Definition and Validation of Consistency rules between UML diagrams

Damiano Cosimo Torre

[redacted]

Ph.D. Cotutelle Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering (Carleton University)

Doctorado en Tecnologías Informáticas Avanzadas (UCLM)

Cotutelle supervisors:

Yvan Labiche – Carleton University, Ottawa, Canada

Marcela Genero – University of Castilla-La Mancha, Ciudad Real, Spain

Maged Elasaar – Carleton University, Ottawa, Canada

Ottawa, September 18th, 2018

Dedication

To my wife Trycia Bazinet.

Alla mia famiglia: mamma, papà, Antonella, Sabina, Sara, Maria Rita, Maria Teresa, Antonio, Davide, Ruggiero e Rocco.

To the memory of Dr. Carlo Marcelletti and Dr. Francois Marie Fontan, and to all the past/present/future Fontan patients.

Acknowledgments

Trycia Bazinet for her love and to be my source of inspiration.

Mamma (Maria Tanzi), papa (Torre Ruggiero), Antonella, e Sabina per il loro supporto e amore incondizionato.

My supervisors for believing in me, for their guidance and constant and reliable support: Prof. Yvan Labiche, Prof. Marcela Genero and Dr. Maged Elasaar.

My friends in Italy, Spain and Canada, especially John, Gello and Marcello.

My colleagues at the SQUALL Lab in Canada, and at the Alarcos Group in Spain.

The administrative staff at Carleton University and Universidad de Castilla-La Mancha.

My former mentor and supervisor Prof. Giuseppe Visaggio.

Ospedale Pediatrico Bambino Gesù in Roma, and especially Dr. Salvatore Giannico.

Abstract

English: UML diagrams describe different views of one piece of software. These diagrams strongly depend on each other and must therefore be consistent with one another, since inconsistencies between diagrams may be a source of faults during software development activities that rely on these diagrams. It is therefore paramount that consistency rules be defined and that inconsistencies be detected, analyzed and fixed. Even though many researchers have proposed, explicitly or not, rules to detect inconsistencies, no well-accepted, as complete as possible set of consistency rules has so far been described and published. Although the UML standard itself contains some consistency rules, often referred to as well-formedness rules, the standard does not offer a complete list since for instance some consistency rules may be specific to the way the UML notation is used. This lack of well-accepted list of UML consistency rules forces researchers to systematically define the consistency rules they rely on for their own research. Although this is good practice, researchers describe similar or even identical consistency rules, over and over again. This fact motivated our main research objective, which is to identify and validate a set, as complete as possible, of well-accepted consistency rules for UML diagrams. To achieve this objective the following two research questions were identified and studied:

1. *What are the UML consistency rules proposed in the literature?* To answer this question, we present the results of a Systematic Mapping Study about UML consistency rules. We finally identified a set of 116 UML consistency rules avoiding redundant definitions or definitions already in the UML standard.
2. *Is this list of UML diagram consistency rules relevant?* To answer this question, we completed a process of validation of the rules in the following way: a) we organized the 1st International Workshop on UML Consistency Rules (WUCOR) during the 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS) in 2015. The workshop represented a valid initial step in the process of validating the rules, but the limited number of attendees called to improve and continue this process; b) we developed a survey (with a questionnaire consisting of 21 questions) with experts in the field from academia and industry with the following main objectives: i) surveying the diffusion and importance of model consistency issues for practitioners, ii)

validating the UML consistency rules, i.e. identifying the rules that should be always enforced. Through this work we finally identified a set of 52 rules that should be enforced in every UML model; c) we developed a case study whereby we presented the process and the results of checking 33 of the 52 consistency rules, translated in OCL, on 34 open source UML Papyrus models with a total of 206 different UML diagrams. Finally, the OCL consistency rules triggered 2731 different inconsistencies among the UML diagrams.

Español: Los diagramas UML describen diferentes vistas de un producto software y en gran medida están relacionados entre sí. Es por ello que estos diagramas deben ser consistentes entre sí, ya que los diagramas inconsistentes pueden ser una fuente de problemas en las actividades de desarrollo del software que se basan en diagramas. Por lo tanto, es fundamental que estas inconsistencias sean detectadas, analizadas y corregidas. A pesar de que muchos investigadores han propuesto, explícitamente o no, reglas para detectar inconsistencias. Aunque el estándar UML propuesto por el OMG en sí contiene algunas reglas de consistencia, llamadas en las versiones anteriores del UML como "well-formedness rules", no ofrece una lista completa de reglas. Por ejemplo, algunas reglas de consistencia pueden ser específicas del dominio en que se utiliza UML. La falta de una lista completa de reglas de consistencia para diagramas UML obliga a los investigadores a definir sistemáticamente nuevas reglas de consistencia. Aunque esta podría considerarse una buena práctica, esto hace que los investigadores tengan que describir reglas similares, una y otra vez. Por esta razón, el principal objetivo de esta investigación, es identificar y validar un conjunto, lo más completo posible, de reglas de consistencia para diagramas UML. Para lograr este objetivo, se han propuesto las siguientes dos preguntas de investigación:

1. ¿Cuáles son las reglas de consistencia para diagramas UML propuestas en literatura? Para responder a esta pregunta, presentamos los resultados de un mapeo sistemático de la literatura llevado a cabo para recolectar las reglas existentes en la literatura. De esta manera identificamos un conjunto de 116 reglas UML de consistencia, sin redundancias y que no están incluidas en el estándar UML.
2. ¿Es esta lista de reglas UML de consistentes relevante? Para responder a esta pregunta, hemos realizado el siguiente proceso de validación de las reglas: a) Organización del 1er International Workshop on UML Consistency Rules (WUCOR) durante la 18 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems

(MoDELS) en 2015. El workshop fue el primer paso para el proceso de validación de las reglas, pero el número limitado de participantes nos llevó a tener que continuar este proceso de validación, ; b) Realización de una encuesta (con un cuestionario que consta de 21 preguntas) con expertos en MDSE de la academia y de la industria con los siguientes objetivos: (i) estudiar la difusión e importancia de los problemas de consistencia del modelo en el contexto general del MDSE, así como en el contexto UML, (ii) validar las reglas UML de consistencia propuestas, es decir, identificar las reglas que siempre deben aplicarse. Este trabajo finalmente identificó un conjunto de 52 reglas que deberían aplicarse en cada modelo UML; c) Desarrollo de un estudio de caso mediante el cual presentamos el proceso y los resultados de verificación de 33 de las 52 reglas de consistencia, traducidas en OCL. Estas reglas se verificaron en 34 modelos UML Papyrus de código abierto con un total de 206 diagramas UML diferentes. Finalmente, las reglas de coherencia OCL encontraron 2731 inconsistencias diferentes entre los diagramas UML.

Contents

Dedication ii

Acknowledgments iii

Abstract iv

Index of Figures xiii

Index of Tables xvi

1. INTRODUCTION 1

1.1 From Model Based Engineering to Model Driven Architecture 2

1.2 The Unified Modelling Language (UML) 4

1.3 UML Consistency 5

1.4 Thesis contributions 7

1.5 Thesis at a glance 10

2. STATE OF THE ART 11

3. WHAT ARE THE UML CONSISTENCY RULES IN THE LITERATURE? (G1) 14

3.1 A Systematic Mapping Study of UML consistency rules 15

3.1.1 Planning the Mapping Study 15

3.1.1.1 Research Questions 15

3.1.1.2 Search strategy 16

3.1.1.2.1 Process of the SS selection 17

3.1.1.3 Selection procedure and inclusion and exclusion criteria 17

3.1.1.4 Data extraction strategy 18

3.1.2 Execution 21
3.1.3 Results 23
3.1.3.1 UML version (RQ1) 23
3.1.3.2 Types of UML diagrams (RQ2) 24
3.1.3.3 UML consistency rules (RQ3) 25
3.1.3.4 UML consistency dimensions (RQ4) 26
3.1.3.5 Research type facets (RQ5) 27
3.1.3.6 Tool support (RQ6) 27
3.1.3.7 UML consistency rules: specification and support (RQ7) 27
3.1.3.8 Additional results 29
3.1.4 Discussion 30
3.1.4.1 Combining RQ1, RQ2 and RQ5 30
3.1.4.2 Combining RQ6 and RQ7 31
3.1.5 Conclusions of Section 3.1 32
3.2 A systematic identification of UML consistency rules 32
3.2.1 Definition of the Systematic Procedure 34
3.2.1.1 Research questions 34
3.2.1.2 Inclusion and exclusion criteria 35
3.2.1.3 Data extraction strategy 36
3.2.2 Execution 38
3.2.3 Results 39
3.2.3.1 What are the existing UML consistency rules? (RQ1) 41
3.2.3.2 Which types of consistency problems are tackled in the existing rules? (RQ2) 42
3.2.3.3 What types of UML diagrams are involved in UML consistency rules? (RQ3) 42
3.2.3.4 For what software engineering activities are UML consistency rules used? (RQ4) 46
3.2.3.5 Combining RQ4 and RQ3 47

3.2.3.6 Which UML diagram elements are involved in UML consistency rules? (RQ5) 50
3.2.3.7 Additional results 51
3.2.4 Conclusions of Section 3.2 51
4. IS THIS CONSOLIDATED LIST OF UML DIAGRAM CONSISTENCY RULES RELEVANT? (G2) . 52
4.1 1st International Workshop on UML Consistency Rules (WUCOR) 52
4.1.1 Previous events on UML consistency 53
4.1.2 Workshop goals 54
4.1.3 Summary of the presented papers 54
4.1.4 Description of the activities 55
4.1.4.1 UML Consistency rules and dimensions (A1) 56
4.1.4.2 UML diagrams involved in UML consistency rules (A2) 56
4.1.4.3 UML Consistency rules in MDD (A3) 57
4.1.5 Results 58
4.1.6 Conclusions of Section 4.1 60
4.2 Online survey: how consistency is handled in MDSE and UML....60
4.2.1 Research Methodology 62
4.2.1.1 Survey Design 62
4.2.1.1.1 Identifying the Target Audience and the Sample of Respondents 62
4.2.1.1.2 Designing Survey Questions 63
4.2.1.2 Execution 64
4.2.1.2.1 Pre-considerations and data validation 65
4.2.2 Survey results 66
4.2.2.1 Demographic Information 66
4.2.2.1.1 Respondents' working countries (Q1) 66
4.2.2.1.2 Respondents' level of education (Q2) 67

4.2.2.1.3 Respondents' current working place (Q3) 67
4.2.2.1.4 Respondents' experience in the context of software engineering (Q4) 68
4.2.2.1.5 Respondents' experience in the context of MDSE (Q5) 69
4.2.2.1.6 Respondents' current position(s) (Q6) 70
4.2.2.1.7 Number of researchers in respondents' research group – number of employees in respondents' companies (Q7) – Which of them are working in the context of MDSE? (Q8) 71
4.2.2.2 Model Consistency 72
4.2.2.2.1 Frequency of model consistency problems while developing software (Q9) 72
4.2.2.2.2 Importance of the model consistency topic (Q10) 73
4.2.2.2.3 MDSE activities where model consistency constraints are used (Q11) 74
4.2.2.2.4 Respondents' familiarity with the different dimensions and types of Consistency (Q12) – Respondents' experience with consistency issues involving different dimensions and types of consistency (Q13) 75
4.2.2.2.5 How model consistency constraints are specified (Q14) 76
4.2.2.2.6 Tools used to check model consistency (Q15) 77
4.2.2.2.7 Tools that are used to represent models (Q16) 77
4.2.2.2.8 Modeling languages that are used to develop models of software systems (Q17) 78
4.2.2.2.9 Combining Q8 with Q14-Q17 79
4.2.2.3 UML Consistency 81
4.2.2.3.1 UML diagrams mostly involved in consistency issues (Q18) 81
4.2.2.3.2 Studying, detecting and handling UML model inconsistencies (Q19) 82
4.2.2.3.3 UML Consistency Rules (Q20 and Q21) 83
4.2.2.3.4 Combining Q4-Q5 with Q20-Q21 87
4.2.3 Conclusions of Section 4.2 89
4.2.3.1 RQ1: Demographic information of the survey participants. 89
4.2.3.2 RQ2: How consistency problems in the context of MDSE are handled and detected 89

4.2.3.3 RQ3: How UML inconsistencies are handled 90
4.2.3.4 RQ4: Relevancy and understandability of the 116 UML consistency rules 90
4.3 UML consistency rules: a case study with Eclipse Papyrus models 91
4.3.1 Related work – Section 4.3 91
4.3.2 Case study 93
4.3.2.1 Design 94
4.3.2.1.1 Encoding the UML consistency rules in OCL. 95
4.3.2.1.2 Collecting UML models. 98
4.3.2.1.3 Checking OCL rules 100
4.3.2.2 Data Collection 100
4.3.2.3 Execution and results 102
4.3.2.4 Analysis of the results 106
4.3.3 Conclusions of Section 4.3 111
5. THREATS TO VALIDITY 112
5.1 Threat to Validity – A Systematic Mapping Study of UML consistency (Section 3.1) 112
5.2 Threat to Validity – A systematic identification of UML consistency (Section 3.2) 113
5.3 Threat to Validity – Online survey: how consistency is handled in MDSE and UML (Section 4.2) 116
5.4 Threat to Validity – UML consistency rules: a case study with Eclipse Papyrus models (Section 4.3) 117
6. CONCLUSION AND FUTURE WORK 120
References 128
Appendix A 141

Appendix B 151

Appendix C 155

I. SMS – Planning and execution 155

II. Systematic mapping study – results 161

III. Summary 170

IV. Conclusions 171

Appendix D 174

Appendix E 184

Index of Figures

Figure 1. MD* Schema 3

Figure 2. Taxonomy of the UML structure and behavior diagrams 5

Figure 3. UML qualities [7] 6

Figure 4. UML version 24

Figure 5. Number of primary studies with rules involving each UML diagram 25

Figure 6. Number of primary studies using a particular language for specifying UML consistency rules 28

Figure 7. Number of primary studies using a tool to check UML consistency rules 28

Figure 8. Number of primary studies per year 30

Figure 9. Combining RQ1, RQ2 and RQ5 31

Figure 10. Summary of UML Consistency rules selection (see accompanying text for a detailed description of the semantics of the figure) 43

Figure 11. UML Consistency rules grouped into UML diagrams (see accompanying text for a detailed description of the semantics of the figure). 45

Figure 12. Rules between UML diagrams and uses in Software Engineering activities (see accompanying text for a detailed description of the semantics of the figure). 47

