reviewed to reveal the state-of-the-art on the studied topic.

CHAPTER 4

RESULTS

In the present study, 103 articles from journals which are published between 2000-2020 undergone through a systematic mapping merely by scrutinizing the abstract section of the articles. In answering three research questions, some articles' abstract section lacked required information and details to find the answer for the questions. In these cases, introduction, results, and conclusion sections of the papers were fully studied to provide more precise and authentic details.

4.1. Factors effective on quality and understandability of UML diagrams

The first research question of the present study was *RQ1. Which factors are studied for quality and understandability of UML diagrams during 2000-2020?*

In order to answer this research question, each article in the corpus of this study is analyzed for identifying the factors that are studied considering the quality and understandability of UML diagrams. After the identification of each factor, they are classified under some groups and at the end 13 factors have been identified addressed by the articles of the present study corpora.

The 13 factors are grouping of the detailed secondary factors, all of which were extracted from the title and abstract section of the reviewed studies. Indeed, all these factors are explicitly stated in the title and abstract sections of the article. The authors of these studies have investigated the effect of one or two factors on quality and understandability of the different diagrams of UML notation. In addition, in these studies with a concentration on improving the UML diagrams, they proposed stereotype as an effective factor on quality of UML models. Accordingly, in this current study, after extracting the secondary factors, for simplification, the similar factors were grouped into the different 13 main factors, namely, Modelling, Diagram, Evaluation, Software Engineer, Representations, Guideline, Testing, Diagram Type, Requirements, Quality Assessment Approach, Redundancy, Refactoring and System Structure. The subfactors including Conceptual mental model, Aspect-oriented design, Good decomposition model, Model types (OO-DFD and Use Case Methods), Domain modeling techniques, Dynamic models, Impact of aspect-oriented modelling, Modelling language and notation, Process modelling notations, Models slicing, Multi-

view refactoring approach, Level of normality/precision, Conallen's notation, Part-whole relationship and VCL on visual modelling all are grouped as the main factor, named Modelling since all of these subfactors are related to the modelling technique and notation. Since stereotypes of Conallen's notation is related to the specific modelling notation, so called Conallen's notation, so this factor was grouped as the subfactor of the modelling factor, while the same subfactor is grouped as the subfactor of the main factor named Software engineer, since in studies [S48] and [S84], the relation between the stereotype and user comprehension has been the main focus of the scholars. Table 4.1 shows the list of these factors and the article id's addressing the related factors. The present research has identified 103 papers in the time period of 2000-2020 which have studied the effect of 13 factors on the quality and understandability of UML diagrams. These factors are Diagram, Diagram Type, Evaluation, Guideline, Modelling, Quality Assessment Approach, Redundancy, Refactoring, Representations, Requirements, Software Engineer, System Structure and Testing.

Table 4.1 Identified effective factors on the quality and understandability of UML diagrams

Factor	Article Id	n	%
Modelling	[S2], [S3], [S19], [S22], [S30], [S31], [S35], [S38], [S49], [S55], [S57], [S67], [S78], [S81], [S96], [S98], [S100]	17	16.50
Diagram	[S1], [S11], [S43],[S45],[S52],[S54],[S58], [S61], [S72], [S80], [S85], [S88], [S97]	13	12.74
Evaluation	[S4], [S9], [S16], [S25], [S26], [S53], [S76], [S86], [S93], [S102], [S103]	11	10.78
Software Engineer	[S12], [S14], [S15], [S20], [S30], [S33], [S39] , [S41], [S44], [S48], [S84]	11	10.78
Representations	[S6], [S7], [S8], [S29], [S46], [S51], [S64],[S69],[S79],[S87]	10	9.80
Guideline	[S23], [S34], [S37], [S60], [S63], [S82], [S92], [S99], [S101]	9	8.82
Testing	[S21], [S27], [S56], [S65], [S68], [S70], [S71], [S73], [S91]	9	8.82
Diagram Type	[S40],[S62],[S75],[S83],[S95]	5	4.90
Requirements	[S17],[S28],[S32], [S36] ,[S94]	5	4.90
Quality Assessment Approach	[S10], [S74], [S77], [s90]	4	3.92
Redundancy	[S13], [S47], [S66], [S89]	4	3.92
Refactoring	[S5], [S18], [S24]	3	2.94
System Structure	[S50], [S59]	2	1.96

With regard to Table 4.1, it can be construed that among the afore-mentioned factors, the modelling factor with percent of 16.5 ranks the first position in terms of the high frequency of the studies, which is scrutinized in the papers, including [S2], [S3],

[S19], [S22], [S30], [S31], [S35], [S38], [S49], [S55], [S57], [S67], [S78], [S81], [S96], [S98] and [S100].

The factor of diagram is in the second position among 13 factors. This factor is examined in the studies, including [S1], [S11], [S43], [S45], [S52], [S54], [S58], [S61], [S72], [S80], [S85], [S88] and [S97]. Almost 12.74% of 103 papers between 2000-2020 have focused on the diagram as the effective factor on the quality and understandability.

The number of the studies focused on evaluation and software engineer as the effective factors on the quality and understandability is the same with total percent of 21.56%. These two factors rank the third position regarding the study frequency and the studies [S4], [S9], [S16], [S25], [S26], [S53], [S76], [S86], [S93], [S102], [S103] and [S12], [S14], [S15], [S20], [S30], [S33], [S39] , [S41], [S44], [S48], [S84] have focused on evaluation and software engineering, respectively.

The fourth position belongs to the representation factor with frequency percent of 9.8%, which was studied as the effective factor on the quality and understandability of UML diagrams in studies including [S6], [S7], [S8], [S29], [S46], [S51], [S64], [S69], [S79] and [S87].

Both factors including guideline and testing have the same frequency among the studies. That is, almost 8.82% of the studies have focused on the guideline and testing as the factors that have an impact on the quality. Guideline as a factor is addressed by studies including [S23], [S34], [S37], [S60], [S63], [S82], [S92], [S99] and [S101]. Testing as the effective factor on the quality and understandability ranks the fifth position among 13 factors identified in 103 studies. The percentage of the studies, which identify testing as the effective factor, is 8.82. These studies are [S21], [S27], [S56], [S65], [S68], [S70], [S71], [S73] and [S91].

The diagram type as one of the factors influencing the quality and understandability of UML diagrams ranks 6^{th} position among 103 factors. This factor is studied by 4.90% of the studies within time period of 2000-2020 in the studies including S40], [S62], [S75], [S83] and [S95].

Requirements as the effective factor on the quality and understandability, same as diagram type factor, ranks the 6^{th} position among studies during 2000-2020 and its frequency among the research is 4.95%. This factor is examined by the studies including [S17], [S28], [S32], [S36] and [S94].

The factor of the quality assessment approach was studied in four studies including [S101], [S74], [S77] and [S90]. The frequency percent of this factor was equal to 3.92% among the papers between 2000-2020.

Almost four studies have focused on the redundancy as the effective factor on the quality and understandability, including [S13], [S47], [S66] and [S68]. The percentage of this factor is equal to 3.92%.

Among 13 factors identified in 103 studies during 2000-2020 the refactoring factor ranks the 8 positions with 2.94% frequency. This factor is addressed by [S5], [S18] and [S24].

The least studied factor with percent of 1.96% was the system structure, which is studied only by two papers [S50] and [S59].

4.1.1. Modelling

The studies on the modelling as the effective factor on the quality and understandability, focused on 15 factors including effect of conceptual mental model on quality of use case diagram and actors in terms of correctness, completeness, and redundancy of the Use Case Diagram (UCD), aspect-oriented design, effect of good decomposition model (GDM), effect of model types (Object-Oriented Analysis with Data Flow Diagram (OO-DFD) and Use Case Methods) on comprehension during functional requirement modelling, domain modeling techniques, dynamic models, impact of aspect-oriented modelling on readability of UML state machines, effect of modelling language and notation, effect of process modelling notations in terms of intuitive complexity, models slicing, effect of multi-view refactoring approach on models quality, level of normality/precision in workflow modeling, Conallen's notation and developers' experience and ability, part-whole relationship and the effects of VCL on visual modelling quality which are presented in Table 4.2 along with the studies addressed the above-mentioned factors.

Table 4.2 Modelling factor

Studied Factors	Article ID
Conceptual mental model	[S98]
Aspect-oriented design	[S22]
Good decomposition model	[S19]
Model types (OO-DFD and Use Case Methods)	[S100]
Domain modeling techniques	[S3]
Dynamic models	[S30]
Impact of aspect-oriented modelling	[S96]
Modelling language and notation	[S38],[S49],[S57]
Process modelling notations	[S67]
Models slicing	[S55]
Multi-view refactoring approach	[S78]
Level of normality/precision	[S2]
Conallen's notation	[S31]
Part-whole relationship	[S35]
VCL on visual modelling	[S81]

In [S2], the main objective of the authors was to study the level of normality/precision in work modelling based on UML activity diagrams, which may have an impact on two aspects of construct comprehensibility: correctness of understanding and task completion time. Workflow modelling can be realized in two styles, namely, precise style (with specific rules and enforced constraints) and ultra-light style (no rule and no constraint). Based on the ultra-light style the activity diagrams are produced without following rules nor constraints, while for the precise style all possible rules and constraints are exerted. The precise style is useful in comprehending workflows, and it can influence two aspects of artifacts, comprehensibility, and quality. The findings show that modeler and developers gain a better comprehension when precise style is being used, while the precision style does not have significant effect on the task completion time. In [S3], the scholars studied the prevalent domain modelling techniques. Domain models are defined as the models visualizing problem domain objects along with the associations and attributes, encompassing the vocabulary and key concepts of the problem, which are created using category list technique or noun phrasing technique. The results show that category list technique is better than noun phrasing technique for the identification of concepts as it avoids generating unnecessary elements i.e., extra concepts, associations, and attributes in the domain model. The noun phrasing technique produces a comprehensive domain model and requires less effort as compared to category list. There is no statistically significant difference between both techniques in case of correctness. The main purpose of [S19] was to assess the effect of good decomposition model. Decomposition is defined as a

top-down process of breaking down a complex system into smaller and relatively independent units. The results verified that GDM is a useful measure in evaluating the understandability. In the paper [S22], the main consideration of the research was centered on the aspect-oriented design. Aspect-orientation provides a new way of modularization by clearly separating crosscutting concerns from non-crosscutting ones. A conceptual reference model is presented as the article's first contribution, centrally capturing the basic design concepts of aspect-oriented modelling (AOM) and their interrelationships in terms of a UML class diagram. From the results obtained in the evaluation, it is not possible to nominate a winner. The selection of an AOM approach thus has to be made in the context of a specific project. In principle, the approaches allow the modeler to consider both structure and behavior through their approach. Nevertheless, currently no approach addresses the full spectrum of UML in terms of UML's structural and behavioral diagrams as well as their composition. In [30], the authors have focused on dynamic modelling, as well as the software engineers' experience and their ability. Dynamic models are used for modelling the dynamic and functional behavior of the users and software systems. The findings proved positive effect of using theory-guided models on outcome and speed of acquiring the models during the initial stages of training. Bunse and Atkinson [S31] investigated the relation between Conallen's notation and developers' experience and ability. Stereotypes are UML modelling entities for which a specific semantics is defined by the user. Stereotyped UML elements can be represented with a user-defined, intuitive graphical notation (i.e., icons), that replaces the standard stereotype specification, and in this study the scholar have used the Conallen's WAE notation stereotypes which are for diagram elements of web application modelling. According to the obtained results, it was observed that there was no significant difference in employing diagrams stereotyped based on Conallen's notation, but such stereotypes mitigate the gap between the participants in terms of their ability and experience, in such a way that using stereotyped UML diagrams in organizations consisted of low experience developers will exhibit high performance. The study [S35] has investigated the factor of representing in terms of part-whole relationship. They state that there should be difference between part of relations and ordinary associations and there is a little agreement on the precise meaning of the part-of relationship. In [S38], the main topic of the research was to assess the effect of modelling languages, namely UML and OPM. Web applications requires a modelling method which should be capable of

clearly specifying the aspects of applications architecture, communication and distributive nature, and there are two choices, namely UML and Object-process methodology OPM. The results show that OPM excels the UML notation in modelling the web applications in terms of dynamic behavior, while there is no meaningful difference between UML and OPM in terms of specifying the structure and distribution. Moreover, in that study, students created more quality OMP models compared to UML models. In [S49], the authors have juxtaposed two modelling languages, including UML vs. Open Modelling Language (OML). UML stands for unified modelling language while OML stands for open modelling language. The results showed that OML language is more efficient in modelling the dynamic behavior of a system, in terms of comprehension and the generated artifacts are easier to interpret, compared to UML language. The authors of the study [S55] focused on the effect of model slicing in perception of the models. Model slicing is a method for model comprehension, which originates from program slicing. In some cases, modelers may be interested in a particular aspect of meta-models, so they may want to hide the rest of the meta-models. The findings show that model slicing based on interactive navigation features is an effective way in enhancing the model comprehensibility. In [S57], the authors studied two modelling notations used for representing the requirements. There are generally two modelling approaches for requirements, namely Use Case, scenario based and Tropos, Goal-based. According to the obtained results, Tropos exhibited more comprehensibility. In [S67], the effect of process modelling notation in terms of intuitive complexity has been surveyed. Process modelling languages encompass a precise syntactic, semantics and visual process modelling notation. UML Activity Diagram (AD): It is noted as de facto standard for software industry modelling language [18]. Business Process Model and Notation: It was initial published in 2004, with a main purpose to provide a comprehensible notation by the business users. Event Driven Process Chain (EPC): It was developed in 1992, at the Saarbrucken University, the basic idea of EPC is defined in such a way that the events trigger the functions and execute functions triggering events, generate a chain of functions and event ([19] and [20]). The findings show that for the process with lesser complexity, BPMN diagrams are more suitable. For process with high complexity, EPC performance was good, compared to UML AD and BPMN. UML AD can be considered as the most versatile modelling language. In [S78], the authors sought to gain an insight into the effect of multi-view refactoring approach on the

models' quality. Refactoring is one of the basic and essential methods for enhancing the quality of the artifacts, through changing the internal structure without any transformation in the external behavior. For multi-view refactoring, a multi-objective evolutionary algorithm can be suggested to detect a trade-off between enhancement in quality of class and activity diagrams. The statistical evaluation results for four open-source systems approved the efficiency of the proposed method. In [S81], the effect of visual contract language (VCL) on the visual modelling was the main consideration. The visual contract language is devised for enhancing the visual modelling by boosting the diagrammatic cognitive effectiveness and increasing the visual expressivity and level of seriousness and formality. The findings shows that VCL is useful for defect identification. In [S96], the impact of aspect-oriented modelling on the readability aspect of UML state machines was investigated. Aspect-orientation supply an enhanced modularization through separating out the crosscutting concerns in the form of separate entities called aspects. Crosscutting concerns refer to the robustness or security aspect and they can be modelled as aspects models. Using aspect-oriented modelling there would be better performance in defect detection and fixing rates, compared to flat and hierarchal state machines, and from perspective of comprehension, no significant difference was observed. In [S98], the effect of conceptual mental model on the quality of use case diagram and actors in terms of correctness, completeness and redundancy of the use case diagrams was studied. The findings show that Conceptual Mental Model (CMM) has a positive impact on the use case diagrams in defining the system requirements, relations and actors and generation of better use case diagrams regarding capturing, analyzing and defining the functional system requirements in terms correctness, completeness and redundancy. In [S100], the authors tried to establish a relationship between the model types, including OO-DFD and use case diagram and the comprehension in the phase of functional requirements modelling. The system requirements can be modelled using various modelling notations. UML use case and object-oriented DFD are two accepted means for modelling the functional requirements. Both of these methods are based on semi-structured descriptions. The findings show that during functional requirements modelling, use case models are better in terms of quality, than the models which are created using OO-DFD transaction method, since the former is simple, while the latter leads to creation of highly detailed models which are error prone.

4.1.2. Diagram

The studies recognized diagram as the factor with an impact on the quality and understandability of UML diagrams encompass factors, including effect of anti-pattern on quality enhancement, effect of refactoring the anti-patterns, effect of computer-drawn and hand-drawn Areas of interest (AOIs), class cohesion, the use of color and icons, effect of composite states, effect of object diagrams on comprehensibility, effect of architecture-level dependability evaluation techniques, class attribute inconsistencies, Level of Detail (LoD) in UML models as a form of style and rigor in UML modeling, effect of transition technique on quality of class diagram, Object Constraint Language (OCL) post-conditions for operations in UML and OCL models, impact of UML design changes, which are presented in the Table 4.3.

Table 4.3 Diagram factor

Studied Factors	Article ID
Anti-pattern	[S97]
refactoring the anti-patterns	[S80]
Areas of interest (AOIs)	[S45]
class cohesion	[S58]
color and icons	[S54]
composite states	[S85]
object diagrams	[S43]
architecture-level dependability	[S1]
attribute inconsistencies	[S52]
Level of Detail	[S61]
Transition technique	[S88]
UML Design Changes	[S72]
OCL post-conditions	[S11]

In [S1], the authors studied the effect of architecture level of dependability evaluation techniques. Dependability is of great importance for critical systems, which can be also evaluated at model level to ensure reliability and availability of software systems. One of the proposed methods for architecture level dependability evaluation of UML-DAM models is based on hierarchical and colored extension of stochastic activity networks. The obtained results demonstrated that improving the dependability of software/hardware components has a positive influence on the dependability of the entire software system. In [S11], the authors proposed a new method for developing comprehensive OCL post-conditions for operations in UML and OCL models. The behavior description in UML and OCL models can be done through OCL pre- and post-conditions. Indeed, the functionality of an operation is described in a declarative way via use of pre- and post-conditions. The proposed method presents post-/frame

condition determining language (PCDL) elements which are based on new formal distinction and the findings showed that the proposed method simplify the specification and representation of the post-conditions. In [S43], the main purpose was to investigate the effect of object diagram on the comprehension. They believe that Object diagrams present a complete or partial view for the software system structure during run-time. Class diagram along with objects diagrams can be used to record the concepts of problem domain of a software and to specify the relationship among these concepts ([21], [22]). The results suggest that the use of OD does not always make a significant contribution to the design comprehensibility. The benefits are strongly depended on the experience and familiarity with UML. In [S45], the effect of computer-drawn and hand-drawn areas of interest is considered. Areas of interests are defined as groups of components of system architecture diagrams with common property, whose visualization can be done automatically and drawn by human. It is not clear whether UML diagrams equipped by AOIs are enhanced in terms of quality. According to the combined user evaluation, it was observed that AOI enhances both quality and understandability of architecture diagrams. In [S52], the class attribute in consistencies was as the main topic under survey. Detection of intra-model design inconsistences is of great importance for quality software, as ensuring consistency and completeness of models is an undeniable aspect of a software design. Detecting and resolving the inconsistencies via adoption of artificial intelligence leads to completeness of the design and development of quality software. In [S54], the authors have tried to show that the use of color and icons play an important role in diagrams perception. Most of the software engineering notations are used in black and white colors, while the visual communication aspect of modelling in terms of cognitive effectiveness in software engineering is of great importance, which can be improved to make the models easier and quicker to be read. The findings show that the use of color and icons improves readability of misuse case diagrams. In [S58], the cohesion aspect of the class diagram was the main consideration. Class cohesion is a significant attribute in assessing the objected-oriented software quality which denotes the relatedness of the class members, and there are three alternatives: method-method, method-attribute, attribute-attribute interactions. Using new cohesion metric called similarity-based class cohesion (SCC), it is possible to address a number of issues arising from existing metrics and it allows early assessment of designs in terms of cohesion. In [S61], the main focus was on the level of detail in UML models, including

the form the style and strictness degree in UML modelling. Level of Detail refers to the amount of information that can be employed to show the modelling element, like representing a message of sequence diagram using an informal labels or label that shows the method, or even a label which is method name plus argument list. The findings refer to the significant impact of level of detail on comprehension of the UML models. In [S72], the impact of UML design changes was examined. It is deemed that the changes in various modeling phase are able to exert modifications on software models, implying model quality deterioration, like incoherence among models, increased complexity. Indeed, any change in one model should be propagated coherently to all related models horizontally and vertically. For resolving the effects of change, an automated approach is available which analyzes the changes effect on design models and considering the impacts on models' quality based on three techniques, graph-based, rule-based, and metric-based. This method showed 88.33% precision and recall of 95%. In [S80], the refactoring of the anti-pattern is studied as the effective factor. The process of refactoring is defined as the means of improving the structure of software artifact free from any change in the behavior. Anti-patterns are defined as the low-quality solutions for re-occurring design problems. For refactoring, model transformation which detect anti-patterns instances in a given use case model can be used. The results demonstrated that model transformation is capable of enhancing the quality of the use case models in a time and cost-effective manner. In [S85], the composite states were introduced as the effective factors. In fact, the composite states provide the modelers with ability to construct UML state charts in a hierarchical manner. According to the findings, the usefulness of composite states for better understandability of UML state chart diagrams was observed. In [S88], the impact of a transition technique on the quality of the class diagram is surveyed. It should be noted that there is no established method for transition from use cases to class diagrams, and there are generally two methods derivation method (class diagram from use cases) and validation method (class diagram created independently and use case is used for validation). The findings demonstrated using the use cases for validating the class diagrams which are based on textual requirements leads to more proper class diagrams. Therefore, the technique used for transition from use cases to class diagrams may influence the quality of class diagrams, and the effect of transition technique is depended on the category of the developers. In [S97], the anti-pattern as a factor for enhancing the quality is investigated. Anti-patterns show the debatable

diagrammatic and textual structures. Through their detection, it is possible to review the debatable structures or to verify their correctness. The results indicate that applying the technique improves the overall quality and clarity of use case models.