Figure 13. Summary of UML Consistency rules between UML diagrams (see accompanying text for a detailed description of the semantics of the figure) 48

Figure 14. Summary of the UML diagram elements involved in UML Consistency rules 50

Figure 15. Combining question 1 and 2 of A3 (see accompanying text for a detailed description of the semantics of the figure). 59

Figure 16. Geographical distribution of respondents..... 67

Figure 17. How long have respondents been working in the context of Software Engineering (see accompanying text for a detailed description of the semantics of the figure). 68

Figure 18. How long have respondents been working in the context of MDSE (see accompanying text for a detailed description of the semantics of the figure)..... 70

Figure 19. Respondents' current positions 71

Figure 20. (a) number of researchers in respondents' research group / company (Q7) – (b) which are working in MDSE (Q8)..... 72

Figure 21. MDSE activities in which models consistency constraints are used (see accompanying text for a detailed description of the semantics of the figure)..... 74

Figure 22. Results about familiarity and experience of respondents regarding to different types and dimensions of model consistency (see accompanying text for a detailed description of the semantics of the figure)..... 75

Figure 23. How models consistency constraints are specified (see accompanying text for a detailed description of the semantics of the figure)..... 76

Figure 24. Tools that are used to check models consistency (see accompanying text for a detailed description of the semantics of the figure)..... 77

Figure 25. Tools that are used to represent models (see accompanying text for a detailed description of the semantics of the figure). 78

Figure 26. Modeling languages that are used to represent models (see accompanying text for a detailed description of the semantics of the figure). 79

Figure 27. Combining Q8 with Q14, Q15, Q17 and Q16 (see accompanying text for a detailed description of the semantics of the figure)..... 80

Figure 28. UML diagrams that are mostly involved in consistency issues (see accompanying text for a detailed description of the semantics of the figure)..... 82

Figure 29. How UML model inconsistencies are studied, detected, and handled (see accompanying text for a detailed description of the semantics of the figure)..... 83

Figure 30. The set of 116 UML consistency rules most selected in Q21 86

Figure 31. Combining Q4-Q5 with Q20-Q21 (see accompanying text for a detailed description of the semantics of the figure)..... 88

Figure 32. Case study plan (see accompanying text for a detailed description of the semantics of the figure)..... 95

Figure 33. UML consistency rules file 96

Figure 34. Eclipse Papyrus steps to check OCL rules (see accompanying text for a detailed description of the semantics of the figure)..... 101

Figure 35. Triggering OCL rules 107

Figure 36. Series of questions for respondents with industry/academic experience 151

Figure 37. UML diagrams in Synthesis Techniques (see accompanying text for a detailed description of the semantics of the figure)..... 162

Figure 38. From → To Synthesis Techniques (see accompanying text for a detailed description of the semantics of the figure)..... 163

Index of Tables

Table 1. Summary of state of the art 11

Table 2. Research questions 16

Table 3. Summary of primary studies selection on 12/12/2012 22

Table 4. Results of SMS 23

Table 5 UML consistency rule between Sequence Diagram and Use Case Diagram 25

Table 6 UML consistency rules of Class Diagram 26

Table 7. Publication venues ranking 29

Table 8. Research questions 34

Table 9. Data extraction form 36

Table 10. Results at a glance 40

Table 11. Q20 and Q21 results for each of the 116 consistency rules 84

Table 12: Summary of related work 92

Table 13: Summary of UML models collection 99

Table 14: Sample of the data collection form 101

Table 15: Raw results after the execution of the case studies. Rules, Diagrams and Model IDs are in bold; number of inconsistencies are below these. 105

Table 16: Summary of the results according UML diagrams 109

Table 17: Analysis of the results according UML diagrams 110

Table 18. Set of UML Consistency rules 141

Table 19. Survey questionnaire 151

Table 20. Summary of Primary Study Selection 160

Table 21. SMS results at a glance 161

Table 22. Summary of UML Consistency Rules 166

Table 23. Summary of Synthesis and Support 170

Table 24. 95 Primary Studies 174

Table 25. Search strings tried on 15/09/2012 184

Table 26. Search strings tried on 05/10/2012. 185

Table 27. Search strings tried on 08/10/2012. 185

Table 28. Search fields on 08/10/2012. 186

Table 29. SS and SF on 03/11/2012 187

Table 30. Search string 187

1. INTRODUCTION

When one opens Google Maps[1] in a browser, one typically sees a map of North America. This map will show only the most significant features, for example, states´ names and borders, the various mountain ranges, the ocean currents, and other extremely large structures. Small features will almost certainly be omitted. A subsequent zoom-in on the map will cover a smaller geographical region and will typically possess more details. For example, a zoom-in on the province of Ontario, will now show major cities and highways. By zooming-in on an even smaller region, such as a town like Ottawa, the map will include smaller geographical features, such as the names of individual mountains, lakes (e.g. see above the Pink lake of Gatineau Park, QC), the most important roads, neighborhoods´ names, etc.

At each level, certain information is included, and certain information is purposely omitted. There is simply no way to represent all the details when an artifact is viewed at a higher level. Fundamentally, people can use only a "few" simple tools to create, understand, or manage complex systems. This cognitive process of the human mind is called abstraction. Abstraction is a process to elicit a subset of objects that shares a common property from a given set of objects and to use the property to identify and distinguish the subset from the whole in order to facilitate reasoning [1]. Abstraction is also widely applied in science and technology, and it is necessary when modeling. A model can be informally defined as a simplified or partial representation of the reality,

[1] https://maps.google.com

defined in order to accomplish a task or to reach an agreement on a topic. Therefore, by definition, a model will never describe reality in its entirety [2]. The term “model” is derived from vulgar Latin word "modellus", equivalent to the Latin word "modulus", which means measure, rule, pattern, example to be followed [3].

In this Ph.D. thesis, we focus on modeling during software development. This topic is involved in the following modelling approaches: Model Based Engineering (MBE), Model-Driven Engineering (MDE), Model Driven Development (MDD) and Model Driven Architecture (MDA) for software artifacts.

1.1 From Model Based Engineering to Model Driven Architecture

In this Section, we present an overview of the acronyms presented at the end of the previous Section. The following schema is inspired by the one presented by Brambilla and colleagues [2]. Figure 1 was created with yUML[2] and it shows a Class Diagram that describes the relations between the above acronyms which describe different modeling approaches. Starting the reading of Figure 1 with Model-Driven Architecture (MDA), it can be regarded as a subset of Model-Driven Development (MDD), where the modeling and transformation languages are standardized by the Object Management Group (OMG). MDD is a development paradigm that uses models as the primary artifact of the development process. Usually, in MDD the implementation is (semi)automatically generated from the models. Model Driven Engineering (MDE) would be a superset of MDD because, as the E in MDE suggests, MDE goes beyond of the pure development activities and encompasses other model-based tasks of a complete software engineering process (e.g., the model-based evolution of the system or the Model-Driven reverse engineering of a legacy system). Later in this manuscript we will refer to Model Driven Software Engineering (MDSE) as MDE in the context of Software Engineering.

[2] https://yuml.me

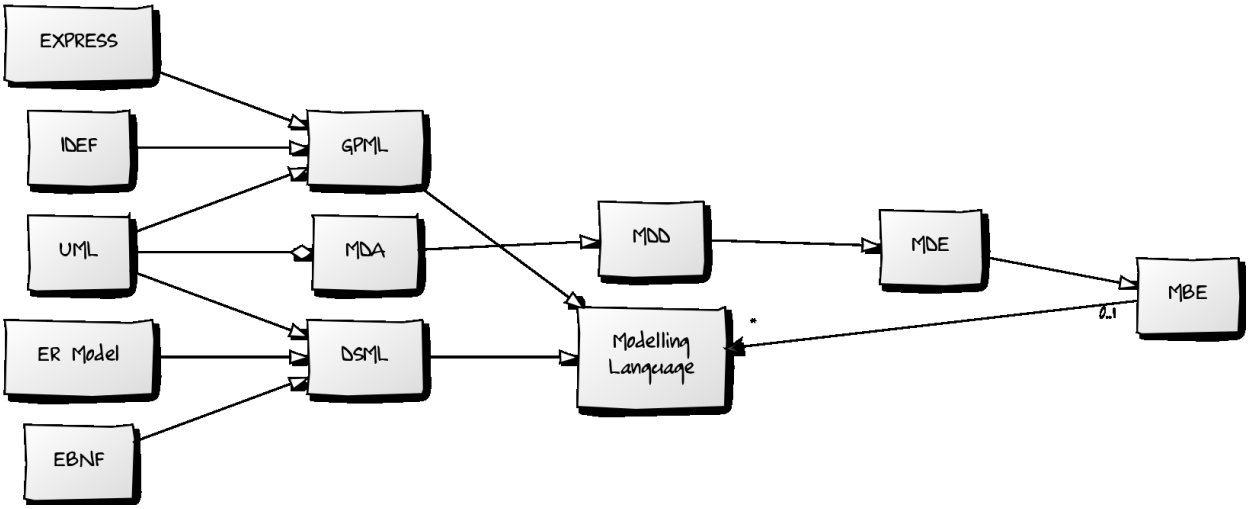

Figure 1. MD* Schema

Finally, Model-Based Engineering (MBE) refers to a softer version of MDE. That is, the MBE process is a process in which software models play an important role although they are not necessarily the key artifacts of the development (i.e., they do not drive the process as in MDE, MDD or MDA). An example would be a development process where, in the analysis phase, designers specify the domain models of the system but then these models are directly handed out to the programmers as blueprints to manually write the code (no automatic code-generation involved and no explicit definition of any platform-specific model). In this process, models still play an important role but are not the central artifacts of the development process and may be less complete (i.e., they can be used more as blueprints or sketches of the system) than those in a MDE, MDD, or MDA approach. MBE is a superset of MDE. All Model-Driven processes are model-based, but not the other way around.

Modeling languages are one of the main ingredients of all these acronyms. A modeling language is a tool that lets designers specify the models for their systems. Modeling languages are the tools that allow the definition of a concrete representation of a conceptual model and may consist of graphical representations, textual specifications, or both. In any case, they are formally defined and ask the designers to comply with their syntax when modeling. Two big classes of languages can be identified [2]:

- Domain-Specific Modelling Languages (DSML) are languages that are designed specifically for a certain domain, context, or company to ease the task of people that need to describe things in that domain. DSMLs have been largely used in computer science even

before the acronym existed: examples of DSMLs are Entity-Relation Model and Extended Backus-Naur Form (EBNF).

- General-Purpose Modeling Languages (GPML) instead represent tools that can be applied to any sector or domain for modeling purposes. The typical example for this kind of languages is the UML language suite, or languages like IDEF[3] or EXPRESS[4].

It is important to say that deciding whether a language is a DSML or GPML is not a binary choice. For instance, at first sight UML and alike languages seem to belong to the latter group in this classification since UML can be used to model any kind of domain. Moreover, the decision on whether to use UML (or any other GPML) or a DSML for one's next modeling project is far from being a trivial one [2].

1.2 The Unified Modelling Language (UML)

In this Section, we briefly present an overview of the Unified Modelling Language (UML). As described in the previous Section, MDA is the particular vision of MDD proposed by the OMG. The MDA [4] promotes a set of transformations between successive models from requirements to analysis, to design, to implementation, and to deployment [5]. Much attention has been paid to MDA by both the academia and industry in recent years [6-8], which has resulted in models gaining even more importance in software development.

[3] http://www.idef.com

[4] https://www.loc.gov/preservation/digital/formats/fdd/fdd000449.shtml

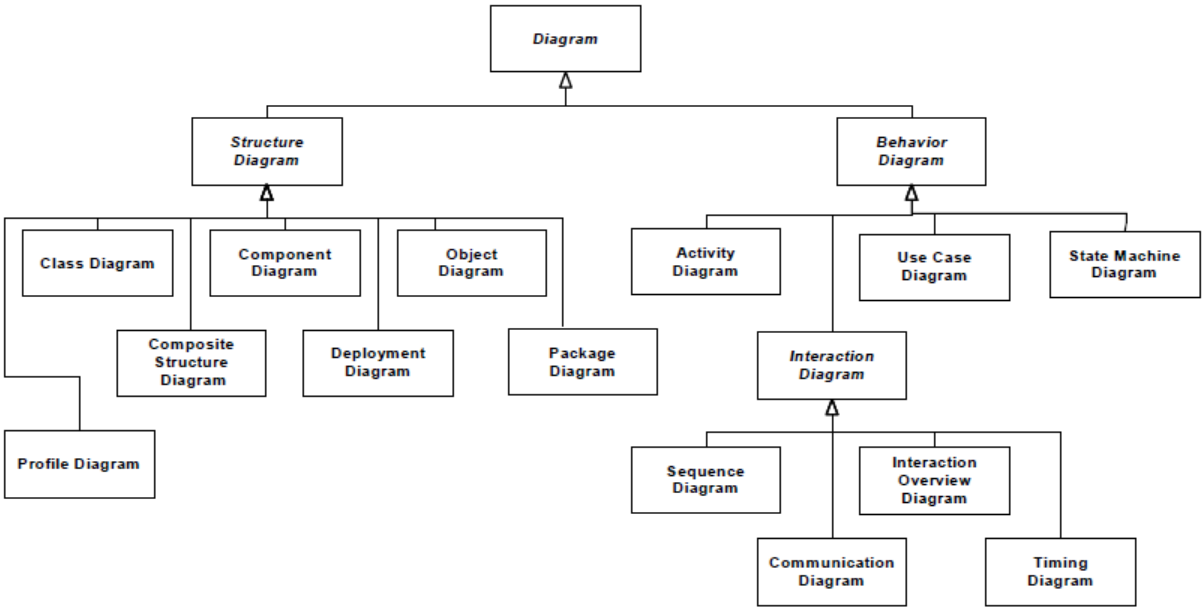

Figure 2. Taxonomy of the UML structure and behavior diagrams

The Unified Modeling Language (UML) [9] is the OMG specification most frequently used and is often described as the de-facto standard modeling language for object-oriented modeling and documentation [10]. It is the privileged modeling language to implement the MDA. It is however not to be used in every single software development project [11]. The UML provides 14 diagram types (see Figure 2) [9] that can be used to describe a system from different perspectives (e.g., structure, behaviour) or abstraction levels (e.g., analysis, design), which helps deal with complex systems and distribute responsibilities between stakeholders, among other benefits. Since the various UML diagrams describe different aspects of one, and only one, software under development, they are not independent but strongly depend on each other in many ways, meaning that UML diagrams describing a software product must be consistent with one another.

1.3 UML Consistency

According to Genero and colleagues [7], UML consistency is the UML quality that has received more attention (among others) by researchers in terms of number of papers (conference, journal, etc.) published from 1997 through 2009 (see Figure 3).

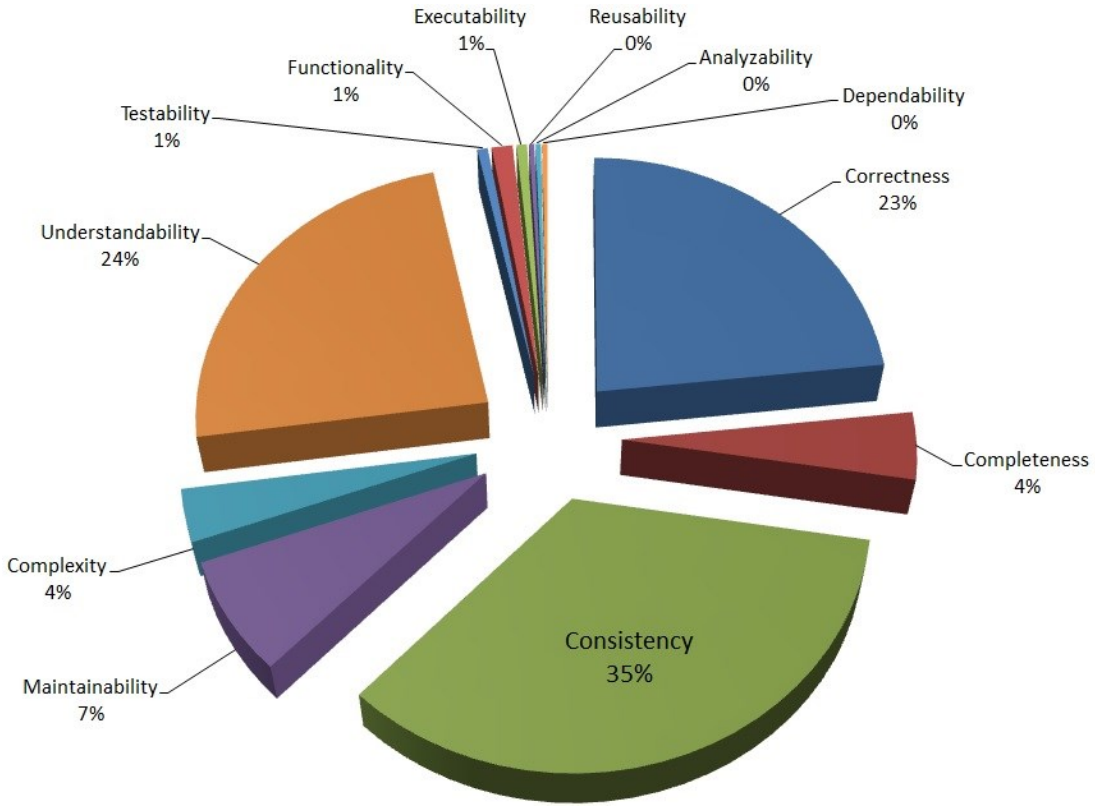

Figure 3. UML qualities [7]

As UML is not a formal notation, inconsistencies may arise in the UML specification of a complex system when such specification requires multiple diagrams/views to describe different perspectives of the software/model [12]. In this thesis we refer to UML diagrams as the different views of a software (i.e. a model) under development. When UML diagrams portray contradicting or conflicting meaning, the diagrams are said to be inconsistent [13]. Such inconsistencies may be a source of faults in software systems [14, 15]. It is therefore paramount that they be detected, analyzed and fixed [16].

Even though many researchers have proposed, explicitly or not, rules to detect inconsistencies, no well-accepted, as complete as possible set of consistency rules has so far been described and published. Although the UML standard provided by the OMG itself contains some consistency rules, often referred in the previous versions of the UML as well-formedness rules, the standard does not offer a complete list since for instance some consistency rules may be specific to the way the UML notation is used.

This lack of well-accepted list of UML consistency rules forces researchers to systematically define the consistency rules they rely on for their own research [12]. Although this is good practice, researchers (as confirmed by some of our own results [11]), describe similar or even identical consistency rules, over and over again. This fact motivated our main research objective, which is to identify and validate a set, as complete as possible, of well-accepted consistency rules for UML diagrams. To achieve this objective the following two main research goals were identified:

G1): What are the UML consistency rules proposed in the literature (Chapter 3)?

G2): Is this list of UML diagram consistency rules relevant (Chapter 4)?

1.4 Thesis contributions

The main goal of this Ph.D. thesis is to carry out the systematic identification and the validation of a set, as complete as possible, of well-accepted consistency rules for UML diagrams. To achieve this objective, we presented two main research goals in Section 1.3, respectively to collect and validate the UML consistency rules. As follows, we present the contributions carried out to answer the two main research goals:

- **G1** (What are the UML consistency rules proposed in the literature?) – Chapter 3.
    - Systematic Mapping Study (SMS) to collect the current state of the art in terms of UML consistency rules (Section 3.1). This SMS and accompanying results have been published and presented at the 18th ACM International Conference on Evaluation and Assessment in Software Engineering (EASE) in 2014, a premier conference in the field [11];
    - A systematic identification of UML consistency rules (Section 3.2). This work presented the results obtained after following a systematic protocol, whose aim was to identify, present and analyze a consolidated set of 116 UML Consistency rules obtained from the work presented in Section. This work was accepted for publication at the International Journal of Systems and Software [17].

- **G2** (Is this list of UML diagram consistency rules relevant?) – Chapter 4.
    - 1st International Workshop on UML Consistency Rules (WUCOR) (Section 4.1). We organized WUCOR during the 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015) (Section 4.1) [18-20]. The goal of this workshop was to gather input and feedback on the 116 UML consistency rules (identified in Sections 3.1 and 3.2) from the MoDELS conference community. Due to limited attendance at the workshop, we were only able to receive feedback from experts in the field on 81 out of 116 UML consistency rules.
    - How consistency is handled in Model Driven Software Engineering and UML: a survey of experts in academia and industry (Section 4.2). This work provided the results of a survey that investigated how model consistency is handled in the context of MDSE and UML with experts from academia and industry. We developed a survey with MDSE experts from academia and industry to: (1) survey the diffusion and importance of model consistency issues; (2) validate the 116 UML consistency rules identified in Sections 3.1 and 3.2. The results of the work presented in Section 4.2 (i.e. [21]) will be submitted to an international journal in the near future.
    - UML consistency rules: a case study with Eclipse Papyrus models (Section 4.3). Starting with the set of 52 UML consistency rules identified in Section 4.2, the goal of this case study was to investigate to what extent existing UML models satisfy those rules. To do that, we described an exploratory flexible indirect case study [22] to verify the UML consistency rules on the UML models. The results of the work presented in Section 4.3 will be submitted to an international conference in the near future.
- Additional publications carried out in the context of this Ph.D. thesis:
    - On collecting and validating UML consistency rules: a research proposal, that has been published and presented the Doctoral Symposium at 18th International Conference on Evaluation and Assessment in Software Engineering (EASE) in 2014 [23].

- On validating UML consistency rules, that has been published and presented at the 26th IEEE International Symposium on Software Reliability Engineering, Doctoral Symposium (ISSRE) in 2015 [24].
- UML consistency rules in technical books, that has been published and presented at the 25th IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE) in 2015 [25].
- Verifying the consistency of UML models, that has been published and presented at the 27th IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE) in 2016 [26].
- Systematic Mapping Study (SMS) to collect the current state of the art in terms of consistency rules identified in UML synthesis techniques (see Appendix C). This work presents the results obtained after applying the SMS protocol with the aim of identifying and evaluating the current approaches for synthesizing UML diagrams from other UML diagrams. This work has been published and presented at the 10th Workshop on Modelling in Software Engineering (MiSE) in 2018 [27].

As follows, we present the summary of our results in regard to G1 and G2, which will be further discussed in the conclusion of this thesis:

- **G1** (What are the UML consistency rules proposed in the literature?)
  - From an initial set of 621 UML consistency rules, published in literature and extracted from seven scientific research databases, a total of 116 rules were eventually selected and analyzed in depth, by following a precise selection protocol driven by four research questions (see Appendix A).
- **G2** (Is this list of UML diagram consistency rules relevant?)
  - Regarding the 116 UML consistency rules we identified in Section 3.2, we conclude that 105 of the 116 (90.5%) UML consistency rules were considered important to check in UML models (always or at least in some situations) by the majority of the respondents (see Section 4.2.2.3.3). Based on our analysis of the rules' responses, we identified 52 out of 116 rules (see Section 4.2.2.3.3) that the vast majority (76.6%) of respondents considered

to be relevant to enforce in every UML model. We believe that this subset of 52 rules can be valuable in the MDSE/UML contexts in many ways, such as: (a) it could be considered to be added to the UML standard, (b) it can be used as a reference to researchers who study UML and model driven technologies in general, and as a result, they will simply need to refer to this list rather than re-inventing the wheel [11], c) it can be used as a practical example in the classroom to better convey to students that UML is more than lines and boxes [28].

- Starting with the set of 52 UML consistency rules identified in Section 4.2, in Section 4.3 we described an exploratory flexible indirect case study [22] to verify the UML consistency rules on the UML models.

1.5 Thesis at a glance

This Ph.D. thesis is structured as follows. In Chapter 2 we provide a discussion on the state of the art. This is followed by a description of the answer to G1 in Chapter 3, where we present the results of a systematic mapping study about UML consistency rules (Section 3.1) and the identification of a consolidated set of UML consistency rules (Section 3.2). The answer to G2 is provided in Chapter 4, where we present the results obtained from the proceedings of the 1st International Workshop on UML Consistency Rules (WUCOR) (Section 4.1), the results of online survey about the relevance of the consolidated set of UML Consistency rules (Section 4.2), and finally the results obtained by the evaluation of some UML consistency rules that were checked against UML models (Section 4.3). Finally, Chapter 5 discusses threats to the validity of our results and Chapter 6 draws the conclusion and future work.

2. STATE OF THE ART

Since this Ph.D. thesis focuses on the systematic identification and validation of UML consistency rules, the state of the art pertains to the systematic discussions of UML consistency. In Table 1, we present a summary of the five pieces of work that fit this description. For each one, the table indicates the period covered by the survey, the main objective, whether the search was driven by clear research questions (RQs) and was systematic, along with the number of reviewed papers and UML consistency rules collected. We first discuss the only review [29] that presented UML consistency rules, and then the other four reviews on UML consistency which we considered important to include.

Table 1. Summary of state of the art

Reference	Search period	Summary of the main objective	RQs	Type of Study	Papers reviewed	Rules identified
[29]	2008-2011	They propose a rule-based method for consistency checking in UML models.	None	Not systematic	8	50
[16]	Up to 2000	They present a survey of techniques and methods to support the management of inconsistencies in various software models.	None	Not systematic	19	None
[8]	Up to 2007	They present a survey of different UML consistency checking techniques.	None	Not systematic	17	None
[6]	2001-2007	They aim to discover the various current model consistency conceptions, proposals, problems and solutions provided in the literature regarding UML model consistency management.	6	SMS	44	None
[30]	2002-2004	They evaluate the existing Description Logic-based approaches with which to check UML consistency.	None	Not systematic	3	None

The work of Kalibatiene and colleagues [29] is, to the best of our knowledge, the closest piece of work to our research since the authors searched for, collected and presented UML diagram consistency rules. They obtained 50 rules by reviewing eight articles, which they did not select by following a systematic protocol (Systematic Literature Review (SLR) or SMS): no mention was made of the processes used to obtain the articles or to include/exclude these documents. After conducting our research, we confirmed that all of their 50 rules are included in our initial set of

621 rules, which we obtained by following a precise, systematic and repeatable procedure. The authors did not present any process to validate their 50 rules.

The following four pieces of work are reviews, whose purpose is to increase the body of knowledge on UML diagram consistency, although not necessarily on UML consistency rules.

Spanoudakis and Zisman [16] presented a literature review on the problem of managing inconsistencies in software models in general, but not specifically UML models. They presented a conceptual framework that views inconsistency management as a process, which incorporates activities, with which to detect overlaps and inconsistencies between software models, diagnose and handle inconsistencies, track the information generated along the way, and specify and monitor the exact means used to carry out each of these activities. They then surveyed 19 research works published prior to 2000 that address one or more of the aspects of their conceptual framework. None of these 19 papers was identified in our study because they do not present any UML consistency rules.

Usman and colleagues [8] presented a literature review of consistency checking techniques for UML models. The authors argued that formalizing UML models is preferable to verifying consistency because this helps remove ambiguities and enforce consistency. They briefly reviewed 17 articles, which represent less than a quarter of the number of articles considered in our work (95): the two pieces of research have only ten studies in common, since our search has a different objective. During this survey, the authors did not follow any SLR or SMS protocols, and they simply provided an initial summary of their findings on UML consistency checking techniques (not consistency rules).

Lucas, Molina, and Toval [6] presented an SMS on UML consistency management in which they reviewed 44 papers published between 2001 and 2007, focusing on the consistency in two or more UML diagrams. This work is different from ours in several ways. The first important difference is that they did not present any UML consistency rules during their review. The second main difference is the purpose: they focused on the management of UML (in)consistencies, i.e., they focused on the techniques used to identify and fix inconsistencies, without providing a detailed discussion of which inconsistencies had to be identified and fixed. In contrast, our work focuses on those inconsistencies that need to be identified and fixed. A direct consequence of this difference is that we considered a broader number of articles in order to consolidate the set of UML

consistency rules (95 articles rather than 44), and approximately half of their studies (24) are not considered in our work.

Ahmad and Nadeem [30] presented a literature review restricted to Description Logic (DL)-based consistency checking approaches. They reviewed three articles, which are also studied in our research. Their main finding is that only class diagram, sequence diagram and state machine diagram inconsistencies were covered in the papers surveyed and that a few common types of inconsistencies were discussed.

Our work differs from the four reviews mentioned above [6, 8, 16, 30] in five ways: a different goal, a more extensive and systematic review process, the presentation of UML consistency rules, the evaluation of the rules with MDE experts, and the evaluation of rules on UML models. Our goal is to identify and validate which consistency rules for all the UML diagrams have been proposed in literature and to create a consolidated set of UML consistency rules. This contrasts with previous reviews that had very different goals, such as focusing on a small subset of UML diagrams, inconsistency management and consistency checking techniques.