4.1.3 Evaluation

The studies which have recognized evaluation as the factor with an impact on the quality and understandability, encompass factors, including a collection of UML class diagram measures, effect of metrics visualization, measuring design testability, metric, model evolution approach, effect of ontological-based verification method on UML Class/OCL model, feedback-inclusive prototype, impact of size on understandability of UML diagrams, size and complexity of the software, enhancing UML models and effect of meta-associations on meta-models which are presented in the Table 4.4 along with studies addressing them.

Table 4.4 Evaluation factor

Studied Factors	Article ID
class diagram measures	[S86]
Metrics visualization	[S26]
design testability	[S102]
metric	[S25]
model evolution approach	[S76]
ontological-based verification	[S9]
feedback-inclusive prototype	[S53]
size and complexity	[S4], [S93]
enhancing	[S16]
meta-associations	[S103]

In [S4], the authors have studied the size and complexity of the intended software. As the complexity and size of the software systems grows rapidly, the use case diagram drawn to capture the requirements becomes more and more complicated. The understandability and readability of the UCD is also affected. A method based on Page Rank and HITS algorithms are used to identify the important use cases. This approach provides information about the key requirements i.e., which part of the initially unknown problem domain should be looked at first by identifying the key use cases in UCD. In [S9], the effect of an ontological-based verification applied on UML class/OCL model was studied. They point out that the promising solution for hindering the models' errors shift into the code is model verification. In The ontology-based verification method, a class diagram is transformed into ontology declared in web

ontology language (WOL). It is worth noting that ontology has many elements, such as class, relations, and individuals like UML class. This method can efficiently cover all aspects of UML class model verification. Boland, Cherry and Dickson [S16], has devised a solution for enhancing the UML models. The correctness of the models considering both internal consistency rules and system requirements and domain constraints is of a great importance. The findings support the usefulness of domain model in realization of more correct and complete models. In [S25], the quality of UML models is studied from metric point of view. The metrics evaluate the object-oriented concepts: methods, classes, coupling, and inheritance. The metrics focus on internal object structure that reflects the complexity of that entity and also on external complexity that measures the interaction among entities. This paper proposes a quality model to assess the design phase of any object-oriented system based on the works of Chidamber, Kemrer and Basili suggests two new metrics. The design phase model proposed in that study has generated quality indication to real-time project practiced in the industries. The result supports that the proposed design phase model improves the quality. In [S53], the evaluation method based on feedback-inclusive prototype is proposed. Feedback-inclusive rapid prototyping method (FIRP) is a simulation tool used for evaluation of a model with respect to the design software systems as defined in the requirements. Findings show that using a prototyping technique has a significant impact on comprehension of system dynamics presented in the form of multiple interacting state charts, which excels the manual inspection of interacting state charts. In [S26], the effect of metric visualization is the study topic. In this study, metrics are considered as tools to appraise the structure, complexity and quality. Metrics visualization can boost the developers' intuitive perception of the metrics. A new approach for 3D visualization of the metrics used in the design phase of object-oriented software is presented, which extracts the eight key metrics using class diagrams. In [S76], a novel model evolution approach is proposed. They declare that present model evolution approaches do not take into account the reusability of the models sufficiently. A new approach can be presented with three stages, and each stage is defined by four important quality dimensions, namely syntactic, semantic, pragmatic, and emotional. The afore-mentioned approach is precise, and sufficiently automated for software implementation, which facilities the way for evolving and discussing models to be accepted. In [S86], the authors provided a collection of measures for evaluating the UML class diagram. It is clear that in order to make an early evaluation

of the understandability and maintainability, it is important to establish a set of indicators, which need to be based on the features of the early artifacts [23]. Eight measures for assessing the structural complexity of UML class designs can be considered. According to the findings, the predictions models which may be useful for class diagrams understandability and modifiability can be set up based on early measures, which encompass structural complexity. In [S93], the UML diagrams are studied in terms of the impact of size in perception of diagrams. It should be noted that there is no consensus over the exact concept of diagram size, which necessitates the need for operationalizing this notion and analyze its impact on model's understandability. It is clear that with increased diagram size, the visual and mental ability of a modeler is stretched, and the poor-quality layout will be more noticeable. The findings show that there exists a negative correlation between diagram size and modeler performance. In [S102], the main purpose was to measure the design testability. In fact, it is noted that the UML class diagram can be studied in terms of detecting and resolving testability weakness. Testability is defined as the easiness to test a piece of software, which is a strongly desired feature of software. It is able to make the validation phase more efficient by exposing faults during testing, and consequently to boost the quality of the end-product for users' satisfaction. It is deemed that the class interaction is an important attribute in terms of design testability. In fact, class interaction emerges when potentially there are concurrent client/supplier associated between classes of the systems and such interactions need to be enhanced. It can be deduced that the class interactions can be combined in different ways; in some cases, not all the class interactions have to be considered, in other cases, it is necessary to deal with all the class interactions. In [S103], the scholars have focused on the effect of meta-association on meta-models.

4.1.4. Software engineer

The studies recognized software engineer as the factor with an impact on the quality and understandability, encompass factors including experience, high ability, learning difficulties, participants' familiarity, reading technique, stereotype-enhanced UML diagrams effect on understandability, stereotypes, Chunking, Ordering and Patterning (CHOP) method effect on improving novice analyst performance (learning difficulties), difficulties in transition from object-oriented analysis (OOA) to Object-

oriented logical design (OOLD) and cognitive support in software design tools which are presented in the Table 4.5 along with studies addressing them.

Table 4.5 Software engineer factor

Studied Factors	Article ID
Experience	[S30]
High ability	[S30]
Learning difficulties	[S12][S14] [S15][S33]
participants' familiarity	[S44]
Reading Technique, Reading Technique	[S20][S41]
stereotype-enhanced UML diagrams, stereotypes	[S48] [S84]
Cognitive support of Design tools	[S39]

In [S12], the learning difficulties associated with UML diagrammatic representation are discussed. In that research, it is deemed that some technical properties of UML diagrammatic representation, coupled with students' cognitive attributes, hinder both perceptual and conceptual processes involved in searching, recognizing, and inferring visual information, which creates learning barriers. That Study also demonstrates the usefulness of participatory cognitive approach in developing the shared understanding between instructors and students on difficult course materials. The concept-mapping technique enables one to assess and represent learning challenges meaningfully and efficiently. Other information system courses can use it to enhance teaching and facilitate learning. In [S14], for enhancing the novice analyst performance, CHOP method was analyzed. CHOP technique is defined as chunking, ordering and patterning, which improve novice analyst's performance in modelling sequence diagram, and it is based on cognitive load theory (CLT). The obtained results demonstrated that the CHOP technique significantly enhances novice analyst's ability to model interactions among objects. In [S15], the learning difficulties associated with the complexity of the sequence diagram is examined. The learning difficulties associated with sequence diagrams are the complexity of learning sequence diagram, due to the unfamiliar metaphors, working with sequence diagram (SD) (SDs have a detailed and very specific syntax), omitting explicit returns (The notion of flow of control through the logic is not evident in an experienced analyst's SDs, since explicit return notations are often omitted to reduce clutter and fit more logic into a given space), Decomposing high-Level Requirements into Logic Steps, Choosing to Centralize or Decentralize Responsibility Assignments, and Rate of Information Presentation. The authors of [S15] state that the suggested recommendations will be

interesting to educators, experienced analysts mentoring novices, and others. In [S20], the authors have focused on the items including software engineers' ability and experience, as well as dynamic models as the sub-factors. For evaluating the quality of the models, defect detection is a proper method which is applied using reading techniques, while there are some limitations and constraints for the reading techniques, and it signifies the importance of the need for new reading technique. The results indicate that the Functional Size Measurement (FSM) is useful since it finds all the defects related to a specific defect type, it finds different defect types than an inspection group, and it finds defects related to the correctness and the consistency of the models. In [S30], the authors have focused on the items of software engineering, such as high ability, experience and dynamic models as the effective factors. Dynamic models are used for modelling the dynamic and functional behavior of the users and software systems. The findings proved the positive effect of using theory-guided models on outcome and speed of acquiring the models during the initial stages of training. In [S33], the difficulties related to the transition from OOA to logical design were the main considerations. The transition from OOA to OOLD imposes a series of difficulties on the learners and software engineers in terms of semantic distance. The findings ascertain that due to two factors the learners and engineers face with difficulties during transition from object-oriented analysis to logical design, namely, semantic distance and the precise degree of transition relations. In [S39], the cognitive suggesting utility in the software design tools are investigated. Most of the software design tools provide the basic `functionalities for the engineers, such as editing, viewing, storing and transforming designs, but lack support for cognitive tasks like decision marking, decision ordering and task-specific design understanding. Some extent of cognitive needs of the software engineer and designers are implemented by Argo/UML. In [S41], two reading techniques namely perspective-based reading (PBR) and checklist-based reading (CBR) along with their effect on quality are surveyed. The act of inspecting a software is consisted of several activities namely, planning, defect detection, defect collection and defect correction [24]. In [S24], two known methods including PBR and CBR are introduced as the effective factors on the quality of UML design document. The former provides the ground for controlling a document from a specific perspective of the stakeholders, while the latter provides no clear advice for inspectors in terms how to accomplish the process and to focus specify on which aspect of the document during the reading activity ([25], [26] and [24]). The results showed

that PBR is more efficient than CBR in error detection and error detecting based on PBR is more cost-effective for UML design document. PBR is also more reliable in enhancing the UML models quality. In [S44], the authors have recognized the participants' familiarity with modelling as a factor with possible effect on understandability. The familiarity refers to the developers experience in modeling and their knowledge of the application domain. The results demonstrated that the developers' familiarity has a meaningful effect on task completion time, while there was no significant effect on models' quality. In [S48], the stereotype is selected as the study factor. They believe that stereotypes are used for customizing the modelling language for its application in specific domains and their main role is to brand an existing models' components with specific semantics, as well as supplying a secondary classification of modelling components. The obtained results refer to the meaningful role of stereotypes in boosting the comprehension of models. In [S84], UML diagrams are equipped with stereotypes, and they were analyzed in terms of understandability. Dong et al. [27] have proposed a UML diagrams equipped with stereotypes, the tagged values and constraints for visualizing the pattern-related information in UML-class diagrams. The tagged values contain information about the role, class, attribute, or a method has in a design pattern. According to the data, it is observed that UML diagrams which are enhanced by stereotype are more useful in identifying the composition and role, than UML collaboration diagram.

4.1.5. Representation

The studies which have recognized representation as the factor with an impact on the quality and understandability, encompass factors including use of iconic signs as UML graphical notations, effect of layout technique, naming for conceptual elements, representation of roles in conceptual models, effect of SORT, effect of use case description quality check list on model quality, graphical and textual notation to represent non-functional requirements, extending the UML state charts notation to model security aspects, customization of the process models representation to resolve lack of fit, the effect of visual syntax of use case and representing physical quantities in software models which are presented in the Table 4.6 along with studies addressing them.

Table 4.6 Representation factor

Studied Factors	Article ID
Iconic signs	[S7]
Layout technique	[S69]
Naming for conceptual elements	[S64]
Representation of roles	[S6]
Effect of SORT	[S87]
Use case description quality	[S79]
Modeling the security aspects	[S28]
Customization of the process models representation	[S29]
Visual syntax of use case	[S46]
Representing physical quantities	[S51]

In [S6], the scholars have focused on the effect of representation of the roles in the conceptual models. They point out that when describing an organization or part of it, a key concept is the notion of roles played by actors in the domain. Actors in an organization act in various roles, hence, showing that roles in a conceptual model can promote understanding of how the organization works. The role can be declared as a conceptual modelling construct and a set of guidelines for representing roles in conceptual models can be defined to suggest rules for modelling roles with the widely used extended entity-relationship grammar. The results indicate that the role-based rules are not only useful for understanding the models, but also provide direct clues as to why this is so. In [S7], they studied the use of iconic signs and its effect on the UML graphical notation. It is only through the process of ‘‘sense making’’ that the relationship between signs and its representation can be established. There are three modes of signs: iconic, symbolic, and indexical. Iconic sign is defined as a mode in which the signifier is perceived as resembling or imitating the signified (recognizable look, sound, feel, taste, or smell) or being similar possessing some of its qualities: e.g., a portrait, a cartoon, etc. In [S7], it was hypothesized that the use of iconic signs as UML graphical notations leads to representation that is more accurately interpreted and that evokes fewer connotations than the use of symbolic signs. The results support our propositions that iconic UML graphical notations are more accurately interpreted by engineers, modelers, and analysts. Moreover, the number of connotations is lower for iconic UML graphical notations than for symbolic UML graphical notations. In [S8], the main purpose of the study was to propose a new graphical and textual notation to represent non-functional requirements. They clarify that in the available notation the representation of functional requirements is feasible while no interactive requirements representation is considered. The results of the evaluation indicate that

the representation of operation cases helps in documenting a complete set of functional requirements, which ultimately results in a comprehensive translation of requirements into design. The findings show that the representation of non-functional requirements makes a significant contribution in terms of translation of the requirements into the design. In [S29], the authors have tried to extend the UML state charts notation to model the security aspects of the system under design and development. Process models are employed during analysis and design of the system. However, there is a need for representations customization in order to resolve the lack of fit between these models and in intended task. The findings ascertain that due to the impediments some customization in the process model's representation is required and such customization can be done from two perspectives of physical and process customizations. In [S46], the influence of visual syntax on UML use case diagrams was assessed. Use case modeling is known as one of preliminary artifacts in capturing system functional requirements. In fact, the model reader gains insight into the mental idea of any model via visual perception, and if it is misread or misinterpreted then the main objective of the modelling has been unsuccessful. According to the findings, the cognitive effectiveness of use case representation can be enhanced by making some enhancements. In [S51], the authors have paid attention to enhancing the UML notation in order to represent the physical quantities. In developing a software dealing with physical entities, like automotive or aerospace domains, it is highly required to consider the formal representations of the uncertainty and units both in design and implementation phases. The findings show that the physical quantities representations can be incorporated in the software models, which enable the modelers to safely represent and manipulate units and measurements uncertainties of physical systems. In [S64], the effect of naming for the conceptual elements was the main concern of the study. Naming the conceptual schema elements is a challenging problem as for any new element, a new name is required, and the impact of name on comprehension of the schema is not ignorable. For naming the element in conceptual models a guideline can be sued. The name which is opted in accordance with the guideline, if it possesses the form and the sentence generated from the sentence pattern, is grammatically well-formed and semantically meaningful. In [S69], the impact of layout technique is studied. Using the layout technique Vertically Aligned Class diagrams (VACDs) by incorporating the usability of diagramming technique into the class diagrams produce analysis models which are more understandable for domain experts. The obtained

results showed that the use of VACDs leads to an increase in efficiency, improvement in communication and reduction in the errors. In [S79], the authors presented a checklist to control the quality of use case description and studied the effect of this checklist on the use case artifacts. The use case model is a tool for depicting the system specification in a natural and understandable way with a focus on the system from user perspective [28]. The use case description checklist can be useful in checking use case guidelines, for testing desired quality features in the description and the quality features themselves are employed for examining and revising the use case descriptions. Either through a direct application or using the guidelines as the use case guidelines, the quality of the use case description will be a good measure to assess and improve the quality of the use cases. In [S87], the effect of SORT on diagrams quality in terms of understandability and verifiability was studied. SORT is a systematic approach for refining and translating UML diagrams ([29], [30] and [31]) which is used for resolving the issues related to the supporting the mapping of UML artifact into executable code. The observations ascertain that SORT leads to generation of more quality models in terms of understandability and verifiable.

4.1.6. Guideline

The studies which have recognized guideline as the factor with an impact on the quality and understandability, encompass factors including explicit reuse guidance and/or variability specification on domain models comprehensibility, rules and guidelines in terms of abstraction rules, effect of theory-guided on learning, visual guidelines on aesthetic quality of the diagrams, knowledge-based object modeling advisor, effect of ontological modelling guideline, effect of using examples on structural model comprehension and effect of work system snapshots which are presented in the Table 4.7 along with studies addressing them.

Table 4.7 Guideline factor

Studied Factors	Article ID
Reuse guidance and/or variability specification	[S82]
Abstraction rules	[S23]
Effect of theory-guided	[S37]
Visual guidelines	[S60]
Object modeling advisor	[S63]
Ontological modelling guideline	[S99]
Using example	[S34], [S92]
Work system snapshots	[S101]

In [S23], the author attempted to propose an abstraction rule. They state that the process of abstraction is the simplification of models by removing details not necessary at a higher, more abstract level. The abstraction technique in [S23] uses abstraction rules that have input and result patterns. Abstraction rules define the semantics of how a set of model elements can be replaced by a less complex, more-abstract model element. It was shown the proposed technique scales, produces correct results, and the addresses issues such as model ambiguities that are inherently part of many (UML) diagrams. They demonstrated various forms of ambiguities and showed that there are ways of living with them—even preserving them during transformation. In [S34], the authors tried to show the usefulness of the example in teaching software designs. Then examples are presented via a searchable repository. These examples are offered via a searchable repository. The results showed that the designs fabricated based on examples were 18% better that the designs which were not produced based on example. In [S37], the paper investigated the effect of theory-guided on improving the learning conceptual models. The effect of theory-guided on learning: For enhancing the learning of the conceptual models several theories have been proposed with main focus on guiding for structuring the conceptual models, such as theory of ontological expressiveness [32], unified foundational ontology [33] and physics of notations [34]. Experiment shows positive effects on both outcome and speed of learning in the initial stages of learning, the real contribution of the study [S37] is methodological, i.e., an eye-tracking procedure to observe the process of learning from conceptual models. In [S60], a set of visual guidelines were provided. The authors state that due to the increase in success of visual specification notation, like UML, visual guidelines are required to standardize the presentation of modelling diagrams in terms of human communication, perception, and readability. A new taxonomy of guidelines was introduced based on diagraming rules from literature and semantic layout rules for UML class diagrams. Findings showed that a subset of rules did not have a significant impact on the model's understandability (may be due to the low number of the study participants). In [S63], the scholars focused on usefulness of knowledge-based object modeling advisor. Due to the lack of mutual understanding of the domain-specific knowledge between system analyst and requirement engineer, as a result miscommunication between the users and technical personnel should be noted as the two common problems of early phases of system development. The obtained results

prove the effectiveness of knowledge-based object modelling advisor system, which is added as an extension to an open-source Computer-Aided Software Engineering (CASE) tool, particularly in terms of advice and suggestions for model completeness aspect in terms of semantic quality. In [S82], the main consideration was the effect of the explicit reuse guidance and variability specification on domain models comprehensibility. The findings showed that the explicitly specification of reuse guidance for domain model contributes to the understandability of model, while for the variability enhanced the comprehensibility only to a limited degree. In [S92], the effect of using examples on the comprehensibility of the structural models was studied. An example-driven modelling is considered as a method which adopts examples for model comprehension and domain knowledge transfer in a systematic manner. The findings show that the developers provided with abstractions which are augmented with samples exhibited higher perception of the diagrams and in their tasks, and they asked %90 fewer clarification domain specific questions. In [S99], the effect of the ontological modelling guideline was studied. The main assumption of Bunge's ontology states that "the world is made up of substantial things which are assumed to really exit". He states that a thing has a properties which can be intrinsic or mutual [35]. The ontological guidelines are capable of enhancing the usefulness of UML class diagrams for describing the application domain. In [S101], the authors sought to show the impact of using work system snapshots on the quality of the use cases diagrams. A work system is defined as a system where human and machines accomplish work using information, technology, and other resources to produce products or offer services. A work system snapshot is a one-page brief description of a work system identifying the main components of six central elements: customers, product/services, processes and activities, participants, information, and technologies. According to the findings, it became evident that a work system snapshot leads to production of quality use case diagrams.

4.1.7. Testing

The studies which have recognized testing as the factor with an impact on the quality and understandability, encompass factors including testing and validation of state chart models, testing of the domain models in early stage, testing of a model in terms of semantic correctness, testing of models considering the role of the models verification, testing the models using light weight and statics verification, testing a model in terms

of considering the security requirements, UML-based Test Generation, effect of testing, testing of conceptual schema, and testing the models in terms of eight use cases which are presented in the Table 4.8 along with studies addressing them.