Another difference is that all but one piece [6] of work performed an informal literature review or comparison with no defined research question, no precise repeatable search process and no defined data extraction or data analysis process. Our work, however, follows a systematic procedure inspired by a strict, well-known protocol [31-33].

Finally, all the five-works presented above are outdated when compared with the study reported in this manuscript.

In summary, we were unable to find answers to the two main research goals (Section 1.3), which confirmed the need for the systematic identification and validation of a consolidated set of UML consistency rules.

Other related works about other specific research steps carried out in this Ph.D. thesis are presented in the next chapters.

3. WHAT ARE THE UML CONSISTENCY RULES IN THE LITERATURE? (G1)

In this chapter, we describe the procedure identified to answer the main research goal 1 (G1) presented in Section 1.3. To answer G1 we devised a procedure made of two major steps:

1. Performing a literature review to collect the current state of the art in terms of UML consistency rules (see Section 3.1).
2. Starting from the results obtained by the previous step: identifying a consolidated set of UML consistency rules (see Section 3.2).

As reported in Chapter 2, since no systematic identification exists about UML consistency rules, a Systematic Mapping Study (SMS) [11] was conducted for the first step above and its results are presented in Section 3.1. Step 1 returned an initial set of 621 UML consistency rules, published in literature and extracted from seven scientific research databases (see Section 3.1). Then in step 2, a set of 116 UML consistency rules were eventually selected and analyzed in depth, by following a precise selection protocol driven by six research questions (see Section 3.2).

The results of the two steps procedure discussed above will be a consolidated list of unique UML consistency rules that complement the well-formedness rules of the UML standard, and that can be considered to be as complete as possible, thanks to our systematic procedure. Finally, this set of UML consistency rules is validated (G2) in Chapter 4.

To carry out the two steps, we followed the guidelines of Kitchenham and Charters [31]. These guidelines are admittedly for Systematic Literature Reviews. However, they can be readily applied when conducting an SMS, and have been applied by other to that end (e.g., [34-36]). We chose the SMS protocol to carry out both steps because it is a systematic research method that provides an objective procedure for identifying the quantity of existing research related to a research question [37]. A significant amount of space in the Section 3.1 and 3.2 has been used to discuss how these guidelines were followed, since we felt that it was of the upmost importance to allow readers to

precisely understand the results and conclusions and to allow others to replicate our study or compare our results with others.

In addition to the two steps mentioned above, in Appendix C we present the result of a systematic mapping study, where we collected UML consistency rules presented in UML synthesis techniques works. This work was not included because the results were obtained late during the course of the thesis and published only in 2018 [27] and so the rules were not validated (G2) and therefore not included in the main body of this Ph.D. thesis.

3.1 A Systematic Mapping Study of UML consistency rules

This Section presents all the steps to carry out the SMS about UML consistency rules. We provide a description of the SMS protocol we followed [31]: the SMS planning (Section 3.1.1), the SMS execution (Section 3.1.2), and the results (Section 3.1.3). A preliminary discussion with the main findings is provided in Section 3.1.4. Finally, Section 3.1.5 draws the conclusions for this SMS. The related work for the SMS presented in this Section can be found in Chapter 2.

3.1.1 Planning the Mapping Study

In the following four sub-Sections we present the main components of the protocol required to carry out a SMS [31].

3.1.1.1 Research Questions

The underlying motivation for the research questions was to determine the current state of the art about UML consistency rules and this guided the design of the review process. In order to identify the current state of the art on UML consistency rules, we considered seven research questions (RQs): Table 2.

Table 2. Research questions

Research questions	Main motivation
RQ1: What are the UML versions used by researchers in the approaches found?	To discover what UML versions are used in the approaches that handle the UML consistency.
RQ2*:* Which types of UML diagrams have been tackled in each approach found?	To discover the UML diagrams that research has focused upon, to reveal the UML diagrams that are considered more important than others, as well as to identify opportunities for further research.
RQ3: What are the UML consistency rules to check?	To find the UML consistency rules to check and to assess the state of the field.
RQ4: Which types of consistency problems have been tackled in the rules found?	To find the types of consistency problems tackled in the rules. The data found are categorized into three consistency dimensions split into three sub-dimensions: 1) horizontal, vertical and evolution consistency; 2) syntactic and semantic consistency; 3) observation and invocation consistency.
RQ5: Which research type facets are used in research on UML model consistency?	To determine if the field is generally more applied or more basic research as well as to identify opportunities for future research. The papers found were categorized into six types: evaluation research, validation research proposal of solution, philosophical papers, opinion papers and personal experience papers.
RQ6: Is the approach presented automatic, manual or semi-automatic?	To discover how the approaches to check the UML consistency are implemented, in other word if their check system is presented with an automatic, manual or semi-automatic way.
RQ7: How the UML consistency rules are specified? How the UML consistency rules are checked?	To discover how the consistency rules to check the consistency of the UML diagrams are specified (e.g., Plain English, OCL, Promela, etc.) and to discover with which tools those consistency rules are checked (e.g., SPIN, OCL-Checker, etc.).

3.1.1.2 Search strategy

Conducting a search for primary studies requires the identification of search strings (SS), and the specification of the parts of primary studies in which the search strings can be looked for in the search fields: abstract, title and keywords. According to the standard terminology in empirical software engineering, the set of research papers finally retained is referred to as the primary studies [31]. To identify our SS, we followed the procedure of Brereton et al [38]:

1. Define the major terms;
2. Identify alternative spellings, synonyms or related terms for major terms;
3. Check the keywords in any relevant papers already available;
4. Use the Boolean OR to incorporate alternative spellings, synonyms or related terms;
5. Use the Boolean AND to link the major terms.

3.1.1.2.1 Process of the SS selection

In the selection of the search string (SS), we considered various alternatives. For example the search string used in the SLR on consistency management [6] was discarded due to the fact that it might not strictly focus on UML consistency rules: we are much more interested in collecting rules than in identifying consistency management issues and solutions (this is the main reason why we obtain a different set of primary studies.) Other SSs were experimented and discussed in Appendix E. From the set of alternative SSs presented in Appendix E, we selected the following one as it allowed us to retrieve the largest number of useful papers, i.e., the largest number of papers focusing on UML consistency:

((uml OR unified modeling language OR unified modelling language) AND (consistency OR inconsistency))

We did not establish any restriction on publication years up to December 12, 2012. The SMS process started in September 2012 and was totally finished on October 2013.

We used the above-mentioned search string with the following seven search engines (partially suggested by Kitchenham and Charters [31]): IEEE Digital Library, Science Direct, ACM Digital Library, Scopus, Springer Link, Google Scholar, and WILEY. The searches were limited to the following search fields: title, keywords and abstract.

3.1.1.3 Selection procedure and inclusion and exclusion criteria

In this Section we discuss the inclusion and exclusion criteria we used to assess the adequacy of each potential primary study [31]. Since a systematic, automated search in databases based on a search string typically returns papers that cannot be used to answer a set of research questions of interest in an SMS (e.g., in our case, a paper that mentions UML consistency rules), the set of papers collected from the search needs to be trimmed in a systematic manner. This is typically done by relying on the so-called inclusion and exclusion criteria. Inclusion and exclusion criteria are based on the research questions and are piloted to ensure that they can be reliably interpreted and that they classify studies correctly [31]. In this Section, we discuss the inclusion and exclusion criteria used. We then discuss the process followed to include research papers in this SMS.

The inclusion criteria were:

- Electronic Papers focusing on UML diagrams consistency which contained at least one UML consistency rule;
- Electronic Papers written in the English language;
- Electronic Papers published in peer-reviewed journals, international conferences and workshops;
- Electronic Papers published up to December 12, 2012.
- Electronic Papers which proposed UML consistency rules that apply to restrictions/simplifications (or extensions) of the UML notation that do not necessarily strictly follow the OMG standard [9].

The exclusion criteria were:

- Electronic Papers not focusing on UML diagrams consistency;
- Electronic Papers which did not present a full-text paper (title, abstract, complete body of the article and references) but were reduced to an abstract for instance;
- Electronic Papers focusing on UML diagrams consistency which did not contain at least one UML consistency rule;
- Duplicated Electronic Papers (e.g., returned by different search engines);
- Electronic Papers which discussed consistency rules between UML diagrams and other, non-UML sources of data, such as requirements or source code.

  Note that, regardless of the origin of the collected consistency rules, all rules were eventually placed in the same bag and processed in a similar fashion.

3.1.1.4 Data extraction strategy

We extracted the data from the primary studies according to a number of criteria, which were directly derived from the research questions detailed in Table 2. Using each criterion to extract the data required from the full-text of the 95 primary studies. We collected the data in an Excel spreadsheet that represent our data form. From each primary study the following information was extracted and collected into the Excel data form:

- Search engines: where the paper was found (see Section 3.1.1.2.1);
- Inclusion and Exclusion Criteria applied (see Section 3.1.1.3);
- Data related to Research Questions (see Section 3.1.1.1):
  - What UML version was used;
  - What are the UML consistency rules discussed (see our online technical report [39]);
  - What diagrams are involved in consistency rules: 1) Class Diagram (CD); 2) Communication Diagram (COMD) or Collaboration Diagram (COD); 3) Use Case Diagram (UCD); 4) State Machine Diagram (SMD) or Protocol State Machine Diagram (PSMD); 5) Sequence Diagram (SD); 6) Component Diagram (CTD); 7) Object Diagram (OD); 8) Profile Diagram(PD); 9) Activity Diagram (AD); 10) Composite Structure Diagram (CSD); 11) Interaction Overview Diagram (IOD); 12) Package Diagram (PD); 13) Timing Diagram (TD); 14) Deployment Diagram (DD).

  UML terminology has changed over the years, and the following pair of UML diagrams were therefore considered and counted (in the statistical reports presented in this research) as only one diagram:

    - COD and COMD (diagram 2) were counted as one diagram since COMD is the UML 2.x equivalent of the COD in UML 1.x [46].
    - PSMD is a kind of SMD (diagram number 4 in the list above) [9]. Nevertheless, we initially decided to collect the information for these two diagrams separately since they have different purposes during software development, but both diagrams were eventually reported as only one diagram.

- Which are the types of UML consistency dimensions. Several possible dimensions of consistency appear in the literature, since consistency can be discussed from different perspectives [40]:
  - Horizontal, Vertical and Evolution Consistency: *Horizontal consistency*, also called intra-model consistency, refers to consistency within a model or between different diagrams of the model at the same level of abstraction, and within the same version [15]. *Vertical consistency*, also called inter-model consistency, refers to consistency between models (and

therefore their diagrams) at different levels of abstraction (e.g., analysis vs. design) [41]. *Evolution consistency* refers to consistency between different versions of the same model (and therefore their diagrams), and has to be maintained when the model is in the process of evolution [15].

- Syntactic versus Semantic consistency: *Syntactic consistency* ensures that a specification conforms to the abstract syntax specified by the meta-model, and requires that the overall model has to be well formed [41]. *Semantic consistency* requires that the behavior of diagrams be semantically compatible [41]. Note that this does not mean that semantic consistency is necessarily restricted to behavioral diagrams. For instance, operation contracts available in the class diagram specify behavior. Semantic consistency applies at one level of abstraction (horizontal consistency), at different levels of abstraction (vertical consistency), and during model evolution (evolution consistency) [30].
- Observation versus Invocation consistency: *Observation consistency* requires that an instance of a subclass behave like an instance of its superclass, when viewed according to the superclass description [42]. In terms of UML state chart diagrams (corresponding to protocol state machines) this can be rephrased as "after hiding all new events, each sequence of the subclass state chart diagram should be contained in the set of sequences of the superclass state chart diagram." *Invocation consistency* requires that an instance of a subclass of a parent class can be used wherever an instance of the parent is required [42]. In terms of UML state chart diagrams (corresponding to protocol state machines), each sequence of transitions of the superclass state chart diagram should be contained in the set of sequences of transitions of the state chart diagram for the subclass.
- Tool support (Automatic, Semi-Automatic, Manual);
  - Automatic means that a tool automatically checks the UML consistency rules with no human intervention;
  - Semi-automatic means that checking the UML consistency rules were partially automated (for instance when the check of a UML model needs the support of user to finish the process);

- Manual means that the UML consistency rules were not supported by any implemented and automatic tool.

- What mechanisms were used to specify the rules: e.g., plain language, OCL, Promela;
- How the UML consistency rules are checked: e.g., SPIN, OCL-Checker;
- Research type facet followed in the paper, for which we used the following classification [43]:
  - Evaluation research (ER): this is a paper that investigates techniques that are implemented in practice and an evaluation of the technique is conducted. That means, the paper shows how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation).
  - Proposal of solution (PS): this is a paper that proposes a solution to a problem and argues for its relevance, without a full-blown validation.
  - Validation Research (VR): this is a paper that investigates the properties of a solution that has not yet been implemented in practice.
  - Philosophical papers (PP): this is a paper that sketches a new way of looking at things, a new conceptual framework, etc.
  - Opinion papers (OP): this is a paper that contains the author's opinion about what is wrong or good about something, how something should be done, etc.
  - Personal experience papers (PEP): this is a paper that emphasizes more on what and not on why.

3.1.2 Execution

The planning for this SMS with the seven search engines begun in September 2012 and was completed on December 12, 2012. In this Section, we present the execution of the search with the seven search engines and the selection of primary studies according to the inclusion/exclusion criteria previously described. In order to document the review process with sufficient details [31], we describe the multi-phase process of four sub-phases we followed:

- **First sub-phase**: the search string was used to search with the seven search engines as mentioned earlier.
- **Second sub-phase**: we deleted duplicates automatically, by using the RefWorks tool[5]; we also removed some duplicates manually.
- **Third sub-phase**: we obtained an initial set of studies by reading the title, abstract and keywords of all the papers obtained after the second sub-phase while enforcing the inclusion and exclusion criteria. When reading just the title, abstract and keywords of a paper was not enough to decide to include or exclude it, we checked the full-text.
- **Fourth sub-phase**: all the papers identified in the third sub-phase were read in their entirety and the exclusion criteria were applied again. This resulted in the final set of primary studies.