Table 4.8 Testing factor

Studied Factors	Article ID
Testing and validation	[S91]
Testing in early stage	[S27]
Testing in terms of semantic correctness	[S21]
Role of models verification	[S73]
Lightweight and static verification	[S71]
Testing of Consistency	[S68]
Test Generation	[S56]
Testing of conceptual schema	[S65]
Testing the models in terms of eight use cases	[S70]

In [S21], the testing of UML conceptual schemas in terms of semantic correctness was studied. The semantic correctness of a conceptual schema can be seen from two different perspectives. On the one hand, from the point of view of its definition, a conceptual schema must be right (verification techniques) that check whether the schema satisfies several correctness properties. On the other hand, from the point of view of the requirements that the information system should satisfy, a schema must also be the right one (validation techniques) which help the designer understand the exact meaning of a schema and to see whether it corresponds to the requirements. In that article an approach to verify and validate UML conceptual schemas, with arbitrary constraints formalized in OCL was proposed. This approach allows the verification and validation of very expressive schemas, ensuring that when a property is satisfied by a schema, an example proving so will always be obtained, as long as the schema satisfies the conditions guaranteeing termination. In [S27], the authors focused on the testing of the domain models in an early stage, since they believe that it is essential to be assured about quality of the early models in model-driven engineering. The quality of the models can be assessed in terms of syntax and semantic, which are so-called verification and validation, respectively. The scholars have proposed a methodology and a tool the deduce a set of customizable questions in natural languages from each model and answering these questions will aid the experts in the defective sections of the models and ensuring the quality of the domain models in early stage. The obtained results proved the effectiveness of the proposed tool, MOTHIA in detecting the modeling errors. In addition, the proposed methodology narrows the gap between

domain experts and modeling experts, which is the main source of early-stage modelling errors. In [S56], the focus was centered on UML-based test generation. The objective of the study was to examine which errors are possible and made when manually deriving test cases from UML activity diagrams or state machines and whether there are differences between these diagram types. The authors aimed to identify the error types that are formed during manually test derivation using UML activity diagrams or state machines. They found that activity diagrams are more error prone, while have a higher perceived comprehensibility. In [S65], the authors proposed a model for validating of conceptual schemas. Testing is defined as an attempt to try to make the program to fail, to uncover the faults by triggering failures, as most of the preceding studies assume that the conceptual schemas are executive so they can also be tested. The proposed method for testing the conceptual schema is based on software testing. It was shown that testing the conceptual model should be regarded as the tools for their validation, and the schemas should be tested with the goal of uncovering the faults by triggering failures. The authors of the study [S68] examined the testing of UMLsec model considering the security requirements during the model evolution. Consistency of security requirements: Using UMLsec models, it is possible to be ensured that the security vulnerabilities are identified at an early stage of the software development, which can be verified in terms of consistency using UMLsec stereotypes. Using UMLsec stereotypes leads to security check of the model evolution, and it has a considerable efficiency that simplifies the re-verification of the entire model. In [S70], the scholars sought to test the models in terms of 8 use cases. Validation and verification are defined as "are we building the right product? and "are we building the product right", respectively [36]. Eight validation and verification use cases play role on quality of the class diagram, namely, model consistency, property satisfiability, constraint implication, constraint independence, solution interval exploration, partial solution completion, equivalence implication and partitioning with classifying terms. The results showed the usefulness of the 8 use cases of validation and verification.

In [S71], testing the models using lightweight and static verification was studied. The lightweight and static verification method can be used to evaluate the correctness of the executable models, since the correctness of these models is highly important due to their ability in facilitating the semi-automatic implementation of software system. The lightweight method performs static analysis of the model, and it examines the

model without executing its operations. This method supplies a repairing feedback to aid the designers for improving the model. In [S73], testing of UML sequence diagram was studied considering the role of the models' verification. Checking the correctness of the software models has a crucial role on ensuring the quality of the final product. For model verification, a systematic method of transformation and verification UML sequence diagram into Z specification can be used to facilitate practitioners for verifying the models based on UML sequence diagram. In [S91], the authors investigated the effect of testing and validation of the state chart models. The complexity of state chart models signifies the importance of advanced model testing and validation to increase in the quality and reliability of behavioral software models, since the designing phase of the statecharts can be error prone. The proposed method for testing and validation the statecharts exhibited good performance in upgrading the traditional state chart designs.

4.1.8. Diagram type

The studies which have recognized diagram type as the factor with an impact on the quality and understandability, encompass factors including order of presentation of documents, paired combination of the dynamic diagrams, Diagram type in data modelling, UML class diagram in terms of supplying both comprehension and modification, effect of sequence and collaboration diagram and application domain and the effect of diagram type (activity and state machines) on comprehensibility in test case derivation which are presented in the Table 4.9 along with studies addressing them.

Table 4.9 Diagram type factor

Studied Factors	Article ID
Order of presentation of the documents Paired combination of dynamic diagrams	[S40]
Data type in modelling	[S83]
Class diagram for providing both comprehension and modification	[S95]
Sequence and collaboration diagram	[S62]
Activity and state machines in comprehension	[S75]

In [S40], the authors have attempted to study the semantic comprehension considering the effect of diagram type, application domain, order of the documents presentation and paired combination of the dynamic diagram. They stated that diagram type in their research includes only sequence, collaboration and state diagrams of UML, and

application domain refers to the business domain for which a software is being developed. The presentation order refers to the delivery order of documents to the study subjects. It was observed that the state diagrams are capable of presenting higher semantic understanding for dynamic aspect the system when the domain is real-time system, while sequence diagrams exhibited better result for management information system. Moreover, it was shown that a greater level of semantic comprehension can be realized using the pair sequence-state for modelling the dynamic aspect of the system.

In [S62], the effect of diagram types including sequence and collaboration diagrams and the application domain are considered. Interaction diagram of UML is presented in two forms, namely sequence diagram, depicting an explicit sequence of stimuli messages communicated between objects instance, while collaboration diagrams is a direct graph, and the nodes denote the communicating entities and edges are used for communications. The results showed that collaboration diagrams can be easily perceived for RT system, compared to sequence diagram, while in management information system, there is no significant difference in comprehension of these two diagram types. Also, the comprehension of interaction diagram for Management Information System (MIS) is easier than RT system. Moreover, for MIS system more quality collaboration diagram can be created, compared to sequence e diagram. In [S75], the effect of diagram type (UML activity and state machines) on the understandability in the test case derivation is investigated. For modelling the behavior of the system from perspective of the user there are two alternatives in UML modelling language, namely activity diagrams and state machines. Testing the system based on the available design artifacts is a variant of system testing which is bonded on explicit models. The findings showed that activity diagrams excel state machines in term of understandability, while these diagrams are more error-prone in derivation of the manual test cases. In [S83], the effect of diagram type on the comprehensibility by data modelling was studied. For data modelling there are two well-known graphical notations, namely Entity-relationship (ER) and UML class diagrams. The results showed that UML class diagrams possess a better comprehension. In addition, UML class diagrams provide a good support for verification activities, compared to ER diagrams. In [S95], the authors focused on UML class diagram with a hypothesis that UML class diagram provides a better support in terms of comprehension and modification. Entity-relationship along with its extensions are considered as the most

common notion used in database conceptual modelling, while object-oriented approach has also encouraged the modelers to use UML class diagrams for representing the data. The findings show that UML class diagram exhibit a proper support both in terms of comprehension and modification during data modelling, compared to ER diagram.

4.1.9. Requirements

The studies which have recognized requirements as the factor with an impact on the quality and understandability, encompass factors including complex interactions between the class diagram constraints, propositional logic constraint patterns, representation of business roles from early stage, extending the UML statecharts notation to model security aspects and requirements reusability which are presented in the Table 4.10 along with studies addressing them.

Table 4.10 Requirements factor

Studied Factors	Article ID
Class diagram constraints	[S94]
Constraint patterns	[S32]
Business roles	[S17]
Requirement's reusability	[S36]
Model security aspects	[S28]

In [S17], the representation of the business roles from early stage of the software development life cycle was studied. They believe that any change in business policy may deeply influence the information system evolvability and efficiency. So, business rules should be represented in the models from the early stages. By business rules modelling, there will be an extension in the state-of-the-art approaches for modelling the business rule and there will be a reduction in the redundancy rate and the modeling of the business rules hinders the conflict between the business rule types. In [S28], the authors have tried to extend the UML state charts notation to model the security aspects. Modelling the security aspect of the system has gained interest among the scholars, while the UML statecharts notation can be also extended to model the security aspect. The obtained results demonstrated that the new notation for UML statecharts is more understandable in terms of reading the models. In [S32], propositional logic constrain patterns were proposed for enriching the UML-based conceptual modelling. Constraints are defined as limitations on the states of the data,

as well as the objects owning them, which can be represented by a set of authentic values or through relationship configurations [36]. It should be noted that the constraint pattern refers to the repeating structures specifying certain categories of k-variable propositions observed on domain of discourse during conceptual modelling and these constraints denote propositional statements easily added to ER or UML class diagram. A set of invariant patterns are suggested for capturing a rich category of propositional constraints, and the results demonstrated that the approach enriches UML-based conceptual modelling for greater completeness, consistence, and correctness via formalized syntax and semantics of these constraint patterns. In [S36], the authors tried to propose an approach for enhancing the requirements reusability. It has been always challenged to develop models, which are able to store the requirements promoting the easy requirements reuse, while at the same time be understandable by software engineers. An ontology-based approach provides the feasibility of reasoning and to detect the missing or incomplete requirements, which reduces the efforts needed for requirements elicitation at the early stage of software development life cycle. In [S94], the complex interactions among the class diagram constraints were studied. The main reason of the correctness defects in the class diagrams is due to the complex interaction between the class-diagrams constraints. Constraints are defined as the restrictions acted on the components of class diagram. The findings show that increasing the awareness leads to a surge in the rate of the identification.

4.1.10. Quality assessment approach

The studies which have recognized quality assessment approach a the factor with an impact on the quality and understandability, encompass factors including the effect of quality assessment approach (QAA) on the syntax, semantic, pragmatic quality, evolvability and capability, quality assessment approaches in terms of timing inconsistencies, Systematic approach for assessing the security models and the application of the Hyperlink-Induced Topic Search (HITS) algorithm in object-oriented designs in order to evaluate the quality which are presented in the Table 4.11 along with studies addressing them.

Table 4.11 Quality assessment approach factor

Studied Factors	Article ID
Effect of QAA on the syntax, semantic, pragmatic quality, evolvability and capability	[S74]
Rule-based quality assessment approaches in terms of timing inconsistencies	[S77]
HITS algorithm quality assessment approach	[S90]
Systematic approach for assessing the security models	[S10]

In [S10], the authors proposed a systematic approach for assessing the security models, so-called UMLsec. In nowadays industry of software, security is only taken into account at implementations stage, which hinders the effective communication of the security solutions to the stakeholders. System security is an integral aspect of any system, which should be also checked at early stage of software development. However, modelling language notation and validation support is required. As a systematic method for assessing the quality of security models, UMLsec is proposed and the findings demonstrated the efficiency of the proposed method in developing high quality security models in terms of correctness, higher semantic completeness, easier modifiability, understandability and easier to communicate between stakeholders.

In [S74], the effect of quality assessment method approach on the syntax, semantic, pragmatic quality, evolvability and capability was studied. As the meta-model-based modelling languages change continuously, there is a need for quality assurance approach to assess the quality of meta-models, as the quality assessment approaches have an impact on quality of meta models. An approach based on the quality models can be used to measure the quality of meta-models, which was able to measure the syntax quality, semantic quality, pragmatic quality, evolvability and capability. In [S77], a rule-based systematic approach was introduced for controlling the timing inconsistencies. Timing inconsistencies and Quality assessment approach in terms of timing inconsistencies may affect the quality of the models for real-time embedded software. UML/Modeling and Analysis of Real-Time and Embedded (MARTE) systems is an approach used for modelling real-time embedded software (RTES). UML/MARTE inconsistencies are defined as the issues related to the modelling faults, which may be hazard during the life cycle of RTES. A systematic approach based on rule can be used for time consistency checking and the obtained results proved the effectiveness of this approach for cruise control system software and guidance and control unit software. In [S90], the HITS algorithm for checking the quality of a model like class diagram was studied. Generally, metrics are tools for assessing and

quantifying several aspects of the software engineering process and product, which are mainly focused on object orientation or measure specific characteristics of design. The HITS algorithm can be employed for assessing the quality of a model, like class diagram. According to this method, findings show that the role of each class in a class diagram is bonded to not only the number of the incoming and outgoing messages, but also on the importance of the class. Using HITS algorithm in evaluation of a class diagram, it is feasible to obtain hub weights for each class which records the combined effect of communicating with other classes. Also, these weights can be used for adopting the adjacency matrix of the direct graph.

4.1.11. Redundancy

The studies which have recognized redundancy as the factor with an impact on the quality and understandability, encompass factors including the absence of the constraint redundancy, construct redundancy and the modelling knowledge of the reader, effect of redundant multiplicity constraint on quality and referential redundancy modelling construct, which are presented in the Table 4.12 along with studies addressing them.

Table 4.12 Redundancy factor

Studied Factors	Article ID
Absence of constraint redundancy	[S47]
Construct redundancy	[S13]
Modelling knowledge	[S13]
Redundant multiplicity constraint	[S89]
Referential redundancy	[S66]

In [S13], the authors focused on the effect of the construct redundancy and modelling knowledge of the reader on the understandability. It should be noted the conceptual models play a key role in comprehending the domain models related to the information system, while in some cases the existence of constructed redundancy induced using conceptual modelling grammar may lead to reduction in the clarity of the conceptual models. The findings show that a model with a weak level of redundancy is more understandable for the readers, while the trained readers are not affected by the weak level of redundancy in a model. In [S47], the main purpose was to check the absence of the constraint redundancy. In the verification process, the models should be checked in terms of correctness, from perspective of not violating the integrity constraint and

avoidance from unnecessary constraints in the models, so-called constraint redundancies. It was observed that considering the diagram structure and constraints it encompasses, it is possible to analyze large class diagrams efficiently using UML to Constraint Satisfaction Problem (CSP). In [S66], the authors have investigated the modelling constraint in the referential redundancy which occurs in a class represented by the several constructs or models elements, thing, or properties of the problem domain. The referential redundancy reduces the model's quality, the referential redundancy can be eliminated at the language level and potentially reduced at the model level. UML language can be reformulated by breaking down a modelling construct into the smallest parts. In [S89], the focus was centered on the effect of the redundancy multiplicity construct on the quality of UML class diagram. The UML class diagrams are capable of representing complex constraints on its components while the interaction among them leads to mitigation in correctness. Redundancy refers to the existence of excessive unnecessary information. An algorithmic rule-based methods are used for resolving the redundancies in the multiplicity constraints

4.1.12. Refactoring

The studies which have recognized refactoring as the factor with an impact on the quality and understandability, encompass factors including a multi-view integrated refactoring approach, refactoring based on formal concept analysis and relation concepts analysis, refactoring for preservation of the class invariants as the refactoring-related facts which are presented in the Table 4.13 along with studies addressing them.

Table 4.13 Refactoring factor

Studied Factors	Article ID
Multi-view integrated refactoring approach	[S5]
Refactoring based on formal concept analysis and relation concepts analysis	[S81]
Refactoring for perseverating the class invariants	[S24]

In [S5], the authors tried to show the usefulness of multi-view approach for refactoring. Software refactoring is the process of changing a software system in a manner that does not alter its external behavior and yet improving its internal structure, which can be also used at modelling level. A multi-view integrated approach to model-driven refactoring using UML models is proposed. The results indicate that more bad smell instances can be detected using the integrated approach rather than the individual

refactoring approach. In [S81], the findings for a refactoring based on formal concept analysis and relational concepts analysis were presented. The visual contract language is devised for enhancing the visual modelling by boosting the diagrammatic cognitive effectiveness, increase the visual expressivity and level of seriousness and formality. The findings show that VCL is useful for defect identification. In [S24], the refactoring in terms of preserving the class invariant and its possible effect on UML class diagram were studied. In the field of software engineering, the term class invariants are known as a valuable term employed to delineate the semantic of UML class diagram elements (attributes and relationships) and must be held throughout the lifetime of instances of the class. Refactoring, the activities of re-distributing classes, attributes, and methods across the class hierarchy, is a powerful technique that is used to improve the quality of software systems. Performing refactoring on UML class diagrams requires a special investigation of invariant-preserving on the refactored models. An approach to preserve class invariants in refactoring UML models was proposed which demonstrates the excellence of the proposed approach when combined with other maintain activities to improve the quality of the software model.

4.1.13. System structure

The papers with a focus on the system structure as the effective factor on the quality and understandability studied the effect of structural complexity and structural complexity of extract, transfer and load (ETL), which are presented in the Table 4.14 along with studies addressing them.

Table 4.14 System structure factor

Studied Factors	Article ID
effect of structural complexity	[S50]
structural complexity of extract, transfer and load	[S59]

In [S50], the structural complexity of a state diagram was investigated. Structural complexity and its effect should be studied in terms of three dimensions, namely the size and control flow complexity, the actions related to states and the sequence of actions. According to the results, it was observed that each of complexity dimensions has a meaningful effect on understandability of a state chart diagram In [S59], the structural complexity of the models for ETL was examined. The conceptual models of Extract, Transform and Load process are software designs, whose structural

complexity may have an influence on their cognitive effectiveness [37] and high cognitive complexity mitigates the understandability and modifiability of the models. According to the correlation results, it can be stated that there is a correlation between the understandability and measures, like NFES which is a complexity measure, and it is defined as the number of input and output flows in the ETL process model.

4.2. Perspectives of reviewed studies on UML diagrams quality and understandability

The second research question of the present study aimed at answering the research question which was RQ2. *From which perspectives the quality and understandability of UML diagrams are studied during 2000 to 2020?*

By applying meta-analysis on 103 journal papers, it was found that there are 19 major themes, namely, UML-based Test Generation, Dependability, Comprehension, Quality, Refactoring, Rules and Guidelines, Effectiveness, Enhancement, Quality Assessment Approaches, Learning challenges and obstacles, Reading technique, Aspect-oriented design, Variability, Error detection and resolution, Metric proposal, Testing the model, Comparison, Maintainability, Refinement and translation and Transition technique, from which the UML diagrams are studied. In Table 4.15 the major perspectives from which the papers during the period of 2000-2020 have studied the quality and understandability of UML diagrams are presented with paper count in descending order along the right-hand side column, with five years intervals, including 2000-2004, 2005-2009, 2010-2014, 2015-2019 and 2020.

Table 4.15 UML-related studies trend

Topic Title	2000-2004	2005-2009	2010-2014	2015-2019	2020	Total count per each topic
Enhancement	4	7	8	10	1	30
Comprehension	1	6	6	8	0	21
Testing the model	0	2	4	7	0	13
Quality Assessment Approaches	1	1	4	1	0	7
Refactoring	0	1	1	5	0	7
Comparison	0	2	4	0	1	7
Rules and Guidelines	0	1	2	3	0	6
Learning challenges and obstacles	0	1	0	3	0	4
Refinement and translation	0	1	0	2	0	3
Effectiveness	0	0	2	0		2
Reading technique	1	0	1	0	0	2
Quality	0	0	1	1	0	2
UML-based Test Generation	0	0	0	2	0	2
Transition technique	0	1	0	1	0	2
Aspect-oriented design	0	0	2	0		2
Variability	0	0	1	0	0	1
Metric proposal	0	0	1	0	0	1
Maintainability	0	1		0	0	1
Dependability	0	0	0	1	0	1
Total	7	24	37	44	2	

With regard to the above Table 4.15, it is clear that most of the studies have investigated UML diagrams in term of enhancement by proposing method to boost the diagrams quality. The peak year for this perspective was 2015-2019 when 10 studies were carried out with a focus on the enhancement aspect of the diagrams; these studies include [S4], [S11], [S28], [S34], [S36], [S37], [S46], [S51], [S54] and [S55]. However, following the time interval starting from 2000 to 2019 it is evident that the number of the studies with an objective of enhancing the UML diagram notation has been increasing.

The second most studied perspective is comprehension aspect of the UML diagrams which has been the main consideration of 21 studies during 2000-2020. The majority of the studies related to the understandability and comprehension has been conducted between 2015 till 2019, including [S2], [S13], [S43], [S53], [S67], [S75], [S92] and [S93].

The perspective of testing the model ranks the third position among 19 perspectives in terms of the frequency, which has been investigated during 2000-2020 by 13 studies, including [S9], [S21], [S25], [S27], [S47], [S65], [S68], [S70], [S71], [S73], [S77], [S91] and [S102].

Refactoring, Quality assessment approaches and comparison perspectives rank the fourth position with a frequency of 7 for each afore-mentioned perspective.

The papers on UML diagrams quality and understandability with refactoring perspective during 2000 till 2020 were [S5], [S24], [S52], [S66], [S78], [S80] and [S89] and the papers with quality assessment approaches perspective on the quality and understandability of UML diagrams during the study period, were [S10], [S20], [S59], [S74], [S76], [S79] and [S90]. The next perspective with the same frequency was comparison, which has been studied in the paper [S38], [S49], [S57], [S81], [S83], [S95] and [S100].

Two perspectives, namely rules and guidelines and learning challenges and obstacles, ranks the fifth and sixth position respectively, with frequency of 6 and 4. The perspective of studies including [S6], [S60], [S64], [S92], [S99] and [S101] was rules and guidelines for UML diagrams and the perspective of the studies [S12], [S15], [S29] and [S33] was learning challenges and obstacles.

In table 4.15 the seventh position belongs to Refinement and translation which was studied by the papers, including [S27], [S87] and [S98].

The eighth position belongs to the perspectives, including Effectiveness, reading technique, Quality, UML-based Test Generation, Transition technique and Aspect-oriented design, which has been studied by the papers, [S30] and [S84], [S41] and [96], [S3] and [S72], [S56] and [S75], [S33] and [S88], [S22] and [S96], respectively. The least studied perspectives of UML diagrams during 2000 to 2020 are Variability, Metric proposal, Maintainability and Dependability, which were studied by the papers, [S42], [S58], [S86] and [S1], respectively. It is worth noting that as approaching toward 2014 and 2019 the number of studies for all 19 major perspectives has been increasing and during 2000-2004 only seven studies have been carried out on UML diagrams quality and understandability in terms of the 19 major perspectives.

The peak period during which 44 studies focused on the quality and understandability of UML diagrams in terms of the perspectives including Enhancement, Comprehension, Testing the model, Quality Assessment Approaches, Refactoring, Rules and Guidelines, learning challenges and obstacles, Refinement and translation, Quality, UML-based Test Generation, Transition technique and Dependability was 2014 till 2015.

It should be noted some papers have studied the quality and understandability of ML diagrams from more than one perspective, and these studies are: [S33], in which two perspectives of transition and learning challenges and obstacles were studied. In [S38], both comparison and comprehension were considered. The paper [S42] focused on

variability and enhancement aspects. While both studies [S57] and [S100] survey the quality and understandability of UML diagrams from two perspectives simultaneous, namely, comprehension and comparison. Similarly, studies [S92] and [S99] studied the quality and understandability of UML diagrams in terms of comprehension and rules and guidelines at the same time. In [S48], the authors studied the UML diagrams from two perspectives of enhancement and comprehension, and similarly in [S62], the UML modelling language was investigated in terms of quality and comprehension. The study [S63] focused on the UML diagrams considering two perspectives of quality and enhancement, while the study [S75] studied UML notation from two points of view, namely, comprehension and UML-based. The paper [S96], focused on two perspectives of aspect-oriented design and reading technique.

4.3. Quality vs. Understandbaility

The third research question of the present study aimed at answering the research question 3, which was RQ3. *Is understandability a subset of quality in UML diagrams?* The implicit information based on the perspective from which the UML notation is examined and based on the explicit statements and some specific terms, like layout, readability, metric, guideline, cognitive effectiveness, and complexity in the abstract text providing the clues to signify the difference between quality and understandability, RQ3 was answered. Totally, among paper selected for systematic mapping only 31 papers provide the required evidence and clues to make a distinction between quality and understandability, in such a way that understandability should be one of the many sub-factors effective on quality of the design artifacts. In what follows, more details are presented regarding the answer to RQ.3. The evidence demonstrating that comprehensibility is one of the many aspects of quality are tabulated in Table 4.16 along with the associated papers.

Table 4.16 Evidence indicating a difference between the quality and understandability

Evidence	Studies
✓ Dependability is an important quality attribute for today's software systems	[S1]
✓ Comprehension of the rules	[S7]
✓ Pre- and post-conditions.	[S11]
✓ Construct redundancy effect on reader's understanding ✓ Readability	[S13] [S96]
✓ Good communication via good decomposition model	[S19]
✓ The quality and understandability motions are in two interrelated notions	[S38]
✓ Effect of stereotypes on comprehension	[S48]
✓ The cognitive effectiveness of the use case diagram by improving its visual syntax	[S46]
✓ Visual syntax, and proposing a method for enhancing that aspect of UML Use Case diagram ✓ Layout and more on visual aspect of the diagram ✓ Comprehension of UML models in terms of layout quality	[S54] [S69] [S93]
✓ Slicing technique	[S55]
✓ In test generation, they have clearly focused on diagrams understandability and this aspect role in generating a quality test cases	[S56]
✓ The proposed metrics focus on the quality of cohesion aspect of the class diagram	[S58]
✓ The structural complexity	[S59] [S67]
✓ Focused on effect of naming guideline	[S64]
✓ The factor of consistency requirements	[S68]
✓ Achieving quality via validation, verification, and exploration ✓ Validation	[S70] [S91]
✓ Quality can be realized through proof of correctness	[S73]
✓ Focused on timing inconsistencies	[S77]
✓ Proposing a method/checklist for assessing the quality of use case description	[S79]
✓ Refactoring in terms of anti-pattern	[S80]
✓ Effect of composite states on improving the understandability	[S85]
✓ In terms of understandability, verifiability, and effort (time)	[S87]
✓ Quality of class diagram during transition process	[S88]
✓ Providing the modelers with examples. These examples may aid the modelers in terms of diagrams layout, completeness, correctness, semantic, syntax, syntactic	[S92]
✓ The effect of guideline on domain models	[S99]
✓ Work system snap shots	[S101]

For instance, in [S1], the authors point out that dependability is an important quality attribute for today's software systems, and the decisions made at the architectural

design stage play a key role in the dependability of software product. While understandability is also another factor effective on quality. It can be inferred that for attaining a quality design and software, as well as other factors, dependability and understandability are also required.

Not explicitly expressed, but by pondering on the perspective from which the conceptual models in this study are investigated, it can be deduced that comprehension of the rules is one aspect of the diagram's quality, implying that diagrams quality and comprehension are two separate notions, in such a way that understandability is part of quality [S7].

In [S11], the authors have just focused on one aspect of UML models, so called pre and post-conditions. Therefore, it can be concluded that quality of UML artifacts is realized by realization of other factors and aspects of the designs.

In [S13], the main focus was on construct redundancy effect on reader's understanding, while other possible components may also exist that have an impact on the readers' comprehension of UML diagrams [S13].

As, the main objective of the paper is centered on understandability and good communication via good decomposition model, so it can be inferred that understandability is subset of quality [S19].

Comprehension was studied in [S38], which clearly refers to fact the quality and understandability motions are in two interrelated notions, in such a way that comprehension is one of the quality aspects of the models.

In [S46] authors focused on improving the cognitive effectiveness of the Use case diagram by improving its visual syntax. In this regard, it seems that such improvement will contribute to the understandability of the diagram to eschew from any misinterpretation, while other enhancements are also possible, such diagrams naming, space between the diagram's constructs, etc. all of which fall into the major category, called quality [S46].

In [S48], the scholars have just focused on effect of stereotypes on comprehension, and such specific focus on a distinct aspect of UML diagrams triggers that other factors are also exist, all of which are under a superior concept, called quality,

In [S54], the authors have just studied the visual syntax, and proposing a method for enhancing that aspect of UML Use Case diagrams, while other aspects of the use case diagrams can be also studied, such as use case description structure, diagram complexity, number of the use cases per each use case diagram, etc.

Based on the used method, so-called slicing technique it is evident that the author has merely focused on model's understandability, and no statement is made regarding models' visual aspect or other items. Therefore, it can be inferred that understandability is one of the elements of models' quality [S55].

In test generation, they have clearly focused on diagrams understandability and this aspect plays a role in generating a quality test cases. Therefore, understandability and quality are two distinct subjects, but understandability is required to realize the quality of UML diagrams [S56].

In [S58], the authors have clearly stated that the proposed metrics focus on the quality of cohesion aspect of the class diagram, implying that quality for the design models is attainable through realization of different qualitative aspects of UML diagrams.

In [S59], the authors have clearly shown that the structural complexity of the models, is one of the components effective on quality of the models, and specifically on the understandability.

The proposed guideline has just focused on the aesthetic quality of the models, which may be one of the many aspects of the model's quality [S60].

The authors has just focused on model comprehension and the discussed factors effective on the models, implying that there is difference between model quality and comprehension. That is, the comprehensibility is the pre-requisite for having a quality diagram [S61].

The authors of [S64] has merely focused on effect of naming guideline. While other quality attributes, including complexity, color, readability, etc. are not studied. Hence, beside understandability, there are also other aspects from which the UML model's quality is studied.

In this paper [S67], the authors have just focused on complexity of diagram in terms of understandability which shows that the authors make a distinction between quality and understandability.

In [S68], the paper has just focused on the factor of consistency requirements; it is evident that that factor is one of the many factors effective on quality of the models.

The paper [S69] has just focused on layout and more on visual aspect of the diagram, which is related to both understandability and in an upper-level quality of the diagrams.

The paper has mainly focused on achieving quality via validation, verification, and exploration, so it can be inferred that quality is superior to understandability, in terms of level [S70].

The study [S73] mainly focused on quality, as the scholars believe that quality can be realized through proof of correctness. That is, correctness is depended on several factors, all which are sub-sets of model quality, including understandability, testability, maintainability, layout, etc.

The authors of the study [S77] has focused on timing inconsistencies which are special case for real time embedded software, and these are one of the factors of the quality, and other aspects including understandability of diagrams for such system may be studied. Therefore, there is a difference between quality and understandability, in such a way that understandability is subset of quality

The authors of the study [S79] mainly focused on the quality of use case diagrams by proposing a method/checklist for assessing the quality of use case description.

In this study [S80], authors focused on quality of use case models by refactoring in terms of anti-pattern.

The study [S85] studied the effect of composite states on improving the understandability of state diagrams. In fact, using composite states is part of diagram element and it influences diagrams visual aspect, as well as affects the diagrams understandability. While, for realization of the diagram's quality, other factors are also effective. Hence, quality encompasses understandability of modelling diagrams.

In [S87], the authors focused on the quality attributes in terms of understandability, verifiability, and effort (time), which means that quality is made of other sub-factors.

The paper [S88] has focused on quality of class diagram during transition process, which is under the effect of transition technique, categories of developer and the tool.

In [S91], The scholars have focused on validation of the state chart models, and validation mainly deals with quality aspect of the models, and it can be deduced that both validation and verification all are required to guarantee the quality of the diagrams, as a result also understandability [S91].

The authors of the paper [S92], focused on comprehension aspect of the models, as the authors have tried to enhance the diagrams perception through providing the modelers with examples [S92]. These examples may aid the modelers in terms of diagrams layout, completeness, correctness, semantic, syntax, syntactic, etc. So, even for realization of understandability there are also other sub-factors, as the understandability itself is sub-factor for quality.

In a study by Störrle [S93], the main focus was centered on the comprehension of UML models in terms of layout quality, which is the only one aspect of quality. So, it can be construed that understandability is sub-set of quality.

In a research by Ali, Yue and Briand [S96], the authors focused on readability, consequently on models' comprehension. Therefore, it is clear that still there are other factors effective on diagrams quality. All these factors are required to meet a specific level of properness to fulfill the quality of models.

In a study by Bera and Evermann [S99], the main focus was on the effect of guideline on domain models. The available and proposed guidelines in the literature not only enhance the model's understandability, but also the ultimate goal in devising guideline is to improve models' quality.

The authors focused on quality and enhancing it via work snap shots [S101]. Providing the modelers with work system snap shots, aid him/her in meeting the quality of diagrams, in terms of all sub-sets, including understandability.

CHAPTER 5

DISCUSSION

In the chapter 4, the answers to three research questions are provided along with providing the related data and evidence. In this chapter, the summary of the findings and the potential implication for software engineering industry, particularly for the software engineers, analysts, modelers, and design reviewers are presented. In this regard, the backbone of the present chapter will be based on chapter 4, where the analysis results of 103 studies conducted during the period of 2000 to 20020 were tabulated and interpreted.

5.1. Important factors for quality UML artifacts

By reviewing 103 studies carried out during 2000-2020, 13 factors as the main group of the factors were identified which are effective on the quality and understandability of UML diagrams. For each of these factors at least two or more studies have been conducted, and these studies have presented the findings and observations for these factors impact on the quality and understandability of UML diagrams. However, there have been several factors which have been studied by at least 90 studies, which signifies the importance of these factors among the scholars. Factors including modelling, diagram, evaluation, representations, software engineer, guideline and testing are among the highly important factors in a discerning the order for generating quality UML artifacts.

5.1.1. Modelling

Each of 17 studies whose main concern was modelling, [S2], [S3], [S19], [S22], [S30], [S31], [S35], [S38], [S49], [S55], [S57], [S67], [S78], [S81], [S96], [S98] and [S100] have focused on different sub-factors of the modelling factor, including the effect of conceptual mental model on quality of use case diagram and actors in terms of correctness, completeness, and redundancy of the UCD, aspect-oriented design, effect of GDM, effect of model types (OO-DFD and Use Case Methods) on comprehension during functional requirement (FR) modelling, domain modeling techniques, dynamic models, impact of aspect-oriented modelling on readability of UML state machines,

effect of modelling language and notation, effect of process modelling notations in terms of intuitive complexity, models slicing, effect of multi-view refactoring approach on models quality, level of normality/precision in workflow modeling, Conallen's notation and developers' experience and ability, part-whole relationship and the effects of VCL on visual modelling quality. For instance, in [S2], it was proved that using the precise style it is possible to design more comprehensible artifacts, and this has an implication for the modeler and designer in such a way that, considering the nature of the software project and application, there are two extremes styles for modelling artifacts, namely, precise style, which is dominated by the specific rules and constraints, and ultra-light style, where there is no rule and no constraint. That is, during the modelling, the style should be selected and customized in accordance with the intended project specification and the developers' needs. The findings of the study [S3] are also of great importance for the modelers and analysts who are responsible to communicate with the stakeholders, as they should use domain models, and the quality of these models is under the effect of two common techniques, namely category list technique and non-phrasing technique. The findings in [S3], ascertained that the noun phrasing technique exhibits better results for identification of the concepts, since, using the afore-mentioned technique the modelers are able to produce more comprehensive domain models in a timely manner, while the correctness and completeness of the domain models produced by both techniques do not differ. Another sub-factor important for modelling factor was identified by the study [S19], named good decomposition model that may be a good index for appraising the comprehension of the models. One of main contribution to the modelling field has been made by [S22], where the authors studied the effect of aspect-orientation as a new approach for modularization, which may be useful for separating the crosscutting concerns from non-crosscutting ones, however, for the modelers it is highly suggested to use aspect-oriented modelling approach in accordance with specific context of the project. Study [S96] also suggested aspect-oriented modelling as an effective factor on the readability of UML state machines. In fact, using the aspect-oriented modelling the modelers and analysts will be able to find and fix the defects in a cost-effective manner. The findings of study [S31], suggest the use of Conallen's Web Application Extension (WAE) notation stereotypes to reduce the gap between the modelers and developers in terms of ability and experience, if they are cooperating in the same project. Another important finding was obtained by study [S38], as they focused on the effect of

modelling languages on the quality of the fabricated diagrams. For the modelers, the use of OPM modelling language is suggested in web application development projects, especially for dynamic behavior modelling. In a study by Otero and Dolado [S49] the same direction was followed, which showed that for dynamic behavior modelling, it would be better to use OML language, to produce more comprehensible models which are easily perceived by the developers and reviewers. Moreover, the findings of the study [S57] are also important for the software industry, especially modelling experts, who want to develop a software based on the principles of object-oriented modelling, as they will need to obtain a good understanding of the system main functional requirements. In this case, they have generally two options, namely, use case and Tropos, the latter is based on a goal, while the former is based on a scenario. The findings showed that Tropos excel use case, in terms of comprehensibility. For the process modelling, the selection of the modelling language should be made considering the process complexity. To put it simpler, the findings of [S67] showed that for the process with less complexity, the use of BPMN notation is suggested, while for the processes with higher complexity EPC is suggested to be more suitable, compared to UML activity diagram. The findings of the study [S81] are also useful for the modelers, as it proved the role of VCL in increasing the cognitive effectiveness of the diagrammatic representations. Indeed, VCL is intended for enhancing the visual modelling with a main focus on cognitive aspects of the models. The last study related to the modelling factor, [S100], shades a light on two available options for modelling the functional requirements, namely UML use case and objected oriented-DFD. It is suggested that the modelers should use case diagrams if their main considering is quality, as OO-DFD models involve high details artifacts with high error prone rate.

5.1.2. Diagram

The diagram factor was studied in terms of various sub-factors by studies [S97], [S80], [S45], [S58], [S54], [S85], [S43], [S1], [S52], [S61], [S88], [S72] and [S11]. In analyzing developing the critical systems, the dependability should be taken into account by the modelers and developers, since reliability and availability of critical software systems are highly significant, for which a method was proposed to evaluate the dependability of UML/ Dependability Modeling and Assessment (DAM) models using hierarchical and colored extension of stochastic activity network, which can be used by the modelers to control the dependability at the architectural level, and it

implicitly enhances dependability of the whole software system. According to [S11], now the modelers can use method for developing a comprehensive object constraint language post-condition. According to [S43], it is suggested to modelers to use the object diagrams along with class diagram for better capturing of the concepts of the problem domain. However, the modelers should note that the benefits of use of object diagram are related a bonded to the modelers' experience and familiarity with UML notation. Now, the modelers can use artificial intelligence to detect the intra-model design inconsistencies, and the details of this method are introduced by [S52]. Every diagram is consisted of icons and color, and it is suggested that the visual appearance of the diagrams is important in terms of cognitive effectiveness in the field of the software engineering and the findings of the study [S54] showed that the use of color and icons boosts the readability of misuse case diagram. Another implication for the modelers, regarding [S61] is the fact that the level of detail for all diagram types should be assessed and the degree of the details should be determined in accordance with the software project type. Modelers should also be cognizant of the positive effect of composite states in terms of better understandability of UML state chart diagrams [S85]. Also, the modelers should never ignore the transition method effect on the quality of class diagrams, as there are generally two accepted transition methods, from use case to class diagram: derivation method and validation method. Based on the findings and observations [S88], the use of validation method is suggested for generating more proper class diagrams, while the category (experience and ability level) of the developers should consider.

5.1.3. Evaluation

The next factor was evaluation which is important for UML diagrams quality and understandability, as it was studied by 11 studies, including [S86], [S26], [S102], [S25], [S76], [S9], [S53], [S4], [S93], [S16] and [S103], in terms of class diagram measures, metric visualization, design testability, metric, model evolution approach, verification based on ontology, feedback-inclusive prototype, size and complexity of UML diagrams and meta association impact on meta-models. All these studies make a separate and specific contribution to the field of model-based software development approach, particularly based on UML modeling language. Similar to diagram factor, in the studies focused on the evaluation factor, the complexity and size of the intended system again was studied, and it can be inferred that the experts of the modelling phase

of the software development life cycle, should pay a special attention on the size of the software for which the modelling artifacts are going to be created. For the system with a great size and high complexity, the HITS algorithms are used to only extract the important use cases and key requirements [S4]. The modelers should always consider the correctness of the models in terms of consistence rules and system requirements, which can be realized using domain model [S16]. Nowadays, the use of prototyping technique based on feedback is gaining interest among the experts and it can be used instead of manual inspection of the state charts, since feedback-inclusive rapid prototyping method build up more quality state charts to gain a good perception of the system dynamics [S53]. It is every important that both modelers and developers have a proper understanding of the metrics which are used for assessing the structure, complexity, and quality of the models. In this regard, one of the methods is 3D visualization of the metrics [S26]. In evaluation as an effective factor on the quality and understandability of UML diagrams, one of the important sub-factors is evolution approach, for which a novel approach was proposed by [S76]. The authors of [S76] clearly point out that the available evolution approaches do not consider the reuse aspect of the models sufficiently. The proposed evolution approach considers the quality in terms of dimensions including, syntactic, semantic, pragmatic and emotional. The modelers can also enhance the quality of class diagrams using predication models based on the early measures [S86]. The findings show that the increase in diagram size in UML notation, makes the modeler visual and mental ability to be stretched and the result of the design phase will be poor quality models [S93].