Table 3. Summary of primary studies selection on 12/12/2012

Sub phase	IEEE	Scopus	Springer Link	Google Scholar	WILEY	ACM	Science Direct	Total
First sub-phase: Raw results	363	601	163	341	9	87	39	1603
Second sub-phase: No duplicates	279	325	159	247	9	80	36	1135
Third sub-phase: First selection	62	64	62	28	4	33	14	267
Fourth sub-phase: Primary studies	16	21	21	12	1	16	8	**95**

Table 3 breaks down the number of papers we have found by sub-phases. The first sub-phase (first row) shows the initial results which were obtained by running the search into the seven search engines selected. The next two rows show the results obtained after applying the second and third sub-phases of the studies selection process. In the end, we collected 95 primary studies for further analysis. Appendix D lists the references of the 95 primary study papers [PS1-PS95] obtained with this SMS.

[5] http://www.refworks.com/2014

3.1.3 Results

To reach the goal of this SMS, i.e., addressing the research questions listed in Section 3.1.1.1, the 95 primary studies were classified according to the criteria detailed in Section 3.1.1.4, then the results of the SMS reported in this Section show the answers to the seven research questions previously presented. A quantitative summary of the results for research questions RQ1, RQ2, RQ4, RQ5 and RQ6 is presented Table 4. More details are provided in the following sub-Sections.

Table 4. Results of SMS

Research question	Possible Answer	Result	
		# Papers	Percent
RQ1**:** UML versions	UML 1.1	1	1.05%
	UML 1.3	13	13.68%
	UML 1.4	6	6.32%
	UML 1.5	8	8.42%
	UML 2.0	31	32.63%
	UML 2.1.X	10	10.53%
	UML 2.2	2	2.11%
	UML 2.3	1	1.05%
	UML 2.4.1	1	1.05%
	NF	19	20.00%
	Ext.	1	1.05%
	Sim.	2	2.11%
RQ2**:** UML diagrams	Class Diagram	68	71.58%
	State Chart Diagram	40	42.11%
	Protocol State Machine Diagram	5	5.26%
	Sequence Diagram	45	47.37%
	Collaboration Diagram	8	8.42%
	Activity Diagram	12	12.63%
	Use Case Diagram	14	14.74%
	Object Diagram	4	4.21%
	Communication Diagram	2	2.11%
	Composite Structure Diagram	1	1.05%
	Interaction Diagram	4	4.21%
RQ5**:** Research type facets	ER	16	16.84%
	VR	29	30.53%
	PS	47	49.47%
	PP	0	0.00%
	OP	0	0.00%
	PEP	3	3.16%
RQ6**:** Type of support	Automatic	24	25.26%
	Semi-Automatic	29	30.53%
	Manual	42	44.21%

3.1.3.1 UML version (RQ1)

Figure 4 plots the number of papers presenting rules for specific versions of the UML. The presence of 29.47% (28 of 95 papers) of the primary studies with an old version (1.x) of the UML

shows that the issue of UML consistency rules started to be relevant from the initial launch of the UML (which has been evolving since the second half of the 1990s [9]). UML 2.0 is the UML version mostly used in the primary studies: 32.63% (31 of 95 papers).

The subsequent UML versions (2.1, 2.1.1 and 2.1.2) were merged into 2.1.X to obtain a more readable graph. NF means "not found" and represents all those primary studies which did not specifically report on the UML version used and for which we were not able to determine the UML version by reading the text. "Ext." and "Sim." represent primary studies which use an extension or simplification of the UML notation that do not strictly follow the UML standard [6].

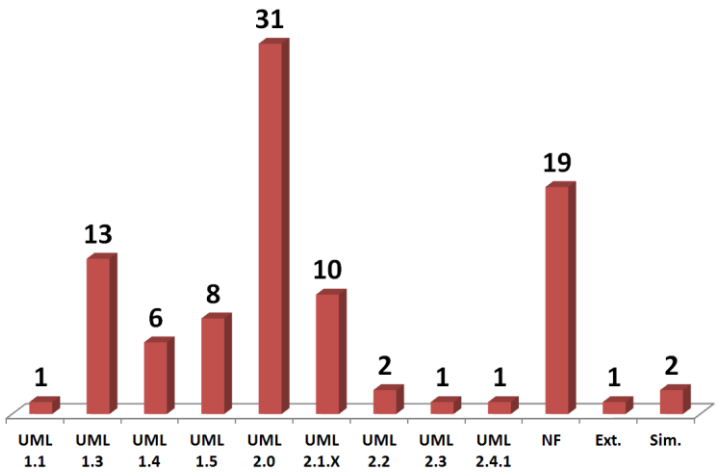

Figure 4. UML version

3.1.3.2 Types of UML diagrams (RQ2)

In this Section, we discuss the different types of UML diagrams involved in primary studies. Figure 5 indicates that collected rules describe consistency on only eleven of the 14 UML diagrams. (We did not collect any rule involving the timing, interaction overview and deployment diagrams, and these diagrams therefore do not appear in the Figure.)

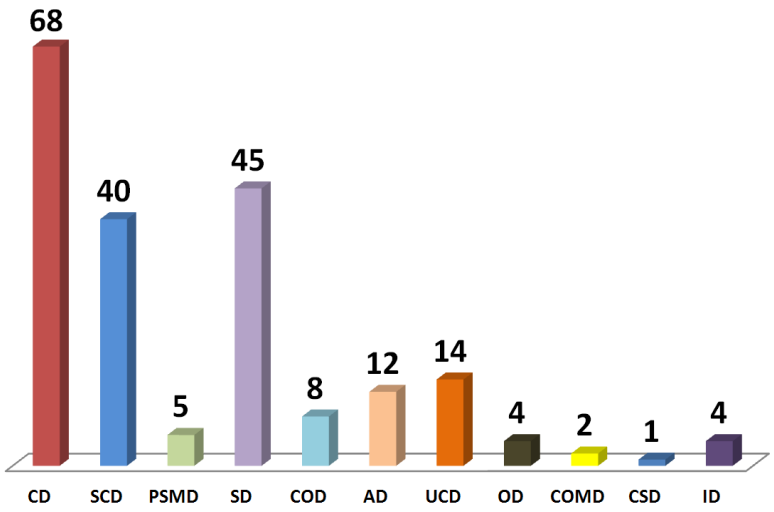

Figure 5. Number of primary studies with rules involving each UML diagram

Not surprisingly, since these are the most used diagrams [44], the Class Diagram (71.58%, 68 out of 95 primary studies), the Sequence Diagram (47.37%), and the State Machine Diagram (42.11%) are the diagrams mostly involved in consistency rules. Research on UML consistency rules has placed much less attention on the Use Case Diagram (14.74%) and the Activity Diagram (12.63%). The Collaboration Diagram was found in 8.42% of the primary studies. The least used diagrams are the Protocol State Machine Diagram, the Object Diagram, the Interaction Diagram, the Communication Diagram and the Composite Structure Diagram.

3.1.3.3 UML consistency rules (RQ3)

The principal aspect shown in this research question is that researchers interested in UML consistency rules typically define a number of similar consistency rules over and over again. Specifically, we collected an initial set of 621 UML consistency rules from the primary studies that will be further discussed in Section 3.2. We describe below a few representative rules, illustrating some of the consistency dimensions (Section 3.1.1.4): the complete list of rules can be found in Appendix A. Table 5 presents rule 20 which is a Horizontal-Semantic consistency rule. It checks UML consistency between the Sequence Diagram and the Use Case Diagram.

Table 5 UML consistency rule between Sequence Diagram and Use Case Diagram

20	A use case is complemented by a set of sequence diagrams, each sequence diagram representing an alternative scenario of the use case. (*Horizontal and Semantic*).

Table 6 presents rules 67 and 69, which are respectively Horizontal-Semantic and Vertical-Syntactic rules. They check UML consistency of the Class Diagram.

Table 6 UML consistency rules of Class Diagram

67	A class diagram is consistent if it can be instantiated without violating any of the constraints in the diagram (e.g., association end multiplicities). (*Horizontal and Syntactic*).
69	A class relationship at a (low) level of abstraction must have an abstraction at a higher level of abstraction, either a class or a relation. (*Vertical and Semantic*).

We acknowledge that even though rule 67 could be considered a semantic rule, in this manuscript we categorized it as syntactic because we believe that, checking association end multiplicities as specified in the class diagram, it is a syntactic aspect of a model. We reserve the category of semantic for the rules that check behavioural aspects of a model.

3.1.3.4 UML consistency dimensions (RQ4)

This sub-Section presents the results about the number of UML consistency rules divided into the UML consistency dimensions presented in Section 3.1.1.4.

First of all, regarding the several dimensions that can be used to classify consistency rules (Section 3.1.1.4), we note that 69.47% (66 of 95 papers) of the primary studies did not mention any such dimension, that 17.89% (17 of 95 papers) presented horizontal and vertical consistency rules, and only 4.21% mentioned also evolution consistency with those two dimensions.

The results show that the great majority of the primary studies presented UML consistency rules that involved Horizontal and Syntactic rules. Researchers described strikingly many more syntactic than semantic consistency rules. Also, although we have not yet, at this stage of this manuscript text, compared the set of 621 rules with well-formedness rules of the UML standard, we suspected that a large majority of the syntactic consistency rules we collected are already in the UML standard: for instance, several authors present the rule whereby a class cannot be a descendant (or ancestor) of itself in a class diagram, which is already a constraint of the UML metamodel.

3.1.3.5 Research type facets (RQ5)

The results of the research type facet classification show that 49.47% (47 of 95 papers) of primary studies proposed solutions to the inconsistency problem, 30.53% (29 of 95 papers) presented validation research, 16.84% (16 of 95 papers) presented evaluation research, and only 3.16% (3 of 95 papers) presented personal experience. We did not find any philosophical paper nor opinion paper (0% for both). This suggests the field is about problem solving and requires more evaluations of the consistency rules than what have been proposed.

3.1.3.6 Tool support (RQ6)

The UML consistency rules presented by researchers are supported by automatic tools (25.26%, 24 of 95 papers), semi-automatic tools (30.53%, 29 of 95 papers), and finally the larger number of publications presented manual verification (44.21%, 42 of 95 papers).

3.1.3.7 UML consistency rules: specification and support (RQ7)

Figure 6 shows that plain English (29.47%) is the most used language to specify UML consistency rules, followed by the Object Constraint Language (OCL) [9] (22.11%), Communicating Sequential Processes (CSP) and Promela (5.26% each). Using OCL makes sense since this is a constraint language that is part of the UML; in addition, it is mostly used in syntactic rules. Languages such as CSP and Promela have been used to specify semantic rules between the sequence diagram and the state machine diagram. The category "other" in Figure 6 summarizes all those proposals (23.16%, 22 of 95 papers) that present a specification mechanism that appears in only one primary study (for instance XML Equivalent Transformation, Prolog, Constraint Logic Programming, etc.).

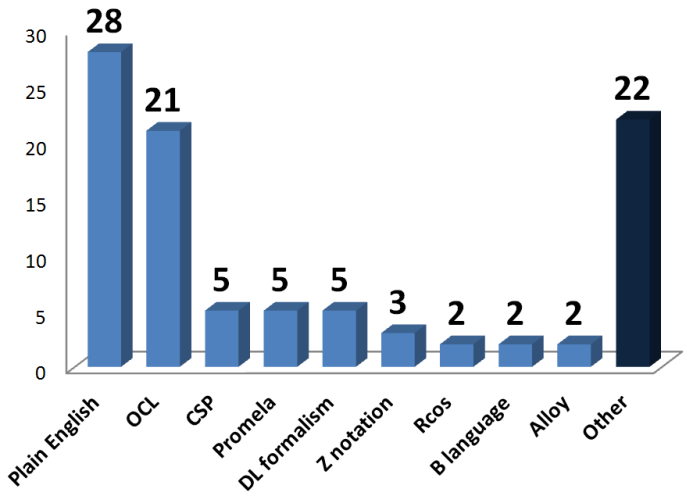

Figure 6. Number of primary studies using a particular language for specifying UML consistency rules

The majority of the papers (55.79%) presented tool support to check UML consistency rules: Figure 7. IBM Rational Rose is the most used tool (10.53%), followed by Spin, UML/Analyzer and OCL interpreters, the last two contributing respectively 6.32% and 5.26% of the total. The category "other" in Figure 7 summarizes all those primary studies that present a UML consistency checking tool that is used in only one primary study (e.g., Poseidon, ArgoUML, etc.). 44.21% of the primary studies did not present any tool support (in Figure 7 "NI" means not implemented) for their UML consistency rules.

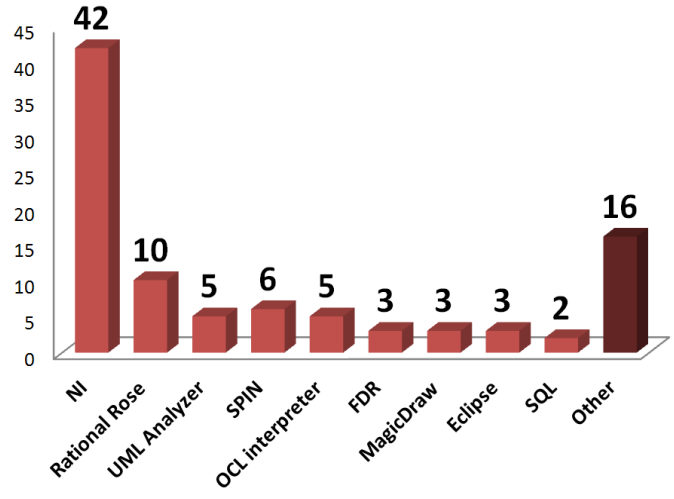

Figure 7. Number of primary studies using a tool to check UML consistency rules

3.1.3.8 Additional results

Table 7 shows the publication venues with the largest number of primary studies on UML consistency rules.

Table 7. Publication venues ranking

Publication	#	Perc.	Venue
International conference on Model driven engineering languages and systems (MODELS)	5	5.26%	Conference
IEEE/ACM International Conference on Automated Software Engineering (ASE)	5	5.26%	Conference
International Conference on Conceptual Modeling (ER)	5	5.26%	Conference
Australian Conference on Software Engineering (ASWEC)	4	4.21%	Conference
Electronic Notes in Theoretical Computer Science	4	4.21%	Magazine
International Conference on Software Engineering (ICSE)	4	4.21%	Conference
IEEE Transactions on Software Engineering	2	2.11%	Journal
Proceedings of the IEEE Region 10 (TENCON)	2	2.11%	Proceeding
IEEE International Conference on Software Engineering and Formal Methods (SEFM)	2	2.11%	Conference
ACM symposium on Applied computing (SAC)	2	2.11%	Conference
International Conference on Computer Systems and Technologies (CompSysTech)	2	2.11%	Conference
ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)	2	2.11%	Conference

The first three have five papers each, together representing 15.79% (15 papers) of the total. The next three, all of them with four papers together represent 12.63% (12 papers) of the total.