5.1.4. Software engineer

Software engineer, as the human factor plays a key role in fabrication of quality and comprehensible UML models. The studies on this factor are [S30], [S12], [S14] [S15], [S33], [S44], [S20], [S41], [S48], [S84] and [S39]. As the UML artifacts are created, it should be noted that the modelers', analysts', reviewers' and developers' ability and experience have an impact on models' quality and understandability [S30]. While there are also some learning difficulties associated with UML notation, which may be resolved using the participatory cognitive approach for creation of the shared understanding between the instructors and students on difficult materials [S12] or for enhancing the performance of the beginner analyst, CHOP method can be used during the creation of sequence diagram [S14]. Moreover, there are some learning difficulties

with UML sequence diagram which should be facilitated when educating software engineer students. These difficulties are unfamiliar metaphors, working with SD syntax (SDs have a detailed and very specific syntax), omitting explicit returns (The notion of flow of control through the logic is not evident in an experienced analyst's SDs, since explicit return notations are often omitted to reduce clutter and fit more logic into a given space), Decomposing high-Level Requirements into Logic Steps, Choosing to Centralize or Decentralize Responsibility Assignments, and Rate of Information Presentation [S15]. Other difficulties which should be taken into account by the instructor of the software engineering discipline are related to transition from object-oriented analysis to logical design. Study [33] showed that due to two semantic distance and precise degree of transition related, the learners and engineers face with difficulties during transition from object-oriented analysis to logical design. Moreover, the options and utilities of the design tool used by the modeler are effective on the quality of the diagram. It is suggested to use a software design tools which is equipped with an option for supporting the cognitive tasks like decision making, decision ordering and task-specific design understanding [S39]. The modelers, reviewers and developers should also pay a heed to the reading method that they use, since PBR reading technique is superior to CBR method in error detection and it is cost-effective [S41.].

5.1.5. Representation

The representation factor identified in 10 studies ([S6], [S7], [S8], [S29], [S46], [S51], [S64],[S69],[S79] and [S87]) among 103 review papers as one of the 13 factors effective on the quality and understandability, clearly refers to the fact that there are some defects in UML modelling notation which can be resolved by applying some amendments and techniques, including use of iconic signs as UML graphical notations, use of naming principles for conceptual elements, representing the roles in conceptual models, using SORT, proposing a checklist for use case description quality, graphical and textual notation to represent non-functional requirements, extending the UML state charts notation to model security aspects, customization of the process models representation to resolve lack of fit and representing physical quantities in software models. For instance, in [S6], it was shown that when describing an organization or part of it, a key concept is the notion of roles played by actors in the domain and probably the roles can be declared in the form a conceptual modelling construct and a

set of guidelines for representing roles in conceptual models can be defined to suggest rules for modelling roles with the widely used extended entity-relationship grammar. Also, using the iconic signs in the UML graphical notation it is possible to have more proper interpretation of the models and the number of the connotations will be decreased [S7]. Moreover, the UML notation lacks the required support for non-functional requirements and such amendment is required which will expedite the translation of the requirements into the design [S8]. As a final point, in [S79], the authors notify the modelers and developers on the quality control of the use case description, which may have great impact on the quality of the use case diagrams. It can be inferred that either through a direct application or using the guidelines as the use case guidelines, the quality of the use case description will be a good index to evaluate and boost the quality of the use cases.

5.1.6. Guideline

Other important factor which is crucial for quality and understandability of UML diagrams is guideline which insist on the use of explicit reuse guidance and/or variability specification for domain models, rules and guidelines in terms of abstraction rules, theory-guided on learning, visual guidelines on aesthetic quality of the diagrams, knowledge-based object modeling advisor, ontological modelling guideline, effect of using examples on structural model comprehension and effect of work system snapshots.

5.1.7. Testing

Testing is the last important factor, which encompasses both validation and verification of the UML models. In the reviewed studies, the testing factor was introduced to be effective on the UML diagrams quality and understandability, during early stage of state chart models, or testing of the domain models, or testing the models in terms of semantic correctness, testing the models based on light weight and statics verification and considering the security requirements, all of which imply that testing can be done in various phases, from various perspectives, like testing time, the artifact type to be tested and testing approach.

5.2. UML modelling notation enhancement

In chapter 4, 103 papers were analyzed to perceive the lens from which the quality and understandability of UML diagrams were studied. In this regard, 19 major perspectives were identified and in Table 4.15, the number of the studies for each perspective during 2000-2004, 2005-2009, 2010-2014, 2015-2019 and 2020 is presented. As it was seen in Table 4.15, approximately one to third of the studied have sought to improve and enhance the quality of UML notation, implying that there are some defects in UML modelling notation, such as representation of non-interactive requirements along with interactive ones (use cases) in one model [S8], proposing a new method for developing comprehensive OCL post-conditions for operations in UML and OCL models [S11], proposing a technique called "CHOP" (Chunking, Ordering, Patterning), which is designed to improve novice analyst performance in modeling an SD [S14], proposing a method addressing both completeness and correctness problems of UML models by adopting a domain analysis approach called application-based domain modeling [S18], presenting the abstraction rules and an algorithm for applying them [S23], proposing a new approach for three-dimensional visualization (including extraction and display) of design phase object-oriented software metrics [S26], propose a new notational set that extends the UML state charts notation for modelling the security aspect of the system [S28], introducing an approach that enhances requirements reuse while capturing the static (functional requirements, use case diagrams) and dynamic (activity diagrams) view of software projects [S36]. Methods to make distributed synchronous modeling and communication among team members easy [S44], improving the use case diagram notation with main a focus on boosting its cognitive effectiveness [S46], introducing the main concepts and mechanisms needed for representing and handling physical quantities in software models [S51], promoting the use of model slicing techniques to build interactive visualization tools for meta-models [S55], proposing the layout technique Vertically Aligned Class Diagrams, incorporating the usability of Parallel Virtual Machine (PVM) into class diagrams [S69], improving modelers' capabilities by constructing a catalog of anti-patterns of correctness and quality problems in class diagram [S94], improving use case diagrams using anti-patterns [S97] and boosting the quality of UML metamodel extensions through concept of induced association [S103].

5.3. Quality vs. Understandability in UML diagrams

The third research question of the present research was an attempt to clarify that for UML diagrams, understandability does not encompass quality. It can be claimed that regarding the McCall's factor model [38], who states that the quality of a software can be appraised in terms of 11 factors, the quality of the UML artifacts can be also assessed in terms of McCall's factor model: Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Flexibility, Testability. Portability, Reusability, Interoperability. To put it simpler, attaining quality UML designs imply that all other quality attributes are also realized. However, in answering the third question of the present research, the abstract of 31 studies were analyzed in terms of identifying the key terms as clues for discerning which perspectives of UML notation are studied and whether there is any statement pointing out the distinction between quality and understandability in UML diagrams. To this end, the abstract of studies including [S7], [S11], [S13], [S96], [S19], [S38], [S48], [S46], [S54], [S69], [S93], [S55], [S56], [S58], [S59], [S67], [S64], [S68], [S70], [S91], [S73], [S77], [S79], [S80], [S85], [S87], [S88], [S92], [S99] and [S101] were analyzed and the key terms and explicit statements presented in Table 4.16 were extracted. The findings showed that understandability should counted as a subset of quality, like all other quality attributes proposed my McCall's model for quality assessment of a software.

5.4. Comparison with preceding SLR studies on UML

One of the findings of the present thesis is in line with that of [6], since both studies found that there has been increase in the number of the papers starting from 2013. One of the main differences was also related to the fact that Dikici et al. [6] reviewed both journals and conferences papers, while we only studied the well-esteemed journals papers. The factors that they showed to be effective on the quality of the process models were modelling notation, structural complexity, modularity, modelling approach, visual layout, model element labelling, model element design, guidelines, model annotation, modelling construct, coloring, and process perspective representation. Other factors were grouped as the personal factors, including modelling expertise, knowledge, experience, cognitive abilities, learning style, learning motive, learning strategy and domain familiarity.

The other study carried out by Merino et al. [7] also was a systematic literature review, but they focused on software visualization evaluation. They point out that almost 62% of the proposed software visualization approaches do not incorporate a reliable evaluation procedure.

Mohaghegghi et al. [9] conducted a SLR on the studies whose main concern was the quality of models to gain an insight into what is the meaning of quality for the models and what are the methods to enhance it. Their findings showed that quality of the models is defined in terms of six aspects, namely, correctness, completeness, consistency, comprehensibility, confinement and changeability, and this finding is in line with the findings of the present work, considering the 13 factors that were identified in the present study as the effective factors on the quality and understandability.

Haug et al. [12] stated that during the years 2000 to 2004, there was no trend in any direction based the studies they have reviewed They just point out that almost half of the studies was found in 2005, while other studies were distributed in years 2000, 2001, 2002, 2003 and 2004. It is interesting to note that in that study also there was an increase in the number of the publications on UML in 2005. In that study, it was also observed that between 2000 to 2005, the UML diagrams were evaluated from aspects, including accuracy, understandability and efficiency, construction completeness and structure, defect detecting efficiency which all are in line with findings of the present research. However, in that study, it is stated that between 2000 to 2005 the comprehension aspect is the most studied aspect while the findings of the present study ascertained that during 2000 to 2020 the most studied aspect of the UML modelling language has been enhancement perspective. As the factors effective on the understandability of UML diagrams, they have also observed that comprehension of UML models is affected by the composite states, diagram type and the complexity.

In a study by Lucas et al. [13], it was shown that the topic of UML model consistency has gained interest among the scholars and they demonstrated that there is a need to propose a solution for dealing with new models and probable inconsistency problem.

Fernández-Sáez et al. [14] found that there are several factors affecting the maintainability of a system, both source code and diagrams. However, some of these factors have a positive influence, while others have a negative influence. In that study, it is stated that understandability directly affects the maintainability. Therefore, they point out that the factors that have an influence on understandability also have an effect

on the maintainability. According to [14], the whole system in terms of maintainability is under the impact of both source code and the related documentation. For instance, they point out that OPM is a suitable choice for the context of modelling the dynamic aspect of the web application. While other variants of UML like UML-B positively influence the maintainability as they make the perception easy. Similar to the findings of the present study, they also found that the maintainers' skill and experience have an impact on the maintainability of a system. Another similar finding was the impact of the complexity of diagram for the system, in such a way that when the system possesses a high complexity, the maintainability is influenced negatively. Another similar result to the present study was the effect of level of detail in UML diagrams, which may affect both source code and diagrams, implying that it is desirable to have a high level of detail. Finally, they ascertained that any enhancement in the aesthetic diagram layout leads to improvement in both maintainability and understandability of UML diagrams.

Polančič and Cegnar [15] studied the process models from point of view of complexity metrics. In their opinion, the understandability of visual models is depended on the degree of the complexity.

5.5 Threats to validity

Construct validity. The research questions of the present study may not cover all the potential study aspects of UML notation, but in order to reduce this threat it was designed through three research questions. The main focus is centered on the factors effective on the quality and understandability of UML diagrams, the border between the understandability and quality in UML diagrams and the studies perspectives for UML notation.

Internal validity. It is worth noting that the act of grouping the subfactors under the main 13 factors is a subjective process and in other studies, the same subfactors may be grouped differently. Even the factors are named by three reviewers in this current study, it should be noted that still there is a level of subjectivity in this approach. This matter should be considered as one of the limitations of the present study. The other limitation and threat to the internal validity of the result is related to the search string. In the search string, we have included "class diagram" and "UML" as key terms of search string, which may lead to excluding some related studies from the corpus. For

instance, in the future studies, the terms like “activity diagram” and “sequence diagram” can be included explicitly in the search term.

External validity. Of course, as we have selected the studies from only one scientific database, so-called Web of Science, we may have missed some papers indexed in Scopus or Google Scholar and this fact may affect the generalizability of the obtained result. However, as Web of Science is indexing high quality studies, the scope of the current study should be taken into account under these considerations.

CHAPTER 6

CONCLUSION

The present research aimed at studying the available literature on the topic of quality and understandability of UML notation. For this purpose, the method of systematic mapping was employed.

For decreasing the risk of replications and the effort needed for exclusion of those repeated papers, only one scientific database, so-called Web of Science was the source of the studied papers. The systematic mapping was exerted on 103 papers and the findings of the review can be summarized as below:

1) In answering the first question, totally 13 factors were identified with possible effect on the quality and understandability of UML language, while the number of the studies for these factors varied. These factors were Diagram, Diagram Type, Evaluation, Guideline, Modelling, Quality Assessment Approach, Redundancy, Refactoring, Representations, Requirements, Software Engineer, System Structure and Testing (Figure 6.1)

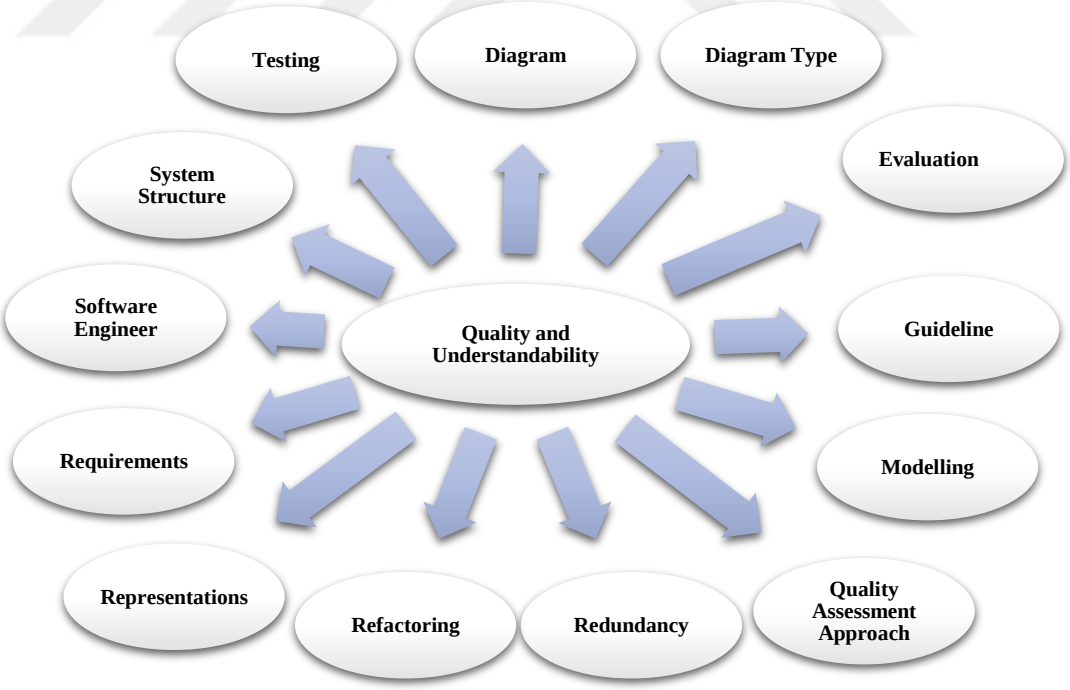

Figure 6.1 Factors effective on the quality and understandability of UML diagrams

2) By reviewing the studies on the diagrams quality and understandability of the UML notation conducted 2000 to 2020, it was observed that the previous

studies can be categorized in terms of major 19 themes, namely, Enhancement, Comprehension, Testing the model, Quality Assessment Approaches, Refactoring, Comparison, Rules and Guidelines, Learning Challenges and Obstacles, Refinement and Translation, Effectiveness, Reading Technique, Quality, UML-based Test Generation, Transition Technique, Aspect-oriented Design, Variability, Metric Proposal, Maintainability and Dependability. It should be noted that enhancement is the most studied aspect of UML notation among 103 papers, which is followed by comprehension, testing, refactoring, comparison, quality assessment approach, etc. During the time period of 2010 to 2020 there has been an increase in the number of the studies on UML diagrams.

The findings of the present study shade a light on the factors effective on the quality and understandability of UML diagram as well as the mostly studied aspects and the relation between quality and understandability, however these findings, particularly the afore-mentioned factors, their impact mode; positive or negative impact per the application domain, may need to be studied experimentally by the prospective studies. Moreover, a comprehensive guideline may be also proposed, and the generated UML artifact based on this guideline may be assessed using eye tracker to get an insight into the mental load, cognitive process and the effectiveness of the proposed guideline, from point of view of modelers, software engineers and analysts.

REFERENCES

[1] V. Vranić *et al.*, "Challenges in preserving intent comprehensibility in software," *Acta Polytech. Hungarica*, vol. 12, no. 7, pp. 57–75, 2015.

[2] J. Coplien and G. Bjørnvig, *Lean Architecture for Agile Software Development*. Cornwall: Wiley, 2010.

[3] J. Kabeli and P. Shoval, "Comprehension and quality of analysis specifications - A comparison of FOOM and OPM methodologies," *Inf. Softw. Technol.*, vol. 47, no. 4, pp. 271–290, 2005.

[4] P. Bera, P. Soffer, and J. Parsons, "Using eye tracking to expose cognitive processes in understanding conceptual models," *MIS Q. Manag. Inf. Syst.*, vol. 43, no. 4, pp. 1105–1126, 2019.

[5] A. Burton-Jones, J. Recker, M. Indulska, P. Green, and R. Weber, "Assessing representation theory with a framework for pursuing success and failure," *MIS Q. Manag. Inf. Syst.*, vol. 41, no. 4, pp. 1307–1333, 2017.

[6] A. Dikici, O. Turetken, and O. Demirors, "Factors influencing the understandability of process models: A systematic literature review," *Inf. Softw. Technol.*, vol. 93, no. 2018, pp. 112–129, 2018.

[7] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "A systematic literature review of software visualization evaluation," *J. Syst. Softw.*, vol. 144, pp. 165–180, 2018.

[8] B. A. Kitchenham *et al.*, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, 2002.

[9] P. Mohagheghi, V. Dehlen, and T. Neple, "Definitions and approaches to model quality in model-based software development - A review of literature," *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1646–1669, 2009.

[10] M. El-Attar and J. Miller, "Constructing high quality use case models: A systematic review of current practices," *Requir. Eng.*, vol. 17, no. 3, pp. 187–201, 2012.

[11] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, and R. Pretorius, "Empirical evidence about the UML: A systematic literature review," *Software - Practice and Experience*, vol. 41, no. 4. 2011.

[12] T. H. Haug, "A Systematic Review of Empirical Research on Model-Driven Development with UML," MSc Dissertation, Department of Informatics, University of Oslo, February, 2007.

[13] F. J. Lucas, F. Molina, and A. Toval, "A systematic review of UML model consistency management," *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1631–1645, 2009.

[14] A. M. Fernández-Sáez, M. Genero, and M. R. V. Chaudron, "Empirical studies

concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study," *Inf. Softw. Technol.*, vol. 55, no. 7, pp. 1119–1142, 2013.

[15] G. Polančič and B. Cegnar, "Complexity metrics for process models – A systematic literature review," *Comput. Stand. Interfaces*, vol. 51, pp. 104–117, 2017.

[16] A. Boland, M. G. Cherry, and R. Dickson, *Doing A Systematic Review*. Liverpool: Sage, 2017.

[17] D. Gough, S. Oliver, and J. Thomas, *An introduction to systematic reviews*. Liverpool: Sage, 2017.

[18] O. M. Group, *Unified Modeling Language (UML)*. 2015. [Online] Available: https://www.omg.org/spec/UML/About-UML/. [Accessed: 10- May- 2021].

[19] K. Kruczynski, "Business process modelling in the context of SOA - An empirical study of the acceptance between EPC and BPMN," *World review of science, technology and sustainable development*, vol. 7, no. 1–2, pp. 161–168, 2010.

[20] K. Van Hee, O. Oanea, and N. Sidorova, "Colored petri nets to verify extended event-driven process chains," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3760 LNCS, pp. 183–201. 2005.

[21] B. Bruegge and A. H. Dutoit, *Object-oriented software engineering : using UML, patterns and Java*. New York: Pearson, 2003.

[22] M. Fowler and K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition) 2nd*. Boston: Addison-Wesley Pub (Sd), 2010.

[23] L. Briand, E. Arisholm, S. Counsell, F. Houdek, and P. Thévenod–Fosse, "Empirical studies of objectoriented artefacts, methods and processes: state of the art and future directions," *Empir. Softw. Eng.*, vol. 4, no. 4, pp. 387–404, 2000.

[24] V. R. Basili *et al.*, "The empirical investigation of perspective-based reading," *Empirical Software Engineering*, vol. 1, no. 2. pp. 133–164, 1996.

[25] M. E. Fagan, "Design and code inspections to reduce errors in program development," *IBM Syst. J.*, vol. 38, no. 2, pp. 258–287, 1999.

[26] D. Graham and T. Gilb, *Software Inspection*. Boston: Addison-Wesley, Reading, MA. 1993.

[27] J. Dong, S. Yang, and K. Zhang, "Visualizing design patterns in their applications and compositions," *IEEE Trans. Softw. Eng.*, vol. 33, no. 7, 2007.

[28] I. Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach. Wokingham*. Boston: Addison-Wesley Next page, 1993.

[29] C. Atkinson and D. Muthig, "Component-based product-line engineering with the uml," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2319, pp. 343–344, 2002.

[30] C. Bunse and C. Atkinson, "Improving Quality in Object-Oriented Software: Systematic Refinement and Translation of Models to Code," in *IESE Report*, no. 036.99/E, p. 35, 1999.

[31] C. Bunse and C. Atkinson, "The normal object form: Bridging the gap from models to code," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1723, 1999.

[32] Y. Wand and R. Weber, "On the ontological expressiveness of information systems analysis and design grammars," *Inf. Syst. J.*, vol. 3, no. 4, pp. 217–237, 1993.

[33] G. Guizzardi, "Ontological Foundations for Structural Conceptual Models," University of Twente, PhD Dissertation, Centre for Telematics and Information Technology, University of Twente 2005.

[34] D. Moody, "The physics of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 756–779, 2009.

[35] M. Bunge, *Ontology 1: The Furniture of The World*. D. Reidel.[PC], 1977.

[36] B. Boehm, "Software risk management, in: C. Ghezzi, J. A. McDermid (Eds.), Proc. 2nd European Software Engineering Conf. (ESEC 1989)," pp. 1–19, 1989.

[37] L. C. Briand, J. Wuest, S. V. Ikonomovski, and H. Lounis, "Investigating quality factors in object-oriented designs: An industrial case study,". In *Proceedings of the 21st international conference on Software engineering*, pp. 345-354. 1999.

[38] D. GALIN, *Software Quality Assurance From theory to implementation*. New York: Pearson, 2004.

[39] O. Laitenberger and J. M. Debaud, "Encompassing life cycle centric survey of software inspection," *J. Syst. Softw.*, vol. 50, no. 1, pp. 5–31, 2000.

[40] B. Meyer, "Seven principles of software testing," *Computer (Long. Beach. Calif).*, vol. 41, no. 8, pp. 99–101, 2008.

APPENDIX A

List of Primary Studies Included in the Systematic Literature Review

[S1] A. Sedaghatbaf and M. Abdollahi Azgomi, "A method for dependability evaluation of software architectures," *Computing*, vol. 100, no. 2, pp. 119–150, 2018.

[S2] G. Reggio, F. Ricca, G. Scanniello, F. Di Cerbo, and G. Dodero, "On the comprehension of workflows modeled with a precise style: results from a family of controlled experiments," *Softw. Syst. Model.*, vol. 14, no. 4, pp. 1481–1504, 2015.

[S3] H. Nisa, S. Imtiaz, M. Uzair, and S. Imtiaz, "Impact of Domain Modeling Techniques on the Quality of Domain Model: An Experiment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 11, pp. 52–61, 2016.

[S4] S. Sabharwal, P. Kaur, and R. Sibal, "Applying page rank and HITS algorithm to identify key use cases," *J. Appl. Sci. Eng.*, vol. 21, no. 4, pp. 655–662, 2018.

[S5] M. Misbhauddin and M. Alshayeb, "An integrated metamodel-based approach to software model refactoring," *Softw. Syst. Model.*, vol. 18, no. 3, pp. 2013–2050, 2019.

[S6] P. Bera, A. Burton-Jones, and Y. Wand, "Improving the representation of roles in conceptual modeling: theory, method, and evidence," *Requir. Eng.*, vol. 23, no. 4, pp. 465–491, 2018.

[S7] K. Siau and Y. Tian, "A semiotic analysis of unified modeling language graphical notations," *Requir. Eng.*, vol. 14, no. 1, pp. 15–26, 2009.

[S8] S. Iqbal, I. Al-Azzoni, G. Allen, and H. U. Khan, "Extending UML use case diagrams to represent non-interactive functional requirements," *E-Informatica Softw. Eng. J.*, vol. 14, no. 1, pp. 97–115, 2020.

[S9] A. Hafeez, S. Hyder, A. Musavi, and A. U. R. Rehman, "Ontology-Based Verification of UML Class/OCL Model," vol. 37, no. 4, pp. 521–534, 2018.

[S10] R. Matulevičius, H. Lakk, and M. Lepmets, "An approach to assess and compare quality of security models," *Comput. Sci. Inf. Syst.*, vol. 8, no. 2, pp. 447–476, 2011.

[S11] N. Desai and M. Gogolla, “Developing comprehensive postconditions through a model transformation chain,” *J. Object Technol.*, vol. 18, no. 3, pp. 1–18, 2019.

[S12] Z. Shen, S. Tan, and K. Siau, “Challenges in learning unified modeling language: From the perspective of diagrammatic representation and reasoning,” *Commun. Assoc. Inf. Syst.*, vol. 43, no. 1, pp. 545–565, 2018.

[S13] P. Bera and G. Poels, “The effects of construct redundancy on readers’ understanding of conceptual models,” *J. Database Manag.*, vol. 28, no. 3, pp. 1–24, 2017.

[S14] T. Syn and D. Batra, “Improving sequence diagram modeling performance: A technique based on Chunking, Ordering, and Patterning,” *J. Database Manag.*, vol. 24, no. 4, pp. 1–25, 2013.

[S15] D. VanderMeer and K. Dutta, “Applying learner-centered design principles to UML sequence diagrams,” *J. Database Manag.*, vol. 20, no. 1, pp. 25–47, 2009.

[S16] I. Reinhartz-Berger and A. Sturm, “Enhancing UML Models,” *J. Database Manag.*, vol. 19, no. 1, pp. 74–94, 2011.

[S17] B. R. Object and O. Design, “Business Rules-driven Object BROOD :,” vol. 19, no. March, pp. 41–73, 2008.

[S18] X. Dolques, M. Huchard, C. Nebut, and P. Reitz, “Fixing generalization defects in UML use case diagrams,” *Fundam. Informaticae*, vol. 115, no. 4, pp. 327–356, 2012.

[S19] A. Burton-Jones and P. N. Meso, “Conceptualizing systems for understanding: An empirical test of decomposition principles in object-oriented analysis,” *Inf. Syst. Res.*, vol. 17, no. 1, pp. 38–60, 2006.

[S20] B. Mańin, G. Giachetti, O. Pastor, T. E. J. Vos, and A. Abran, “Using a functional size measurement procedure to evaluate the quality of models in MDD environments,” *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 3, 2013.

[S21] A. Queralt and E. Teniente, “Verification and validation of UML conceptual schemas with OCL constraints,” *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 2, 2012.

[S22] M. Wimmer, A. Schauerhuber, G. Kappel, W. Retschitzegger, W. Schwinger, and E. Kapsammer, “A survey on UML-based aspect-oriented design modeling,” *ACM Comput. Surv.*, vol. 43, no. 4, 2011.

[S23] A. Egyed, “Automated abstraction of class diagrams,” *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 4, pp. 449–491, 2002.

[S24] T. H. Dao, X. T. Nguyen, and N. T. Truong, “Preservation of Class Invariants in Refactoring UML Models,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 29, no. 2, pp. 221–244, 2019.

[S25] E. Ramaraj and S. Duraisamy, “Design optimization metrics for UML based object-oriented systems,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 17, no. 3, pp. 423–448, 2007.

[S26] A. Lakshminarayana, “Three-dimensional visualization of object-oriented software design metrics,” *J. Electron. Imaging*, vol. 9, no. 4, p. 368, 2000.

[S27] M. Autili, A. Bertolino, G. De Angelis, D. Di Ruscio, and A. Di Sandro, “A Tool-Supported Methodology for Validation and Refinement of Early-Stage Domain Models,” *IEEE Trans. Softw. Eng.*, vol. 42, no. 1, pp. 2–25, 2016.

[S28] M. El-Attar, H. Luqman, P. Karpati, G. Sindre, and A. L. Opdahl, “Extending the UML Statecharts Notation to Model Security Aspects,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 7, pp. 661–690, 2015.

[S29] B. M. Samuel, L. Watkins, A. Ehle, and V. Khatri, “Customizing the representation capabilities of process models: Understanding the effects of perceived modeling impediments,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 1, pp. 19–39, 2015.

[S30] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, “Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 327–342, 2013.

[S31] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato, “How developers’ experience and ability influence web application comprehension tasks supported by UML stereotypes: A series of four experiments,” *IEEE Trans. Softw. Eng.*, vol. 36, no. 1, pp. 96–118, 2010.

[S32] J. P. Davis and R. D. Bonnell, “Propositional logic constraint patterns and their use in UML-based conceptual modeling and analysis,” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 427–440, 2007.

[S33] S. S. Shin, “A study on the difficulties of learning phase transition in object-oriented analysis and design from the viewpoint of semantic distance,” *IEEE Trans. Educ.*, vol. 58, no. 2, pp. 117–123, 2015.

[S34] B. Karasneh, R. Jolak, and M. R. V. Chaudron, “Using examples for teaching software design: An experiment using a repository of UML class diagrams,” *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2016-May, pp. 261–268, 2016.

[S35] R. Motschnig-Pitrik and J. Kaasbøll, “Part-whole relationship categories and their application in object-oriented analysis,” *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 5, pp. 779–797, 1999.

[S36] T. Diamantopoulos and A. Symeonidis, “Enhancing requirements reusability through semantic modeling and data mining techniques,” *Enterp. Inf. Syst.*, vol. 12, no. 8–9, pp. 960–981, 2018.

[S37] P. Bera and G. Poels, “How quickly do we learn conceptual models?,” *Eur. J. Inf. Syst.*, vol. 28, no. 6, pp. 663–680, 2019.

[S38] I. Reinhartz-Berger and D. Dori, “OPM vs. UML - Experimenting with comprehension and construction of web application models,” *Empir. Softw. Eng.*, vol. 10, no. 1, pp. 57–80, 2005.

[S39] J. E. Robbins and D. F. Redmiles, “Cognitive support, UML adherence, and XMI interchange in Argo/UML,” *Inf. Softw. Technol.*, vol. 42, no. 2, pp. 79–89, 2000.

[S40] M. C. Otero and J. J. Dolado, “Evaluation of the comprehension of the dynamic modeling in UML,” *Inf. Softw. Technol.*, vol. 46, no. 1, pp. 35–53, 2004.

[S41] O. Laitenberger, C. Atkinson, M. Schlich, and K. El Emam, “Experimental comparison of reading techniques for defect detection in UML design documents,” *J. Syst. Softw.*, vol. 53, no. 2, pp. 183–204, 2000.

[S42] A. Heuer, V. Stricker, C. J. Budnik, S. Konrad, K. Lauenroth, and K. Pohl, “Defining variability in activity diagrams and Petri nets,” *Sci. Comput. Program.*, vol. 78, no. 12, pp. 2414–2432, 2013.

[S43] M. Torchiano, G. Scanniello, F. Ricca, G. Reggio, and M. Leotta, “Do UML object diagrams affect design comprehensibility? Results from a family of four controlled experiments,” *J. Vis. Lang. Comput.*, vol. 41, pp. 10–21, 2017.

[S44] G. Scanniello and U. Erra, “Distributed modeling of use case diagrams with a method based on think-pair-square: Results from two controlled experiments,” *J. Vis. Lang. Comput.*, vol. 25, no. 4, pp. 494–517, 2014.

[S45] H. Byelas and A. Telea, “Towards realism in drawing areas of interest on architecture diagrams,” *J. Vis. Lang. Comput.*, vol. 20, no. 2, pp. 110–128, 2009.

[S46] M. El-Attar, “Evaluating and empirically improving the visual syntax of use case diagrams,” *J. Syst. Softw.*, vol. 156, pp. 136–163, 2019.

[S47] J. Cabot, R. Clarisó, and D. Riera, “On the verification of UML/OCL class diagrams using constraint programming,” *J. Syst. Softw.*, vol. 93, pp. 1–23, 2014.

[S48] M. Staron, L. Kuzniarz, and C. Wohlin, “Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments,” *J. Syst. Softw.*, vol. 79, no. 5, pp. 727–742, 2006.

[S49] M. C. Otero and J. J. Dolado, “An empirical comparison of the dynamic modeling in OML and UML,” *J. Syst. Softw.*, vol. 77, no. 2, pp. 91–102, 2005.

[S50] J. A. Cruz-Lemus, A. Maes, M. Genero, G. Poels, and M. Piattini, “The impact of structural complexity on the understandability of UML statechart diagrams,” *Inf. Sci. (Ny).*, vol. 180, no. 11, pp. 2209–2220, 2010.

[S51] L. Burgueño, T. Mayerhofer, M. Wimmer, and A. Vallecillo, “Specifying quantities in software models,” *Inf. Softw. Technol.*, vol. 113, no. January 2018, pp. 82–97, 2019.

[S52] R. George and P. Samuel, “Fixing class design inconsistencies using self regulating particle swarm optimization,” *Inf. Softw. Technol.*, vol. 99, pp. 81–92, 2018.

[S53] G. Sedrakyan, S. Poelmans, and M. Snoeck, “Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel UML statecharts by novice modellers,” *Inf. Softw. Technol.*, vol. 82, pp. 159–172, 2017.

[S54] F. Saleh and M. El-Attar, “A scientific evaluation of the misuse case diagrams visual syntax,” *Inf. Softw. Technol.*, vol. 66, pp. 73–96, 2015.

[S55] A. Blouin, N. Moha, B. Baudry, H. Sahraoui, and J. M. Jézéquel, “Assessing the use of slicing-based visualizing techniques on the understanding of large metamodels,” *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 124–142, 2015.

[S56] M. Felderer and A. Herrmann, “Manual test case derivation from UML activity diagrams and state machines: A controlled experiment,” *Inf. Softw. Technol.*, vol. 61, pp. 1–15, 2015.

[S57] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca, and A. Susi, “Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments,” *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1823–1843, 2013.

[S58] J. Al Dallal and L. C. Briand, “An object-oriented high-level design-based class cohesion metric,” *Inf. Softw. Technol.*, vol. 52, no. 12, pp. 1346–1361, 2010.

[S59] L. Muñoz, J. N. Mazón, and J. Trujillo, “A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses,” *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1188–1203, 2010.

[S60] H. Eichelberger and K. Schmid, “Guidelines on the aesthetic quality of UML class diagrams,” *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1686–1698, 2009.

[S61] A. Nugroho, “Level of detail in UML models and its impact on model comprehension: A controlled experiment,” *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1670–1685, 2009.

[S62] C. Glezer, M. Last, E. Nachmany, and P. Shoval, “Quality and comprehension of UML interaction diagrams-an experimental comparison,” *Inf. Softw. Technol.*, vol. 47, no. 10, pp. 675–692, 2005.

[S63] N. Bolloju and V. Sugumaran, “A knowledge-based object modeling advisor for developing quality object models,” *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2893–2906, 2012.

[S64] D. Aguilera, C. Gómez, and A. Olivé, “A complete set of guidelines for naming UML conceptual schema elements,” *Data Knowl. Eng.*, vol. 88, pp. 60–74, 2013.

[S65] A. Tort and A. Olivé, “An approach to testing conceptual schemas,” *Data Knowl. Eng.*, vol. 69, no. 6, pp. 598–618, 2010.

[S66] A. L. Opdahl and B. Henderson-Sellers, “A unified modelling language without referential redundancy,” *Data Knowl. Eng.*, vol. 55, no. 3, pp. 277–300, 2005.

[S67] G. Jošt, J. Huber, M. Heričko, and G. Polančič, “An empirical investigation of intuitive understandability of process diagrams,” *Comput. Stand. Interfaces*, vol. 48, pp. 90–111, 2016.

[S68] S. Wenzel, D. Poggenpohl, J. Jürjens, and M. Ochoa, “Specifying model changes with UMLchange to support security verification of potential evolution,” *Comput. Stand. Interfaces*, vol. 36, no. 4, pp. 776–791, 2014.

[S69] A. Haug, L. Hvam, and N. H. Mortensen, “A layout technique for class diagrams to be used in product configuration projects,” *Comput. Ind.*, vol. 61, no. 5, pp. 409–418, 2010.

[S70] M. Gogolla, F. Hilken, and K. H. Doan, “Achieving model quality through model validation, verification and exploration,” *Comput. Lang. Syst. Struct.*, vol. 54, pp. 474–511, 2018.

[S71] E. Planas, J. Cabot, and C. Gómez, “Lightweight and static verification of UML executable models,” *Comput. Lang. Syst. Struct.*, vol. 46, pp. 66–90, 2016.

[S72] D. Kchaou, N. Bouassida, and H. Ben-Abdallah, “Managing the Impact of UML Design Changes on Their Consistency and Quality,” *Arab. J. Sci. Eng.*, vol. 41, no. 8, pp. 2863–2881, 2016.

[S73] N. A. Zafar, “Formal Specification and Verification of Few Combined Fragments of UML Sequence Diagram,” *Arab. J. Sci. Eng.*, vol. 41, no. 8, pp. 2975–2986, 2016.

[S74] J. J. López-Fernández, E. Guerra, and J. De Lara, “Assessing the quality of meta-models,” *CEUR Workshop Proc.*, vol. 1235, no. 4, pp. 3–12, 2014.

[S75] M. Felderer and A. Herrmann, “Comprehensibility of system models during test design: a controlled experiment comparing UML activity diagrams and state machines,” *Softw. Qual. J.*, vol. 27, no. 1, pp. 125–147, 2019.

[S76] A. Ganser, H. Lichter, A. Roth, and B. Rumpe, “Staged model evolution and proactive quality guidance for model libraries,” *Softw. Qual. J.*, vol. 24, no. 3, pp. 675–708, 2016.

[S77] J. Choi, E. Jee, and D. H. Bae, “Timing consistency checking for UML/MARTE behavioral models,” *Softw. Qual. J.*, vol. 24, no. 3, pp. 835–876, 2016.

[S78] U. Mansoor, M. Kessentini, M. Wimmer, and K. Deb, “Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm,” *Softw. Qual. J.*, vol. 25, no. 2, pp. 473–501, 2017.

[S79] K. T. Phalp, J. Vincent, and K. Cox, “Assessing the quality of use case descriptions,” *Softw. Qual. J.*, vol. 15, no. 1, pp. 69–97, 2007.

[S80] Y. A. Khan and M. El-Attar, “Using model transformation to refactor use case models based on antipatterns,” *Inf. Syst. Front.*, vol. 18, no. 1, pp. 171–204, 2016.

[S81] N. Amálio, L. Briand, and P. Kelsen, “An experimental scrutiny of visual design modelling: VCL up against UML+OCL,” *Empir. Softw. Eng.*, vol. 25, no. 2, pp. 1205–1258, 2020.

[S82] I. Reinhartz-Berger and A. Sturm, *Comprehensibility of UML-based software product line specifications A controlled experiment*, vol. 19, no. 3. 2014.

[S83] A. De Lucia, C. Gravino, R. Oliveto, and G. Tortora, “An experimental comparison of ER and UML class diagrams for data modeling,” *Empir. Softw. Eng.*, vol. 15, no. 5, pp. 455–492, 2010.

[S84] G. C. Porras and Y. G. Guéhéneuc, “An empirical study on the efficiency of different design pattern representations in UML class diagrams,” *Empir. Softw. Eng.*, vol. 15, no. 5, pp. 493–522, 2010.

[S85] J. A. Cruz-Lemus, M. Genero, M. E. Manso, S. Morasca, and M. Piattini, “Assessing the understandability of UML statechart diagrams with composite states-A family of empirical studies,” *Empir. Softw. Eng.*, vol. 14, no. 6, pp. 685–719, 2009.

[S86] M. Genero, E. Manso, A. Visaggio, G. Canfora, and M. Piattini, *Building measure-based prediction models for UML class diagram maintainability*, vol. 12, no. 5. 2007.

[S87] C. Bunse, “Using patterns for the refinement and translationof UML models: A controlled experiment,” *Empir. Softw. Eng.*, vol. 11, no. 2, pp. 227–267, 2006.

[S88] B. Anda and D. I. K. Sjøberg, “Investigating the role of use cases in the construction of class diagrams,” *Empir. Softw. Eng.*, vol. 10, no. 3, pp. 285–309, 2005.

[S89] M. Balaban and A. Maraee, “Removing redundant multiplicity constraints in UML class models,” *Softw. Syst. Model.*, vol. 18, no. 4, pp. 2717–2751, 2019.

[S90] A. Chatzigeorgiou, “Mathematical Assessment of Object-Oriented Design Quality,” *IEEE Trans. Softw. Eng.*, vol. 29, no. 11, pp. 1050–1053, 2003.

[S91] T. Mens, A. Decan, and N. I. Spanoudakis, “A method for testing and validating executable statechart models,” *Softw. Syst. Model.*, vol. 18, no. 2, pp. 837–863, 2019.

[S92] D. Zayan, A. Sarkar, M. Antkiewicz, R. S. P. Maciel, and K. Czarnecki, “Example-driven modeling: on effects of using examples on structural model comprehension, what makes them useful, and how to create them,” *Softw. Syst. Model.*, vol. 18, no. 3, pp. 2213–2239, 2019.

[S93] H. Störrle, “On the impact of size to the understanding of UML diagrams,” *Softw. Syst. Model.*, vol. 17, no. 1, pp. 115–134, 2018.

[S94] M. Balaban, A. Maraee, A. Sturm, and P. Jelnov, “A pattern-based approach for improving model quality,” *Softw. Syst. Model.*, vol. 14, no. 4, pp. 1527–1555, 2015.

[S95] G. Bavota *et al.*, “A fine-grained analysis of the support provided by UML class diagrams and ER diagrams during data model maintenance,” *Softw. Syst. Model.*, vol. 14, no. 1, pp. 287–306, 2015.

[S96] S. Ali, T. Yue, and L. C. Briand, *Does aspect-oriented modeling help improve the readability of UML state machines?*, vol. 13, no. 3. 2014.