The distribution per year of the 95 primary studies is shown in Figure 8. Between 2003 and 2010, the number of publications remained relatively stable from 7 to 13 articles a year, except in 2004 and 2009. We also notice that the release of UML 2.0 in 2005 did not impact numbers much. All of this suggests that the topic of UML diagrams consistency remains important to the research community. The number of publications decreased in 2012, but it is likely due to the fact that many papers published in that year were not yet available at the time we performed searches. In addition, this downward trend, which started in 2009, may be due to the increased popularity of DSLs (as discussed in Section 4.2.2.2.8). Many DSLs are however based on, or custom versions of the UML.

So, although this should be studied in future, we believe a good part of our results (i.e. the set of UML consistency rules presented in the next Sections) would apply to DSL contexts as well.

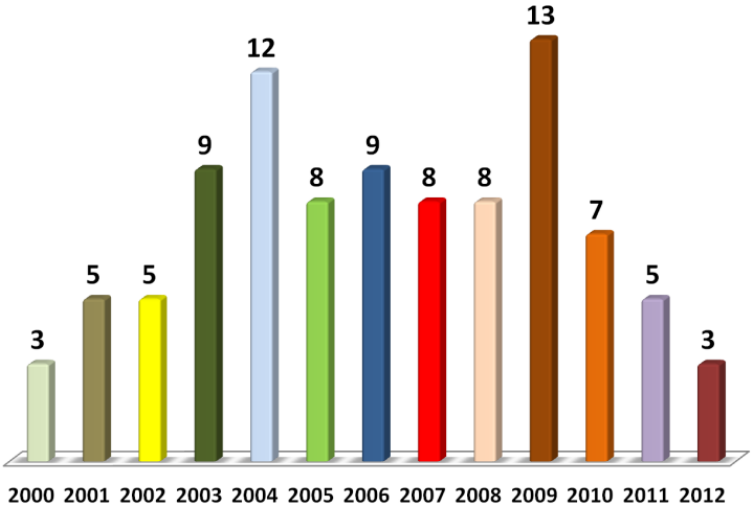

Figure 8. Number of primary studies per year

3.1.4 Discussion

The following sub-Sections describe the analysis of the results for RQ1 to RQ6, defining bubble plots in order to report the frequencies of combining the results from different research questions. A bubble plot is basically two x–y scatter plots with bubbles in category intersections. This synthesis method is useful to provide a map and give a quick overview of a research field [45].

3.1.4.1 Combining RQ1, RQ2 and RQ5

Combining the results of RQ1, RQ2 and RQ5, we obtained (Figure 9) the mapping of the research type facets being used in a primary study to the version of the UML involved and the type of UML diagrams. In the same way, the different UML versions are shown according to the UML diagrams. The results about the UML versions show that with 23 proposals, the Class Diagram is the most used UML diagram with UML version 2.0. It is closely followed by the Sequence Diagram with 20 papers in the same UML version. This is an observation that we can consistently make across UML versions. Proposals which used State Chart Diagrams were constant (in numbers) between UML versions 1.3 and 2.1.X; in fact, the number of publications remained relatively stable from 4 to 9 articles for version reaching its peak with 9 articles for the UML version 2.0. Little has been

proposed for UML versions 2.2 and 2.3, perhaps because of the small changes to the metamodel from UML 2.1.

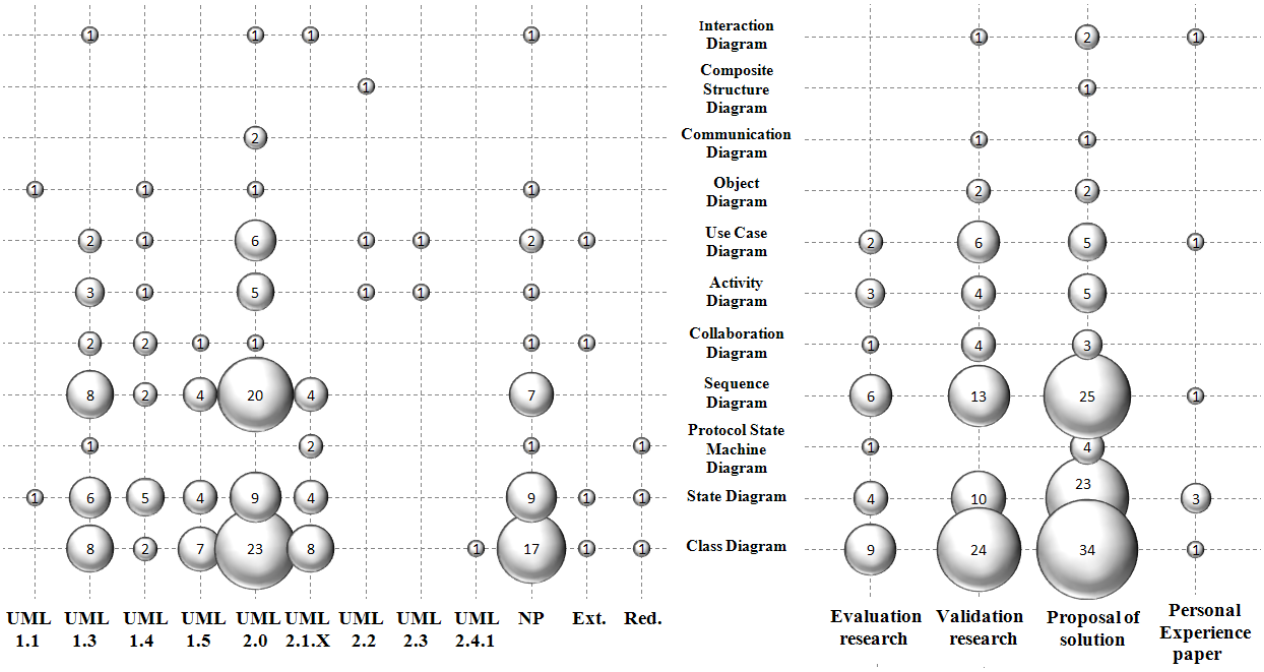

Figure 9. Combining RQ1, RQ2 and RQ5

As shown in Figure 9, most of the primary studies which present a Proposal of Solution paper, present rules that involve the Class Diagram, Sequence Diagram and State Chart Diagram, respectively with 72.34% (34 papers), 53.19% (25 papers) and 48.94% (23 papers) of the 47 Proposal of Solution primary studies. It is important to note that the vast majority of the primary studies (24 of 29, 82.76%) that perform some validation (classified as Validation Research) focus on the class diagram. We also observe an imbalance of the ratio Evaluation Research + Validation Research primary studies (evaluation and validation) over Proposal of Solution primary studies for the three main diagrams: i.e., we see a higher (Evaluation Research + Validation Research / Proposal of Solution) for the class diagram (97%) than for the sequence diagram and state machine diagram (76% and 60.8%, respectively).

3.1.4.2 Combining RQ6 and RQ7

As shown earlier, most of the studies about UML consistency rules did not present any UML CASE tool to support those rules. In fact, this aspect is confirmed by the fact that, plain English is

the language mostly used to specify UML consistency rules. There is still not one UML tool used by researchers that can be considered standard to execute UML consistency rules.

3.1.5 Conclusions of Section 3.1

In recent years, a great number of UML consistency rules have been presented by researchers to fix inconsistencies between UML diagrams. However, no mapping study exists that summarizes these UML consistency rules since the majority of studies are informal literature surveys.

This work presented the results obtained after carrying out a Systematic Mapping Study of literature with the aim to identify and evaluate the current state of the art about UML consistency rules. The SMS was carried out following well-known guidelines [31]. A significant amount of space in this chapter has been used to discuss how these guidelines were followed, since we felt that it was of the upmost importance to allow readers to precisely understand the results and conclusions and to allow others to replicate our study or compare our results with others. From an initial set of 1134 papers, a total of 95 primary studies were found by following a precise selection protocol driven by seven research questions. Primary studies were then classified according to several criteria, also derived from those research questions. In addition, we collected an initial set of 621 UML consistency rules from the primary studies that will be further discussed in Section 3.2.

3.2 A systematic identification of UML consistency rules

In an attempt to obtain an accurate picture of current research in the area of UML consistency rules, we conducted a systematic mapping presented in Section 3.1, which resulted in a number of findings concerning publications in the domain. Although the aim of this mapping study was to discover frequencies of publications in different categories and was not specifically to study published UML diagram consistency rules, the study provided anecdotal evidence that: (1) authors routinely define consistency rules for different UML based software engineering activities; (2) some authors tend to define similar, often identical rules, over and over again, and (3) some authors are not aware that the UML standard specification defines consistency rules (a.k.a., well-formedness rules). We also observed that even though many researchers have either explicitly or

implicitly proposed rules with which to detect inconsistencies in the UML, no consolidated set of UML consistency rules had been published to date. By the word "consolidated", we mean the process of combining a number of UML consistency rule sets that authors feel are important for UML into a single, coherent set. We believe that, even though the rules required may be domain, organization, or project specific, this lack of a consolidated list of UML consistency rules forces researchers to define the rules that they rely on for their own research [12], thus resulting in some rules being defined over and over again. This motivated this second step, which is to identify a consolidated set of consistency rules for UML diagrams. Identifying such a set of UML consistency rules has several benefits: it will be a reference for practitioners, educators, and researchers alike; it will provide guidance as to which rules to use in each context; and it will highlight which areas are more developed and which need further work.

In order to identify such a consolidated set of UML Consistency rules, we revisited the primary studies obtained in our systematic mapping study presented in Section 3.1, which led us to collect and analyze the rules from the publications in question. This was done by following a systematic procedure inspired by well-known guidelines for empirical research in software engineering [31].

Of the 95 primary studies identified during the mapping study in Section 3.1, we collected 621 consistency rules, including duplicates and rules already in the UML standard, which we then analyzed in order to answer a number of research questions:

RQ1: What are the existing UML consistency rules?

RQ2: Which types of consistency problems are tackled in the existing rules?

RQ3: What types of UML diagrams are involved in UML consistency rules?

RQ4: For what software engineering activities are UML consistency rules used?

RQ5: Which UML diagram elements are involved in UML consistency rules?

We subsequently employed a systematic refinement process with the objective of removing duplicates and rules already in the UML standard, as explained later in this chapter, and we eventually compiled a consolidated set of 116 rules.

The next Sections are structured as follows: Section 3.2.1 provides the definition of the systematic procedure that we followed. Section 3.2.2 presents the execution of the study; the results are described in Section 3.2.3; and finally, our conclusions are shown in Section 3.2.4. The related

work for the systematic identification of UML consistency rules presented in this Section can be found in Chapter 2.

3.2.1 Definition of the Systematic Procedure

In this Section, we present the procedure we followed, inspired by a well-accepted systematic search and analysis process [31], to obtain our consolidated set of UML consistency rules, starting from the initial set of 621 UML consistency rules collected from the 95 primary studies (Section 3.1). From here on, until the end of this chapter, UML consistency rules will be referred to simply as rules. We present the main components of the planning of our work, which are: the specification of research questions that the study aim to answer (Section 3.2.1.1); the procedure followed to select (or reject) rules to be part of our final consolidated set of rules (Section 3.2.1.2); and the procedure used to extract data from the rules selected in order to answer the research questions (Section 3.2.1.3).

3.2.1.1 Research questions

The underlying motivation for the research questions was to analyze the state-of-the-art regarding rules. In order to do this, we considered five research questions (RQs): Table 8.

Table 8. Research questions

Research questions	Main motivation
RQ1: What are the existing UML consistency rules?	To find the UML consistency rules used in literature in order to assess the state of the field.
RQ2: Which types of consistency problems are tackled in the existing rules?	To find the types of consistency problems tackled in the UML consistency rules: 1) horizontal, vertical and evolution consistency; 2) syntactic and semantic consistency; 3) observation and invocation consistency; and assess the state of the field as regards that dimension of UML diagram consistency.
RQ3: What types of UML diagrams are involved in UML consistency rules?	To discover the UML diagrams that research and practice has focused on, to reveal the UML diagrams that are considered more important than others, and to identify opportunities for further research.
RQ4: For what software engineering activities are UML consistency rules used?	To find the different software engineering activities that require UML diagrams to be consistent according to specific sets of consistency rules, in order to understand what they should focus on.
RQ5: Which UML diagram elements are involved in UML consistency rules?	To find the UML diagram elements that are most frequently tackled in the UML consistency rules.

Three of those five research questions (specifically, RQs 1, 2, and 3) had already been used in the previous Section 3.1. The main differences between the three RQs in this Section and those in Section 3.1 lies in how and for what they are used:

RQ1: In Section 3.1 we did not present any rule and solely discussed metadata about the rules, while in this Section we present the consolidated set of rules and provide a detailed analysis of the rules.

RQ2: Unlike Section 3.1, we now filter out the rules we collected that are already in the UML standard.

RQ3: In Section 3.1, the focus of the study was a series of papers and we therefore studied how UML diagrams are involved in papers that present consistency rules, whereas in this Section the focus are the rules and we therefore study how UML diagrams are involved in rules.

The other two research questions (RQs 4 and 5) are new to this chapter.

3.2.1.2 Inclusion and exclusion criteria

In this Section, we discuss the inclusion and exclusion criteria used to refine the set of rules. Since we used the 621 rules found in Section 3.1 as a starting point, and those rules had already been obtained through the use of inclusion and exclusion criteria, we only applied additional exclusion criteria:

- Inaccurate rules:
  - Rules that we considered wrong. For instance, one rule specified that each message in a sequence diagram should trigger a transition in a state machine diagram. We believe that this is incorrect since some of the classifiers receiving a message in a sequence diagram may not have a state-based behavior, and even if they have, not all messages need to trigger state changes;
  - Rules not focusing on UML diagram consistency. For instance, rules which discussed consistency between UML diagrams and other non-UML data, such as requirements or source code, were discarded;

- Rules that were considered obsolete because they focus on UML characteristics that are no longer supported in the latest UML standard (i.e. UML 2.5);

- Redundant rules, i.e., all those rules that can be implied by other rules;
- Rules that were already included in previous UML standard versions.

3.2.1.3 Data extraction strategy

We answered the research questions (Table 8) by extracting data from the collected rules, and recording that data on an Excel spreadsheet (our data extraction form) presented in Table 9.