[S97] M. El-Attar and J. Miller, “Improving the quality of use case models using antipatterns,” *Softw. Syst. Model.*, vol. 9, no. 2, pp. 141–160, 2010.

[S98] D. Beimel and E. Kedmi-Shahar, “Improving the identification of functional system requirements when novice analysts create use case diagrams: the benefits of applying conceptual mental models,” *Requir. Eng.*, vol. 24, no. 4, pp. 483–502, 2019.

[S99] P. Bera and J. Evermann, “Guidelines for using UML association classes and their effect on domain understanding in requirements engineering,” *Requir. Eng.*, vol. 19, no. 1, pp. 63–80, 2014.

[S100] M. Dahan, P. Shoval, and A. Sturm, “Comparing the impact of the OO-DFD and the Use Case methods for modeling functional requirements on comprehension and quality of models: A controlled experiment,” *Requir. Eng.*, vol. 19, no. 1, pp. 27–43, 2014.

[S101] N. Bolloju and S. Alter, “Better use case diagrams by using work system snapshots,” *Int. J. Inf. Technol. Syst. Approach*, vol. 9, no. 2, pp. 1–22, 2016.

[S102] B. Baudry and Y. Le Traon, “Measuring design testability of a UML class diagram,” *Inf. Softw. Technol.*, vol. 47, no. 13, pp. 859–879, 2005.

[S103] X. Burgués, X. Franch, and J. M. Ribó, “Improving the accuracy of UML metamodel extensions by introducing induced associations,” *Softw. Syst. Model.*, vol. 7, no. 3, pp. 361–379, 2008.

APPENDIX B

Definition and observations for the factors effective on the quality and understandability of UML diagrams

Factor	Description and Observations	Study
Effect of Architecture-level dependability evaluation techniques:	Dependability is of great importance for critical systems, which can be also evaluated at model level to ensure reliability and availability on software systems. *One of the proposed method for architecture level dependability evaluation of UML-DAM models is based on hierarchical and colored extension of stochastic activity networks. The obtained results demonstrated that improving the dependability of software/hardware components has a positive influence on the dependability of the whole software system.*	[S1]
level of normality/precision in workflow modeling, based on UML activity diagrams, influences two aspects of construct comprehensibility: correctness of understanding and task completion time	Workflow modelling can be realized in two styles, namely, precise style (with specific rules and enforced constraints) and ultra-light style (no rule and no constraint). *Based on the ultra-light style the activity diagrams are produced without obey rules nor constraints, while for the precise style all possible rules and constraints are exerted.* *The precise style useful in comprehending workflows and it can influence two aspects of artifacts, comprehensibility. The findings show that modeler and developers gain a better comprehension when precise style is being used, while the precision style does not have significant effect on the task completion time.*	[S2]
Domain Modeling Techniques:	Domain models is defined as the model visualizing problem domain objects along with the associations and attributes, encompassing the vocabulary and key concepts of the problem, which is creating using category list technique or noun phrasing technique. *The obtained results show that category list technique is better than noun phrasing technique for the identification of concepts as it avoids generating unnecessary elements i.e. extra concepts, associations and attributes in the domain model. The noun phrasing technique produces a comprehensive domain model and requires less effort as compared to category list. There is no statistically significant difference between both techniques in case of correctness.*	[S3]
size and complexity of the software	As the complexity and size of the software systems are rapidly growing, the use case diagra drawn to capture the requirements is becoming more and more complicated. The understand ability and read ability of the UCD is also affected. A method based on Page Rank and HITS algorithms are used to identify the importance use cases. This approach provides information about the key requirements i.e. which part of the initially unknown problem domain should be looked at first by identifying the key use cases in UCD.	[S4]
Refactoring method	Software refactoring is the process of changing a software system in a manner that does not alter its external behavior and yet improving its internal structure, which can be also used at modelling level. *A multi-view integrated approach to model-driven refactoring using UML models is proposed. The results indicate that more bad smell instances can be detected using the integrated approach rather than the individual refactoring approach.*	[S5]
representation of roles in conceptual models	When describing an Organization or part of it, a key concept is the notion of roles played by actors in the domain. Actors in an Organization act in various roles, hence, showing that roles in a conceptual model can promote understanding of how the Organization works. *The role can be declared as a conceptual modelling construct and a set of guidelines for representing roles in conceptual models can be defined to suggest rules for modeling roles with the widely used extended entity-relationship grammar. The results indicate that the role-based rules are not only useful for understanding the models but also provide direct clues as to why this is so.*	[S6]
use of iconic signs as UML graphical notations	It is only through the process of ‘‘sense making’’ that the relationship between signs and its representation can be established. There are three modes of signs: Iconic, Symbolic and indexical. Iconic sign is defined a mode in which the signifier is perceived as resembling or imitating the signified (recognizable look, sound, feel, taste or smell) or being similar possessing some of its qualities: e.g., a portrait, a cartoon, etc. *We hypothesized that the use of iconic signs as UML graphical notations leads to representation that is more accurately interpreted and that arouses fewer connotations than the use of symbolic signs. The results support our propositions that iconic UML graphical notations are more accurately interpreted by subjects and that the number of connotations is lower for iconic UML graphical notations than for symbolic UML graphical notations.*	[S7]
Proposed a new graphical and textual notation to represent non-functional requirements	In the available notation the representation of functional requirements is feasible while no interactive requirements representation are not taken into account. The results of the evaluation indicate that the representation of operation cases helps in documenting a complete set of functional requirements, which ultimately results in a comprehensive translation of requirements into design. The findings show that the representation of non-functional requirements make a significant contribution in terms of translation of the requirements into the design.	[S8]
Effect of ontological-based verification	The promising solution for hindering the models errors shift into the code is model verification. In *The ontology-based verification method a class diagram is transformed into ontology declared in web ontology*	[S9]

method on UML Class/OCL model:	*language). It is worth noting that ontology has many elements, such as class, relations and individuals like UML class.* *This method is able to efficient caver all aspect of UML class model verification.*	
Quality Assessment method	In nowadays industry of software, security is only taken into account at implementations stage, which hinders the effective communication of the security solutions to the stakeholders. System security is an integral aspect of any system, which should be also check at early stage of software development. However, modelling language notation and validation support is required. *A systematic method for apprising the quality of security models, UMLsec is proposed and the findings demonstrated the efficiency of the proposed method in developing high quality security models in terms of correctness, higher semantic completeness, easier modifiability, understandability and easier to communicate between stakeholders.*	[S10]
a new method for developing comprehensive OCL postconditions for operations in UML and OCL models	To describe behavior in UML and OCL models can be done through using OCL pre and post-conditions. Indeed, the functionality of an operation is described in a declarative way via use of pre and post-conditions. *The proposed method presents PCDL elements which are based on new formal distinction and the findings showed that the proposed method simplify the specification and representation of the post-conditions.*	[S11]
Learning difficulties	some technical properties of UML diagrammatic representation, coupled with students' cognitive attributes, hinder both perceptual and conceptual processes involved in searching, recognizing, and inferring visual information, which creates learning barriers. *Study demonstrates the usefulness of participatory cognitive approach in developing the shared understanding between instructors and students on difficult course materials. The concept-mapping technique enables one to meaningfully and efficiently assess and represent learning challenges, and other IS courses can use it to enhance teaching and facilitate learning.*	[S12]
Construct Redundancy & Modelling Knowledge of the reader	It should be noted the conceptual models play a key role in comprehending the domain models related to the information system, while in some cases the existence of constructed redundancy induced by the use of conceptual modelling grammar may lead to reduction in the clarity of the conceptual models. *The findings show that a model with a weak level of redundancy are more understandable for the readers, while the trained readers are not affected by the weak level of redundancy in a model.*	[S13]
CHOP method effect on improving novice analyst performance	CHOP technique is defined as chunking, ordering and patterning, which improved novice analyst performance in modeling sequence diagram and it is based on cognitive load theory. *The obtained results demonstrated that indicated that the CHOP technique significantly enhances novice analyst's ability to model interactions among objects;*	[S14]
Learning difficulties	complexity of learning sequence diagram, due to the Unfamiliar Metaphors, Working with SD Syntax (SDs have a detailed and very specific syntax), omitting Explicit Returns (The notion of flow of control through the logic is not evident in an experienced analyst's SDs, since explicit return notations are often omitted to reduce clutter and fit more logic into a given space), Decomposing high-Level Requirements into Logic Steps, Choosing to Centralize or Decentralize Responsibility Assignments, Rate of Information Presentation, Designing for quality. *developed a set of recommendations aimed at reducing the difficulty associated with the task of learning to build SDs by applying concepts from the theory of LCD. We believe these recommendations will be of interest to educators, experienced analysts mentoring novices, and others.*	[S15]
Enhancing UML models	The correctness of the models considering both internal consistency rules and system requirements and domain constraints is of a great importance. *The findings support the usefulness of domain model in realization of more correct and complete models.*	[S16]
Representation of Business roles from early stage	Any change in business policy may deeply influence the information system evolviablity and efficiency. So, business rules should be represented in the models from the early stages. *By business rules modelling, there will be an extension in the state of the art approaches for modelling the business rule and there will be a reduction in the redundancy rate and hindering the conflict between the business rule types.*	[S17]
Refactoring:	Use case diagrams are used during the early stage of UML-based development approach, which capture the user requirements. *It is feasible to refactor the use diagrams based on Formal Concept Analysis and Relational Concepts Analysis and the observations are promising regarding that the precision of the use case diagrams increases*	[S18]
Effect of good decomposition model (GDM)	Decomposition is defined as a top down process of breaking down a complex system into smaller and relatively independent units *The results verified that GDM is useful measure in evaluating the understandability*	[S19]
Reading Technique	For evaluating the quality of the models, defect detection is a proper method which is applied using reading techniques, while there are some limitations and constraints for the reading techniques and it signify the importance of the need for new reading technique. *The results indicate that the Functional Size Measurement (FSM) is useful since it finds all the defects related to a specific defect type, it finds different defect types than an inspection group, and it finds defects related to the correctness and the consistency of the models.*	[S20]
Testing the model in terms of semantic correctness	The semantic correctness of a conceptual schema can be seen from two different perspectives. On the one hand, from the point of view of its definition, a conceptual schema must be right (verification techniques) that check whether the schema satisfies several correctness properties. On the other hand, from the point of view	[S21]

	of the requirements that the information system should satisfy, a schema must also be the right one (validation techniques) which help the designer understand the exact meaning of a schema and to see whether it corresponds to the requirements. *In this article an approach to verify and validate UML conceptual schemas, with arbitrary constraints formalized in OCL was proposed. This approach allows the verification and validation of very expressive schemas, ensuring that when a property is satisfied by a schema, an example proving so will always be obtained, as long as the schema satisfies the conditions guaranteeing termination.*	
Aspect-oriented design	Aspect-orientation provides a new way of modularization by clearly separating crosscutting concerns from non-crosscutting ones *a conceptual reference model is presented as the article's first contribution, centrally capturing the basic design concepts of AOM and their interrelationships in terms of a UML class diagram. From the results obtained in the evaluation, it is not possible to nominate a winner. The selection of an AOM approach thus has to be made in the context of a specific project. In principle, the approaches allow the modeler to consider both structure and behavior through their approach. Nevertheless, currently no approach addresses the full spectrum of UML in terms of UML's structural and behavioral diagrams as well as their composition*	[S22]
Rules and Guidelines in terms of abstraction rules	The process of abstraction is the simplification of models by removing details not necessary on a higher, more abstract level. *Our abstraction technique uses abstraction rules that have input and result patterns. Abstraction rules define the semantics of how a set of model elements can be replaced by a less complex, more-abstract model element. We showed that our technique scales, produces correct results most of the time, and addresses issues such as model ambiguities that are inherently part of many (UML) diagrams. We demonstrated various forms of ambiguities and showed that there are ways of living with them—even preserving them during transformation.*	[S23]
Refactoring	In the ¯field of software engineering, the term class invariants is known as a valuable term employed to delineate the semantic of UML class diagram elements (attributes and relationships) and must be held throughout the life-time of instances of the class. Refactoring, the activities of re-distributing classes, attributes and methods across the class hierarchy, is a powerful technique that is used to improve the quality of software systems. Performing refactoring on UML class diagrams obviously requires a special investigation of invariant-preserving on the refactored models. *An approach to preserve class invariants in refactoring UML models was proposed which demonstrates the excellence of the proposed approach when combining with other maintain activities to improve the quality of the software model.*	[S24]
Metric	Effect of early quality evaluation or quality model The metrics evaluate the object-oriented concepts: methods, classes, coupling, and inheritance. The metrics focus on internal object structure that reflects the complexity of that entity and also on external complexity that measures the interaction among entities. *This paper proposes a quality model to assess the design phase of any object-oriented system based on the works of Chidamber, Kemrer and Basili and suggests two new metrics. The design phase model that we proposed has generated quality indication to real-time project practiced in the industries. The result supports our concept and it is consistently proving its quality.*	[S25]
Effect of Metrics visualization	Metrics are consisted as tools to appraise the structure, complexity and quality. Metrics visualization can boost the developers' intuitive perception of the metrics. *A new approach for 3D visualization of the metrics used in the design phase of object-oriented software is presented, which extracts the eight key metrics using class diagrams.*	[S26]
Testing the domain models in early stage	It is essential to be assured about quality of the early models in mode-driven engineering. The quality of the models can be assessed in terms of synaptic and semantic, which are so called verification and validation, respectively. *The scholars have proposed a methodology and a tool the deduce a set of customizable inquires in natural languages from each model, and answering these questions will aid the experts the defective sections of the models, and ensuring the quality of the domain models in early stage. The obtained results proved the effectiveness of the proposed tool, MOTHIA in detecting the modeling errors Also, the proposed methodology narrow the gap between domain experts and modeling experts, which is the main source of early stage modelling errors.*	[S27]
Extending the UML Statecharts Notation to Model Security Aspects	Modelling the security aspect of the system has gained interest among the scholars, while the UML statecharts notation can be also extended to model the security aspect. *The obtained results demonstrated that the new notation for UML statecharts is more understandable in terms of reading the models.*	[S28]
Customization of the process models representation to resolve lack of fit	Process models are employed during analysis and design of the system. However, there is need for representations customization in order to resolve the lack of fit between these models and in intended task. *The findings ascertain that due to the impediments some customization in the process models representation is required and such customization can be done from two perspectives of physical and process customizations.*	[S29]
High ability Experience Dynamic Models	Dynamic models are used for modelling the dynamic and functional behavior of the users and software systems. *The findings proved positive effect of using theory-guided models on outcome and speed of acquiring the models during the initial stages of training,*	[S30]
Conallen's notation & Developers' experience and ability	Stereotypes are UML modeling entities for which a specific semantics is defined by the user. Stereotyped UML elements can be represented with a user-defined, intuitive graphical notation (i.e., icons), that replaces the standard stereotype specification, and in this study the scholar have used the Conallen's WAE notation stereotypes which are for diagram elements of web application modelling *According to the obtained results, it was observed that there was no significant different in employing diagrams stereotyped based on Conallen's notation, but such stereotypes mitigates the gap between the participants in*	[S31]

	terms of their ability and experience, in such a way that using stereotyped UML diagrams in organizations consisted of low experience developers will exhibit high performance.	
Propositional Logic Constraint patterns	Proposition logic constraints: constraints are defined as limitations on the states of the data, as well as the objects owning them, which can be represented by a set of authentic values or through relationship configurations [36]. It should be noted that the constraint pattern refer to the repeating structures specifying certain categories of k-variable propositions observed on domain of discourse during conceptual modelling and these constraints denote propositional statements easily added to ER or UML class diagram. A set of invariant patterns are suggested fir capturing a rich category of propositional constraints, and the results demonstrated that the approach enriches UML-based conceptual modellings for greater completeness, consistence, and correctness via formalized syntax and semantics of these constraint patterns	[S32]
Difficulties in transition from object-oriented analysis (OOA) to logical design (OOLD)	The transition from object-oriented analysis to logical design exerts a series of difficulties on the learners and software engineers in terms of semantic distance. *The findings ascertain that due to two factors the learners and engineers face with difficulties during transition from object-oriented analysis to logical design, namely, semantic distance and the precise degree of transition relations.*	[S33]
Example	Example: These examples are offered via a searchable repository. *The results showed that the designs which were fabricated based on examples were 18% better that the designs which were not produced based on example.*	[S34]
Part-Whole Relationship	There should be different between part of relations and ordinary associations-There is a little agreement on the precise meaning of the part-of relationship. ……	[S35]
Requirements Reusability	It has been always challenge to develop models which are able to store the requirements promoting the easy requirements reuse, while at the same time be understandable by software engineers. *An ontology-based approach provide the feasibility of reasoning and also to detect the missing or incomplete requirements, which reduce the efforts need for requirements elicitation at early stage of software development life cycle.*	[S36]
The effect of theory-guided on learning	The effect of theory-guided on learning: For enhancing the learning of the conceptual models several theories have been proposed with main focus on guiding for structuring the conceptual models, such as theory of ontological expressiveness [32], unified foundational ontology [33] and physics of notations [34]. *Experiment shows positive effects on both outcome and speed of learning in the initial stages of learning, the real contribution of our paper is methodological, i.e. an eye-tracking procedure to observe the process of learning from conceptual models.*	[S37]
Effect of modelling language	Effect of modelling language UML VS OPM: Web applications required a modelling method which should be capable of clearly specifying the aspects of applications architecture, communication and distributive nature, and there two possible choices, namely UML and Object-process methodology OPM *The results show that OPM excel the UML notation in modelling the web applications in terms of dynamic behavior, while there is no meaningful difference between UML and OPM in terms of specifying the structure and distribution. Also, students create more quality OMP models compared to UML models.*	[S38]
Cognitive support in software design tools	Most of the software design tools provide the basic `functionalities for the engineers, such as editing, viewing, storing and transforming designs, but lack support for cognitive tasks like decision marking, decision ordering and task-specific design understanding. *Some extent of cognitive needs of the software engineer and designers are implemented by Argo/UML.*	[S39]
1) Diagram Type 2) Application domain 3) Order of presentation of the documents 4) Paired combination of dynamic diagrams	Herein, diagram type including sequence, collaboration and state diagrams of UML, while application domain refers to the business domain for which a software is being developed. The presentation order refers to the delivery order of documents to the study subjects. *It was observed that state diagrams are capable of presenting higher semantic understanding for dynamic aspect the system when the domain is real-time system, while sequence diagrams exhibited better result for management information system. Also, it was showed that greater level of semantic comprehension can be realized using the pair sequence-state for modelling the dynamic aspect of the system*	[S40]
Reading Technique	The act of inspecting a software is consisted of several activities namely, planning, defect detection, defect collection and defect correction [39]. Here, for defect detecting two known method including Perspective-based reading (PBR) which provide the ground for controlling a document from a specific perspective of the stakeholders [24] and Checklist-based reading (CBR), which provide no clear advice for inspectors in terms how to accomplish the process and to focus specify on which aspect of the document during the reading activity ([25], [26] and [39]) are introduced as the effective factors in quality of UML design documents. *The results showed that PBR is more efficient than CBR in error detection and error detecting based on PCR is more cost-effective for UML design document. PBR is also more reliable in enhancing the UML models quality.*	[S41]
Variablity Support in Activity diagrams	In product line engineering, quality assurance is challenging since any defect in the domain artifact can have an impact on several products line, so the proper quality assurance method should pay attention to the product line variability, and since activity diagrams are used testing in product line engineering, so the variability of the activity diagrams is also important. *Through defining both syntax and semantic variability in activity diagrams it is possible to support the models and compared to the control flow models, activity diagrams enable the modelling of parallel control follows*	[S42]
The effect of object diagrams on comprehensibility	Object diagrams present a complete or partial view for the software system structure during run-time. Class diagram along with objects diagrams can be used to record the concepts of problem domain of a software and to specify the relationship among these concepts ([21], [22]).	[S43]

	The results suggest that the use of object diagram does not always make a significant contribution to the design comprehensibility. The benefits are strongly depended on the experience and familiarity with UML.	
participants' familiarity	The familiarity refers to the developers experience in modeling and their knowledge of the application domain. *The results demonstrated that the developers' familiarity have meaningful effect on task completion time, while there was no significant effect on models quality.*	[S44]
effect of computer-drawn and hand-drawn Areas of interest (AOIs	Areas of interest (AOIs) are defined as groups of components of system architecture diagrams with common property, whose visualization can be done automatically and drawn by human. It is not clear whether UML diagrams equipped by AOIs are enhanced in terms of quality. *According to the combined user evaluation, it was observed that AOI enhances both quality and understandability of architecture diagrams.*	[S45]
The effect of Visual Syntax of Use Case	Use case modeling is known as one of preliminary artifacts in capturing system functional requirements . In fact, the model reader gain insight into the mental idea of any model via visual perception, and if it is misread or misinterpreted then the main objective of the modelling has been unsuccessful. According to the findings, the cognitive effectiveness of use case representation can be enhanced by making some enhancements.	[S46]
absence of constraint redundancies	In the verification process, the models should be check in terms of correctness, from perspective of not violating the integrity constraint and avoidance from unnecessary constraints in the models, so called constraint redundancies. *It was observed that considering the diagram structure and constraints it encompasses, it is possible to analyze large class diagrams efficiently using UML to CSP.*	[S47]
stereotypes	Stereotypes are used for customizing the modelling language for its application in specific domains and their main role is to brand an existing models components with specific semantics, as well as supplying a secondary classification of modelling components. *The obtained results refer to the meaningful role of stereotypes in boosting the comprehension of models.*	[S48]
Effect of modelling notation/language	Effect of modelling language UML vs. OML. UML stands for unified modelling language while OML stands for open modelling language. *The results showed that OML language is more efficient in modelling the dynamic behavior of a system, in terms of comprehension and the generated artifacts are easier to interpret, compared to UML language.*	[S49]
Structural complexity:	Structural complexity and its effect should be studied in terms of three dimensions, namely the size and control flow complexity, the actions related to states and the sequence of actions. *According to the results it was observed that each of complexity dimensions have a meaningful effect on understandability of a state chart diagram*	[S50]
Representing physical quantities in software models	In developing a software dealing with physical entities, like automotive or aerospace domains, it is highly required to consider the formal representations of the uncertainty and units both in design and implementation phases. *The findings show that the physical quantities representations can safely incorporated in the software models, which enable the modelers to safely represent and manipulate units and measurements uncertainties of physical systems.*	[S51]
Class Attribute Inconsistencies	Detection of intra-model design inconsistences is f great importance for quality software, as ensuring consistency and completeness of models is an undeniable software design. *Detecting and resolving the inconsistencies via adoption of artificial intelligence leads to completeness of the design and development of quality software*	[S52]
Feedback-inclusive prototype	Feedback-inclusive rapid prototyping method (FIRP) is a simulation means used for evaluation of a model with respect to the design software systems as defined in the requirements. *Findings show that using a prototyping technique have a significant impact on comprehension of system dynamics presented in the form of multiple interacting state charts, which excel the manual inspection of interacting state charts.*	[S53]
The use of color and icons	Most of the software engineering notations are used on Black and white color, while the visual communication aspect of modelling in terms of cognitive effectiveness in software engineering is of great importance, which can be improved to make the models easier and quicker to be read. *The finings show that the use of color and icons improve readability of misuse case diagrams*	[S54]
Models slicing:	Model slicing is a method for model comprehension which originate from program slicing. In some cases, modelers may be interested in a particular aspect f meta models, so they may want to hide the rest of the meta models. *The findings show that model slicing based on interactive navigation features is an effective way in enhancing the met model comprehensibility.*	[S55]
UML-based Test Generation	The objective of the study presented in this paper therefore is to examine which errors are possible and actually made when manually deriving test cases from UML activity diagrams or state machines and whether there are differences between these diagram types. *The authors aimed at identifying the errors types that are formed during manually test derivation using UML activity diagrams or state machines.* *The found that activity diagrams are more error prone, while have a higher perceived comprehensibility.*	[S56]
Modelling Languages	There are generally two modelling approaches for requirements, namely Use Case; scenario based and Tropos, Goal-based. *According to the obtained results, Tropos exhibited more comprehensibility, in terms of three types of requirements analysis tasks, while using more time.*	[S57]
Class Cohesion	Class cohesion is a significant attribute ion assessing the objected-oriented software quality, denotes the relatedness of the class members, and there are three alternatives: method-method, method-attribute, attribute-attribute interactions. *Using new cohesion metric called SCC metric, it is possible to address a number of issues arising from existing metrics and it allows early assessment of designs in terms of cohesion.*	[S58]

Structural complexity of ETL (Extract, Transform and Load) Process Models	The conceptual models of Extract, Transform and Load process are software designs, whose structural complexity may have an influence on its cognitive effectiveness[37] and high cognitive complexity mitigates the understandability and modifiability of the models. *According to the correlation results, it can be stated that there is a correlation between the understandability and measures, like NFES which is a complexity measure and it is defined as the number of input and output flows un the ETL process model.*	[S59]
visual guidelines on aesthetic quality of the diagrams	Due to the increase in success of visual specification notation, like UML, visual guidelines are required to standardize the presentation of modelling diagrams in terms of human communication, perception and readability. *A new taxonomy of guidelines was introduced based on diagraming rules from literature and semantic layout rules for UML class diagrams. Findings showed that the a subset of rules did not have a significant impact on the models understandability (may be due to the low number of the study participants.)*	[S60]
Level of Detail (LoD) in UML models as a form of style and rigor in UML modeling	Level of Detail (LoD) refers to the amount of information that can be employed to show the modelling element, like representing a message of sequence diagram using an informal labels or label that shows the method, or even a labels that is method name plus argument list. *The findings refer to the significant impact of level of detail on comprehension of the UML models.*	[S61]
Effect of Diagram type (Sequence and Collaboration) and Application domain	Interaction diagram of UML are presented in two forms, namely sequence diagram, depicting an explicit sequence of stimuli messages communicated between objects instance, while collaboration diagrams is a direct graph, and the nodes denotes the communicating entities and edges are used for communications. *The results showed that collaboration diagrams can be easily perceived for RT system, compared to sequence diagram, while in management information system, there is no significant different in comprehension of these two diagram types. Also, the comprehension of interaction diagram for MIs is easier than RT system. Moreover, for MIS system more quality collaboration diagram cab be created, compared to sequence e diagram.*	[S62]
knowledge-based object modeling advisor	The lack of mutual understanding of the domain-specific knowledge between system analyst and requirement engineer, as a result miscommunication between the users and technical personnel should be noted as the two common problem of early phases of system development. *The obtained results prove the effectiveness of knowledge-based object modelling advisor system, which is added as an extension to an open source CASE tool, particularly in terms of advices and suggestions for model completeness aspect in terms of semantic quality.*	[S63]
Naming for conceptual elements	Naming the conceptual schema elements is a challenging problem as for any new element, a new name is required, the impact of name on comprehension of the schema is not ignorable. For naming the element in conceptual models a guideline can be sued. The name which is opted in accordance with the guideline, if it possess the form and the sentence generated from the sentence pattern, is grammatically well-formed and semantically meaningful.	[S64]
Testing of *conceptual schema*	Testing is defined as an attempt to try to make the program to fail, to uncover the faults by triggering failures [40], as most of the proceeding studies assume that the conceptual schemas are executive so they can also be tested. *The proposed method for testing the conceptual schema is based on software testing. It was shown that testing the conceptual model should be regarded as the tools for their validation, and the schemas should be tested with the goal of uncover faults by triggering failures.*	[S65]
referential redundancy modelling constructs:	Referential redundancy occurs in a class is represented by the several constructs or models elements, thing or properties of the problem domain. The referential redundancy reduces the models quality, *The referential redundancy can be eliminated at the language level and potential reduced at the model level. UML language can be reformulated by breaking down a modelling construct into the smallest parts.*	[S66]
Effect of process modelling notations in terms of intuitive complexity	Process modelling languages encompass a precise syntax, semantics and visual process modeling notation. UML Activity Diagram: It is noted as de facto standard for software industry modelling [18]. Business Process Model and Notation: It was initial published in 2004, with a main of to provide notation comprehensible by the business users [18]. Event Driven Process Chain: It was developed in 1992, at the Saarbrucken University, the basic idea of EPC is defined in such a way that the events trigger the functions and execute functions triggering events, generating a chain of functions and event [19]. *The findings show that for the process with lesser complexity, BPMN diagrams are more suitable. For process with high complexity, EPC performance was wore, compared to UML AD and BPMN. UML AD can be considered as the most versatile modelling language*.	[S67]
Testng the model in terms of Consistency of security requirements during models evolution	Consistency of security requirements: Using UMLsec models it is possible to be ensured that the security vulnerabilities are identified at an early stage of the software development, which can be verified in terms of consistency using UMLsec stereotypes. *Using UMLsec stereotypes leads to security check of the model evolution, and it has a considerable efficiency that simple re-verification of the entire model.*	[S68]
Effect of layout technique	Using the layout technique Vertically Aligned Class diagrams by incorporating the usability of diagramming technique into the class diagrams produce analysis models which are more understandable for domain experts. *The obtained results showed that the use of VACDs leads to an increase in efficiency, improvement in communication and reduction in the errors.*	[S69]
Testing the models in terms of eight use cases	Validation and verification are defined as "are we building the right product? "and "are we building the product right", respectively [36]). Eight validation and verification use cases play role on quality of the class diagram, namely, model consistency, property satisfiability, constraint implication, constraint independence, solution interval exploration, partial solution completion, equivalence implication and partitioning with classifying terms. *The results show the usefulness of the 8 use cases of validation and verification.*	[S70]
Testing the models using lightweight and static verfication	The lightweight and static verification method can be used to evaluate the correctness of the executable models, since the correctness of these models is highly importance due to their ability in facilitating the semi-automatic implementation of software system. The lightweight method perform static analysis of the model, and it examines the model without executing its operations.	[S71]

	This method supply a repairing feedback to aid the designers for improving the model.	
Impact of UML Design Changes	The changes in various modeling phase are able to exert modifications on software models, implying model quality deterioration, like incoherence among models, increased complexity. Indeed, any change in one model should be propagated coherently to all related models horizontally and vertically. *For resolving the effects of change, an automated approach available which analyze the changes effect on design models and taken into account the impacts on models quality based on three techniques, graph-based, rule-based and metric-based. This method showed 88.33% precision and recalled of 95%*	[S72]
Testing the models considering The role of models verification	Proofing the correctness of the software models has a crucial role on ensuring the quality of the final product *For model verification, a systematic method of transformation and verification UML sequence diagram into Z specification can be used to facilitate practitioners for verifying the models based on UML sequence diagram.*	[S73]
Effect of quality assessment approach	AS the meta-model-based modelling languages change continuously, there is a need for quality assurance approach for assessing the quality of meta models, as The quality assessment approaches have an impact on quality of meta models. *An approach based on the quality models can be used to measure the quality of meta models, which was able to measure the syntax quality, semantic quality, pragmatic quality, evolvability and capability*	[S74]
Effect of diagram type (UML activity and state machines) on comprehensibility in test case derivation	For modelling the behavior of the system from perspective of the user there are two alternatives in UML modelling language, namely activity diagrams and state machines. Testing the system based on the available design artifacts is a variant of system testing which is bonded on explicit models. *The findings showed that activity diagrams excel state machines in term of understandability, while these diagrams are more error-prone in derivation of the manual test cases.*	[S75]
Model evolution approach	The present model evolution do not take into account the reuse aspect of the models sufficiently. A new approach can be presented with three stages, and each stage is defined by four important quality dimensions, namely syntactic, semantic, pragmatic and emotional. *The afore-mentioned approach is precise, and sufficiently automated for software implementation, which facilities the way for evolving and discussing models to be accepted by the models.*	[S76]
Quality Assessment Approach approaches (in terms of timing inconsistencies)	Timing inconsistencies and Quality assessment approach in terms of timing inconsistencies may affect the quality of the models for real-time embedded software. UML/MARTE is an approach used for modelling real-time embedded software. UML/MARTE inconsistencies are defined as the issues related to the modelling faults which may be hazard during the life cycle of RTES. *A systematic approach based on rule can be used for time consistency checking and the obtained results proved the effectiveness of this approach for cruise control system software and guidance and control unit software.*	[S77]
Effect of multi-view refactoring approach on models quality	Refactoring is one of the basic and essential method for enhancing the quality of the artifacts, through changing the internal structure without any transformation in the external behavior. *For multi-view refactoring, a multi-objective evolutionary algorithm can be suggested to detect a trade-off between enhancement in quality of class and activity diagrams. The statistical evaluation results for four open-source systems approved the efficient of the proposed method.*	[S78]
Effect of use case description quality check list on model quality:	The use case model is a tool for depicting the system specification in a natural and understandable way with a focus on the system from user perspective [28] The use case description checklist can be useful in checking use case guidelines, for testing desired quality features in the description and the quality features themselves are employed for examining and revising the use case descriptions. *Either through a direct application or using the guidelines as the use case guidelines, the quality of the use case description will be a good measure to assess and improve the quality of the use cases.*	[S79]
Effect of refactoring the anti-patterns:	The process of refactoring is defined as the means of improving the structure of software artifact free from any change in the behavior. Anti-patterns are defined as the low quality solutions for re-occurring design problems. *For refactoring model transformation which detect anti-patterns instances in a given use case model can be used. The results demonstrated that model transformation is capable of enhancing the quality of the use case models in a time and cost-effective manner.*	[S80]
the effects of VCL on visual modelling quality	The visual contract language (VCL) is devised for enhancing the visual modeling by boosting the diagrammatic cognitive effectiveness, increase the visual expressivity and level of seriousness and formality. *The findings shows that VCL is useful for defect identification.*	[S81]
Explicit reuse guidance and/or variability specification on domain models comprehensibility	*The findings showed that the explicitly specification of reuse guidance for domain model contribute the understandability of model, while for the variability increase enhanced the comprehensibility only to a limited degrees.*	[S82]
Diagram type in data modelling	For data modelling there are two well-known graphical notations, namely Entity-relationship and UML class diagrams. *The results showed that UML class diagrams possess a better comprehension. Also, UML class diagrams provide a good support for verification activities, compared to ER diagrams.*	[S83]
stereotype-enhanced UML diagrams effect on understandability:	Generally, for design pattern representations the collaboration diagrams are used. Dong et al. [27] have proposed a UML diagrams which are equipped with stereotypes, the tagged values and constraints for visualizing the pattern-related information in UML-class diagrams. The tagged values contain information about the role, that class, attributed or a method have in a design pattern. *According to the data it is observed that UML diagrams which are enhanced by stereotype are more useful in identifying the composition and role, than UML collaboration diagram. For locating the class*	[S84]

Effect of composite states:	The composite states provide the modelers with ability to construct UML state charts in a hierarchical fashion. *According to the findings, it is observed that the there are some about the usefulness of composite states for better understandability of UML state chart diagrams.*	[S85]
A collection of Uml class diagram measures	In order to make an early evaluation of the understandability and maintainability, it is important to establish a set of indicators, which need to be based on the features of the early artifacts [23]. Eight measures for assessing the structural complexity of UML class designs can be considered. *According to the findings, the predictions models which may be useful for class diagrams understandability and modifiability can be set up based on early measures, which encompass structural complexity.*	[S86]
Effect of Sort:	SORT is a systematic approach for refining and translating UML diagrams ([29], [30] and [31]) which is used for resolving the issues related to the supporting the mapping of UML artifact into executable code *The observations ascertain that SORT leads to generation of more quality models in terms of understandability and verifiable.*	[S87]
Effect of transition technique on quality of class diagram:	It should be noted that there is no established method for transition from use cases to class diagrams, there are generally two methods derivation (class diagram from use cases) and validation method (class diagram created independently and use case is used for validation) *The findings demonstrated using the use cases for validating the class diagrams which are based on textual requirements more proper class diagrams. So, the technique used for transition from use cases to class diagrams may influence the quality of class diagrams, and the effect of transition technique is depended on the category of the developers.*	[S88]
Effect of redundant multiplicity constraints on quality	The UML class diagrams are capable of representing complex constraints on its components while the interaction among them leads to mitigation in correctness. Redundancy refers to the existence of excessive unnecessary information. An algorithmic rule-based methods are used for resolving the redundancies in the multiplicity constraints	[S89]
The application of the HITS algorithm in object-oriented designs in order to evaluate the quality	Generally, metrics are tools for assessing and quantifying several aspects of the software engineering process and product, which are mainly focused on object orientation or measure specific characteristics of design. *Hyperlink Induced Topic Search algorithm can be employed for assessing the quality of a model, like class diagram. According to this method, findings show that the role of each class in a class diagram is bonded to not only the number of the incoming and outgoing messages, but also on the importance of the class. Using HITS algorithm in evaluation of a class diagram, it is feasible to obtained authority and hub weights for each class which records the combined effect of communicating with other classes. Moreover, these weights can be used for adopting the adjacency matrix of the direct graph.*	[S90]
Effect of testing and validation of State chart models	The complexity of state chart models signify the importance of advanced model testing and validation to increase in the quality and reliability of behavioral software models, since the designing phase of the statecharts can be error-prone. *The proposed method for testing and validation the statecharts exhibited good performance in upgrading the traditional state chart designs*	[S91]
Effect of using examples on structural model comprehension	A example-driven modelling is considered as a method which adopts examples for model comprehension and domain knowledge transfer in a systematic manner. *The findings show that the developers provided with abstractions which are augment with samples exhibited higher perception of the diagrams and in their tasks, and they asked %90 fewer clarification domain specific questions.*	[S92]
Impact of size on understandability of UML diagrams:	*It should be noted that there is no general consensus over the exact concept of diagram size, which necessitate the need for operationalizing this notion and analyze its impact on models understandability. It is clear that with increase diagram size, the visual and mental ability of a modeler is stretched, and the poor quality layout will be more noticeable.* *The findings show that there exists a negative correlation between diagram size and modeler performance.*	[S93]
complex interactions among class-diagram constraints:	The main reason of the correctness defects in the class diagrams is due to the complex interaction between the class-diagrams constraints. Constraints are defined as the restrictions acted on the components of class diagram. *The findings show that increasing the awareness leads to a surge in the rate of the identification.*	[S94]
Diagram type: UML class diagrams generally provided a better support for both comprehension and modification:	Entity-relationship along with its extensions are considered as the most rampant notions used in database conceptual modelling, while due to the success of object-oriented approach has also encouraged the models to use UML class diagrams for representing the data. *The findings show that UML class diagram exhibit a proper support both in terms of comprehension and modification during data modelling, compared to ER diagram.*	[S95]
Impact of aspect-oriented modelling on readability of UML state machines:	Aspect-orientation supply an enhanced modularization through separating out the crust cutting concerns in the form of separate entities called aspects. Crosscutting concerns refer to the robustness or security aspect and they can be modelled as aspects models. *Using aspect-oriented modelling there would be better performance in defect detection and fixing rates, compared to flat and hierarchal state machines, from perspective of comprehension, no significant difference was observed.*	[S96]
Effect of antipattern on quality enhancement	Antipatterns show the debatable diagrammatic and textual structures. Through their detecting it is possible to review the debatable structures or to verify their correctness. The results indicate that applying the technique improves the overall quality and clarity of use case models.	[S97]
Effect of conceptual mental model on quality of use case diagram and actors in terms of correctness, completeness, and redundancy of the UCD	Full Read Required *The findings show that CMM has a positive impact on use case diagrams in defining the system requirements, relations and actors and generation of better use case diagrams regarding capturing, analyzing and defining the functional system requirements in terms correctness, completeness and redundancy.*	[S98]

Effect of ontological modelling guideline :	The main assumption of Bunge's ontology states that "the world is made up of substantial things which are assumed to really exit". He states that a thing has a properties which can be intrinsic or mutual [35]. The ontological guidelines are capable of enhancing the usefulness of UML class diagrams for describing the application domain.	[S99]
Effect of model types (OO-DFD and Use Case Methods) on comprehension during FR modelling:	The system requirements can be modelled using various modelling notations. UML use case and object-oriented DFD are two accepted means for modelling the functional requirements. Both of these methods are based on semi-structured descriptions *The findings show that during functional requirements, use case models are better in terms of quality, than the models which are creased using OO-DFD transaction method, since the former is simple, while the latter leads to creased of highly detailed models which are error prone.*	[S100]
Effect of work system snapshots:	A work system is defined as a system where human and machines accomplish work using information, technology and other resources to produce products or offer services, the a work system snapshot is a one-page brief description of a work system identifying the main components of six central elements: customers, product/services, processes and activities, participants, information, and technologies. *According to the findings, the experiments, it became evident that a work system snapshots leads to production of quality use case diagrams.*	[S101]
Measuring design testability	The UML class diagram can be studied in terms of detecting and resolving testability weakness. Testability, is defined as the easiness to test a piece of software, is a strongly desired feature of software. It is able to make the validation phase more efficient by exposing faults during testing, and consequently to boost the quality of the end-product for users' satisfaction. It is deemed that the class interaction is an important attribute in terms of design testability. In fact class interaction emerges when potentially there are concurrent client/supplier associated between classes of the systems and such interactions need to be enhanced. *we can deduce that the class interactions can be combined in different ways; in some cases, not all the class interactions have to be considered, in others cases, it is necessary to deal with all the class interactions*	[S102]
Effect of metaassociations on metamodels	There exits generally two approaches for extending UML models: Heavyweight extensions and Lightweight extensions. By adopting a heavyweight extension of UML meta model, it is common to encounter with emergence of some metaassociations in the metamodel at level of M2. *It was observed that, albeit imposing metassociations are rampant for UML metamodel extensions, but they are not correctly defined in those extensions, which causes the models to be inaccurate and incomplete.*	[S103]