Table 9. Data extraction form

Extraction form – UML consistency rules				
Paper	Reference		Year of publication	
Rule	Definition			
	Type		Dimension	
	UML diagram(s) involved		SE activity involved	
Reason for exclusion (if applicable)				

For each rule, we collected the following data:

- Reason for excluding the rule, as per the exclusion criteria (see Section 3.2.1.2)
- Precise definition of the rules (see Section 3.2.3 and Appendix A). When the rule, as originally specified by others, was not sufficiently clear or precise, we redefined it, whilst maintaining the original intent as we understood it, thus hopefully facilitating comparisons and the consolidation of rules. We also recorded the papers that introduced each rule, which can, for instance, be used to identify the rules that are the most important to users;
- Which of the 14 UML diagrams are involved in the rule (see Section 3.1.1.4 for UML diagrams).
- UML consistency dimensions and types (see Section 3.1.1.4).
- The software engineering activity that required UML diagrams to be consistent with one another in some way as specified in rules. The following eight definitions describe activities

in software engineering in which UML diagram consistency was required, as explained by some authors. Since this is a list of activities from the primary studies we found in Section 3.1, this list is not meant to exhaustively represent all the software engineering activities that require diagrams to be consistent:

- The verification of consistency (*Verification*) is the process of detecting inconsistencies between (or in a singular) UML diagram(s) during software development. (This definition is in line with the IEEE definition of verification [47].)
- Consistency management (*Management*) includes the detection of inconsistencies in a software model, the diagnosis of the significance of inconsistencies, and the handling of detected inconsistencies [16].
- Model Refinement and Transformation (*Ref. & Tra.*) are defined as follows [48]:
  - Refinement, which is a semantics-preserving transformation applied to a model, and which produces the same kind of model, e.g., refining a Platform Independent Model (PIM) into a new PIM;
  - Transformation (also called mapping), which generates a new kind of model, e.g., transforming a Platform Independent Model into a Platform Specific Model (PSM), or a PSM into executable code. These transformations generally add detail to the model.
- Safety and Security consistency (*Saf. & Sec.*): From the point of view of automated analysis, safety consistency implies that there are no conflicting requirements and unintentional (internal) non-determinism in a UML model [49]. Pilskalns and colleagues define security consistency as the consistency of bounded values in non-abstract elements of a UML diagram [50].
- Impact Analysis (*Impact analysis*) is defined as the process of identifying the potential consequences (side-effects) of a change to the model, and estimating what needs to be modified to accomplish a change [51].
- Model formalization (*Formalization*) describes the formal process used to specify/develop a UML model [52, 53].

- Model understanding (*Understanding*) is the process employed to understand that a UML model is much more than a set of annotated boxes and lines as it reflects the meaning of a software, that is, the UML has semantics. The semantics of the UML is expressed through its metamodel and in particular through both the so-called well-formedness rules and context/domain dependent consistency rules, which describe in plain language and often using the Object Constraint Language (OCL) constraints that UML model elements have to satisfy [28].

3.2.2 Execution

In this Section, we report the results obtained, in terms of number of collected rules, after conducting the protocol discussed in Section 3.2.1. Please recall that we started from the primary studies collected in Section 3.1, which began in September 2012 and was completed in October 2013: the primary studies selected span the period from 2000 to 2012; the list of primary studies can be found in Appendix D. The execution of the rule selection protocol of our SMS began in May 2014 and was completed in at the beginning of 2015.

We then started to study the 95 primary study papers obtained from our SMS [11], which allowed us to identify 621 rules. In order to document the execution activity of the protocol described in Section 3.2.1 in sufficient detail, we describe the following multi-tasks process:

- First task (T1): 621 unique rules were eventually specified.
- Second task (T2): we obtained a reduced set of rules by deleting all the inaccurate ones, according to the exclusion criteria. We identified 197 inaccurate rules and obtained a set of 424 accurate rules.
- Third task (T3): we further reduced the set of rules by deleting all the redundant ones, again according to the exclusion criteria. We identified 242 redundant rules and obtained a set of 182 accurate and non-redundant rules.
- Fourth task (T4): we further reduced the set of rules by deleting those already presented in previous UML standards, again according to the exclusion criteria. We identified 66 rules that were already in the standard (i.e., well-formedness rules) and obtained a set of 116 rules that are accurate, non-redundant and not already in the standard.

The set of 424 accurate rules (T2), and the final set of 116 consistency rules (T4) were then classified according to the data extraction strategy discussed in Section 3.2.1.2. The reason for the selection of these two sets (i.e. T2 and T4) is explained in the next sections.

3.2.3 Results

The data recorded on our data extraction form (an excel file) permits the five research questions shown in Section 3.2.1.1 to be answered. A quantitative summary of the results for research questions RQ2 to RQ5 is presented in Table 10.

More details are provided in the following sub-Sections, in which we shall refer to Table 10 for raw data. Two different set of rules were used to answer our five research questions:

1) The final set of 116 consistency rules (T4 in Section 3.2.2) to answer RQ1, RQ2 and RQ3.
2) The set of 424 accurate rules (T2 in Section 3.2.2) for RQ4 and RQ5. We chose to include the 242 redundant rules (T3 in Section 3.2.2) and the 66 rules already presented in the UML standard (T4 in Section 3.2.2) to answer these two research questions, because we wished to show the whole picture of what researchers have focused on.

Table 10 (first raw) simply refers to Appendix A for research question RQ1 since the purpose of the question is to report on the consistency rules we have coalesced. The Section of Table 10 that contains data for research question RQ3 shows all the combinations of UML diagrams for which we have found consistency rules in the primary studies, and the number of consistency rules for each combination after conducting the protocol discussed in Section 3.2.1 (e.g., after applying inclusion/exclusion criteria): e.g., we have found 14 rules involving only the state machine diagram (SMD), five rules involving the sequence and use case diagrams at the same time (SD and UCD), and one rule involving the class, state machine and activity diagrams at the same time (CD, SMD and AD). Primary studies showed rules for 36 different diagram combinations, i.e., a small subset of all the possible combinations of UML diagrams. Note also that the Table shows combinations with no rule (e.g., UCD and SMD): this means that we found rules in primary studies involving those diagrams, but the rules were eventually discarded because of our include/exclusion criteria.

Table 10. Results at a glance

Research Question 1: What are the existing UML consistency rules? (see Appendix A)											
Research Question 2: Which types of consistency problems are tackled in the existing rules?											
Horizontal	97	Invocation	8	Vertical	7	Evolution	3				
Observation	1	Syntactic	95	Semantic	21	-	-				
Research Question 3: What types of UML diagrams are involved in UML consistency rules?											
1	SMD	14	**10**	UCD and SMD	0	**19**	COMD	1	**28**	CD	33
2	CD, SMD and AD	1	**11**	CD and SMD	7	**20**	UCD and CD	3	**29**	ID and CD	0
3	SD and UCD	5	**12**	PSMD and CD	0	**21**	OD and AD	0	**30**	ID and UCD	2
4	AD	0	**13**	SD and AD	7	**22**	COMD and AD	1	**31**	PSMD and SMD	0
5	SD	6	**14**	UCD and OD	0	**23**	COMD and CD	3	**32**	CSD	8
6	OD and COMD	0	**15**	SMD and COMD	1	**24**	COMD and UCD	0	**33**	SD and CD	9
7	OD and CD	1	**16**	COMD and CD	1	**25**	UCD and AD	4	**34**	SD and COMD	0
8	AD and CD	4	**17**	SMD and SD	1	**26**	UCD	4	**35**	CD, SMD and COMD	0
9	SMD and AD	0	**18**	PSMD	0	**27**	SMD and OD	0	**36**	UCD, SD and CD	0
Research Question 4: For what software engineering activities are UML consistency rules used?											
Verification	220	Ref/ &Tra.	54	Understanding	21	Saf. & Sec.	7				
Impact Analysis	57	Management	51	Formalization	14	-	-				
Research Question 5: Which UML diagram elements are involved in UML consistency rules?											
class/es	161	action/s	18	invariant/s	9	capsule/es	4				
state/s	86	Collaboration	17	interface/s	9	class name	3				
operation/s	56	chart/s	17	Interaction	9	super class	2				
association/s	45	machine/s	17	Sender	8	sub sequence	2				
use case	41	pre-condition	16	Flow	8	sub classes	2				
activity/ies	39	Strings	15	composite	8	scenario	2				
message/s	38	Link	13	property	7	query	2				
sequence/s	38	event/s	12	connector/s	7	post-state	2				
object/s	36	guard/s	12	Call	7	package	2				
instance/s	30	Visibility	11	node	6	swimlanes	1				
name/s	26	Aggregation	11	generalization/s	6	stimulus	1				
transition/s	26	post-condition/s	10	constraint/s	6	stimuli	1				
type/s	23	method/s	10	Port	5	name pace	1				
attribute/s	19	Receiver	10	Parameter	5	edge	1				
Behavior	18	Multiplicity	9	Lifeline	4	actor	1				

The colored column shows the unique ID# we gave to each of the 36 UML diagram combinations; these unique IDs will later be used in Figures to simplify the data analysis; the color code for combinations will later be used in Section 3.2.3.3. The colored column is white when the number of rules for the corresponding combination of diagrams is zero. The 36 UML diagram

combinations refer to: 1) rules involving only one diagram such as AD, SD; 2) rules involving pairs of diagrams such as SD and UCD; 3) rules involving 3-tuples of diagrams such as CD, SMD and AD. However, not all the 14 diagrams appear in Table 10, since a diagram (or a pair or a 3-tuples of diagrams) is not shown in Table 10 if we were unable to find a rule involving that (those) diagram(s) in primary studies.

3.2.3.1 What are the existing UML consistency rules? (RQ1)

The principal observation we can make about RQ1 is that the researchers of UML consistency rules have typically defined a number of similar rules over and over again. Specifically, we collected a list of 621 rules from the 95 primary studies. After removing rules that were inaccurate, redundant, and already presented in UML standards, we obtained a final set of 116 rules that are presented in Table 18 in Appendix A. In other words, only 18.68% (116 out of 621) of the rules initially collected are included in our final set of rules because of our include/exclusion criteria (Section 3.2.1.2). The remaining rules were eliminated because they were inaccurate: 31.72%, 197 out of 621; resulting in 424 accurate rules. Specifically, 87 (14.00%) were not consistency rules (e.g., rules describing good modeling practices); 28 (4.50%) were explained in a too ambiguous language; 14 (2.25%) were obsolete because they referred to UML elements that are no longer used in the latest UML standard; and 68 (10.95%) were simply wrong (i.e., contradicting the UML specification). Other rules were mostly eliminated because of redundancies (57.08%, 242 out of 424). Finally, 66 out of 182 (36.26%) non-redundant rules were excluded because they are already part of the UML standard.

The rules with the largest number of references in primary studies are rule 112 (with 48 references), and rules 110, 48, and 115 (respectively with 33, 25, and 15 reference each). In a nutshell, rule 112 focusses on the required visibility in order to exchange messages in a sequence diagram; rule 110 concerns the consistency of messages in a sequence diagram and the class diagram; rule 48 specifies the consistency of behavior specification in lifelines of a sequence diagram and a state machine of the corresponding classifiers; and rule 115 describe the consistency of class names in class diagrams and sequence diagrams. The complete list of references for the 116 rules is presented in Table 18 of Appendix A, and elsewhere [17].

3.2.3.2 Which types of consistency problems are tackled in the existing rules? (RQ2)

The results obtained for RQ2 show (see Table 10) that the vast majority of rules are Horizontal (83.62%, 97 out of 116 rules) and Syntactic (81.90%, 95 out of 116 rules). Moreover, we found 21 (18.10%) Semantic rules. Researchers strikingly described many more syntactic than semantic consistency rules. We conjecture that the main reason for this is that syntactic rules are easier to specify than semantic rules and that the UML standard more formally specifies syntax than semantics, which hinders the specification of semantic rules. It may also be the case that semantic rules are specific to the way in which the UML notation is actually used, which is organization, project, or team specific, and are therefore seldom described in published manuscripts.

Researchers have also paid much less attention to: 1) Invocation rules (8 rules, 6.90%); 2) Vertical rules (7 rules, 6.03%); Evolution rules (3 rules, 2.59%); and Observation rules (only one rule, 0.86%). We believe that the mappings between vertical levels, the evolution of a UML model, the invocation and observation consistency, are concepts much less easy to understand than the mere specification of a UML model or the horizontal consistency levels, and this explains why, even though we found papers discussing these consistency dimensions, these papers did not present many consistency rules.

3.2.3.3 What types of UML diagrams are involved in UML consistency rules? (RQ3)

In this Section, we use a bubble plot to represent multiple dimensions of the results in one Figure. Combined with the bubble plot are some bubbles (left-hand side) which are pie charts to help us describe the many dimensions involved in this research.

Figure 10 shows the mapping of the selection process onto the 36 combinations of UML diagrams presented in Table 10: the combinations are numbered vertically in Figure 10. On the left-hand side of Figure 10, each bubble pie chart describes the number of rules included in the final set (in red) and the number of rules deleted (in yellow) for each of the 36 UML diagram combinations. For instance, for combination 1 (i.e., SMD, as per Table 10, i.e., State Machine Diagram as per Section 3.2.1.3) the initial set of rules involving only the State Machine Diagram contained 66 rules (14+52), of which we removed 52 and kept only 14. On the right-hand side, we present our reasons for deleting rules and the magnitude of each reason for each combination of diagrams. For

instance, of the 52 deleted rules involving only the State Machine Diagram, 10 were deleted because they were ambiguous (beyond repair), 24 were already in the standard, six were redundant (with rules we kept), eight were not consistency rules, and four were wrong.

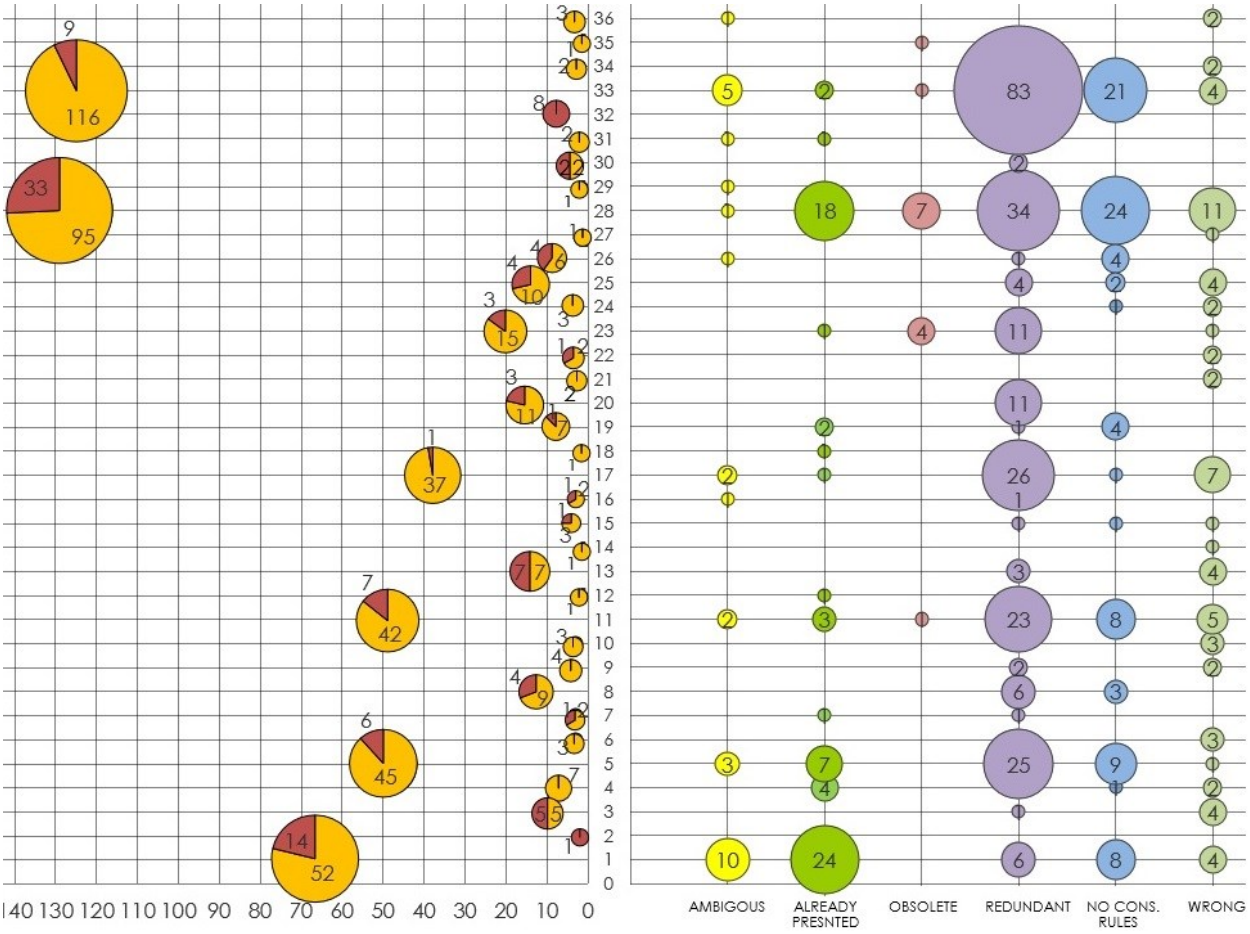

Figure 10. Summary of UML Consistency rules selection (see accompanying text for a detailed description of the semantics of the figure)

The results show that the diagram combination that involves the largest number of rules in the initial set, specifically 128 rules (33+95), is combination 28 (y-axis), i.e., the Class Diagram: the majority of the rules involve only the Class Diagram. This is followed by combination 33, i.e., rules involving the Class Diagram and the Sequence Diagram, with 125 rules (9+116). These two sets of rules make up 40.74% (253 out of 621 rules) of the total number of initial rules, and 36.21% of the final set of rules (42 out of 116). It is also important to mention that there is a huge gap between these two most involved diagram combinations and the next most involved diagram combinations, which are the State Machine Diagram (66 rules, y-axis value 1), Sequence Diagram (51 rules, y-axis value 5), Class Diagram with State Machine Diagram (49 rules, y-axis value 11),

and State Machine Diagram with Sequence Diagram (38 rules, y-axis value 17). The absence of rules for some diagrams or some diagram combinations can be explained by the fact that the semantics and syntax of some diagrams overlap (e.g., package, object and class diagrams). However, it is possible to argue that the fact that the semantics and syntax of those diagrams overlap, and that there is a need for these different diagrams, should justify the definition of rules to ensure that the diagrams are consistent, unless all the rules required are already in the UML standard (which is unlikely). Another possible reason for a lack of rules for some diagrams (or diagram combinations) is that users of the UML notation follow a specific UML (behavior) modeling practice that is assumed to be standard and does not therefore require the definition of consistency rules. Nevertheless, since the UML allows different behavior modeling practices, rules underpinning such practices should be specified so as to be clear for all stakeholders.

Another interesting finding is the high percentage (88.33%) of rules eliminated for the Sequence Diagram. If we consider all the UML diagram combinations in which the Sequence Diagram is involved (rows 3, 5, 13, 17, 33 and 34 in Figure 10), it is actually possible to see that, when starting with the total of 240 rules collected, only 28 (11.67%) made it to the final set of 116 rules. The main reason why rules were discarded is redundancy, i.e., rules presented several times by authors: for instance, 25 of the 45 discarded rules involving only the Sequence Diagram (row 5 in Figure 10) were redundant. One possible explanation for this is that the sequence diagram is one of the most ill-specified diagrams in the UML specification and researchers are attempting to make sense of it by specifying rules which, when coalesced, happen to be redundant with one another (138 of the 212 eliminated rules, 65.09%, were redundant with other rules). We discussed the 36 different combinations of UML diagrams (detailed in Table 10) covered by the final set of 116 rules. Our results show that rules collected describe consistency in only 10 of the 14 UML diagrams.

Figure 11 shows only seven diagrams because we grouped rules for the Sequence, Communication, Interaction Overview and Timing diagrams together under the Interaction Diagram (ID) category since these diagrams are different variants of the ID [9]. We did not find any rule involving the Package, Component, Profile, Timing, Interaction Overview and Deployment diagrams. We conjecture that there are several reasons for the lack of rule involving these diagrams. First, as already mentioned, the fact that the Package Diagram and the Deployment Diagrams overlap with the Class Diagram might be a reason for the lack of rule involving these diagrams; However, as discussed previously, this similarity between the two diagrams could be the very reason needed to

justify more rules involving these two diagrams or the Package Diagram with other diagrams. The lack of widespread tool support and underspecification/ambiguity in the specification for the Timing and Interaction Overview diagrams is a possible reason for the lack of rules involving these diagrams.

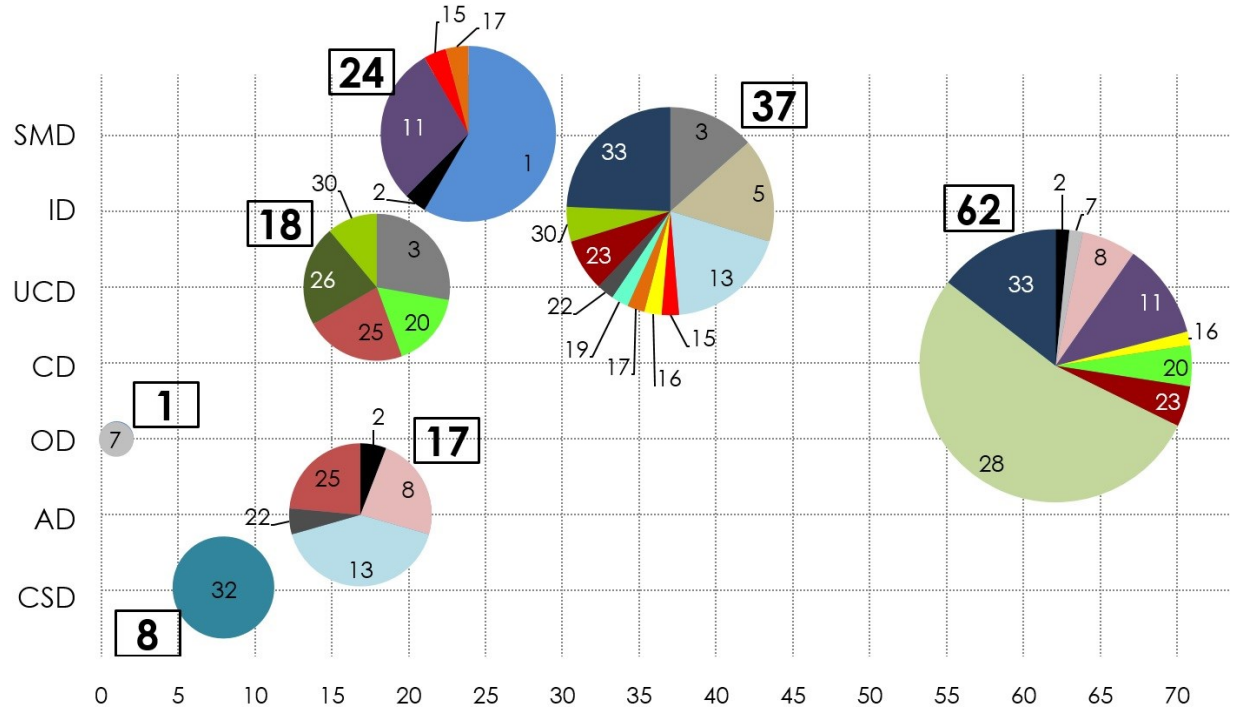

Figure 11. UML Consistency rules grouped into UML diagrams (see accompanying text for a detailed description of the semantics of the figure).

Figure 11 uses the same color code and ID# numbers for diagram combinations as Table 10. The Figure shows, for each of the seven diagrams involved in at least one rule (labels on the y-axis), the total number of rules found that involve that diagram (large font size, bold faced, square-bordered number) and the proportions of those rules that are shared with other diagrams as a pie chart. These data were taken directly from Table 10, although they are presented in a different form. For instance, the Activity Diagram (AD) was found to be involved in 17 rules. Table 10 indicates that one rule involves AD with CD and SMD (combination number 2), four rules involve AD and CD (combination 8), seven rules involve AD and SD (combination 13), one rule involves AD and COD (combination 22) and four rules involve AD and UCD (combination 25); These appear clockwise from the top of the pie-chart for row AD in Figure 11. Note that upon summing up the rule count of each pie chart in Figure 11 we obtain 167 rather than 116 because when a rule involves more than one diagram it contributes to more than one pie chart.

Not surprisingly, the UML diagrams that are most involved in the final set of 116 rules are the Class Diagram (62 rules, 53.45%), the Interaction Diagram (37 rules, 31.90%), and the State Machine Diagram (24 rules, 20.69%). Research on UML consistency rules has paid much less attention to the Use Case Diagram (22 rules, 15.52%) and the Activity Diagram (17 rules, 14.66%).

The diagrams that are least covered are the Composite Structure Diagram and the Object Diagram, which only had 6.90% (eight rules) and 0.86% (one rule) of the total of 116 rules, respectively.

3.2.3.4 For what software engineering activities are UML consistency rules used? (RQ4)

In this Section, we discuss the results of the different uses of rules. As we already explained in Section 3.2.3, we answered this RQ by considering the set of 424 rules rather than only the final set of 116 rules. We chose to do this because, since redundant rules may have been defined in the context of different Software Engineering activities, we wished to show what researchers have focused on.

Results (pie chart on the left-hand side of Figure 12) show that 51.89% (220 out of 424) of the rules are proposed for model consistency verification (Verification in the Figure), 13.44% (57 out of 424) for impact analysis (Impact Analysis in the Figure), 12.74% (54 out of 424) for model refinement and transformation (Ref. & Tra.), 12.03% (51 out of 424) for consistency management (Management), 4.95% (21 out of 424) for model understanding (Understanding), 3.30% (14 out of 424) for model formalization (Formalization), and 1.65% (7 out of 424) for safety and security consistency (Saf. & Sec.). All the percentages shown in this Section are rounded off in the pie chart (left-hand side) of Figure 12.

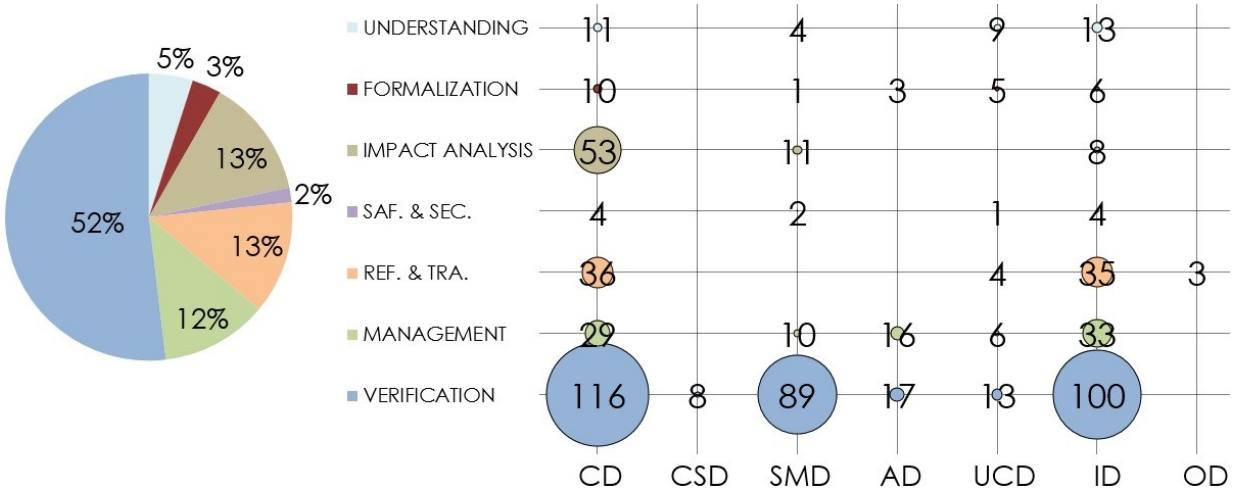

Figure 12. Rules between UML diagrams and uses in Software Engineering activities (see accompanying text for a detailed description of the semantics of the figure).

In the light of these Figures, the reader may be surprised to discover that half of the consistency rules have been defined without any additional context of use such as specific Model-Driven activities. This may suggest that not many UML-based software engineering activities require UML diagrams to be consistent, or at least that papers describing UML-based software engineering activities do not discuss (or need to rely on?) consistency rules. On the other hand, this may rather mean that typical UML-based software engineering activities require the same set of consistency rules, which can therefore be described independently of their context of use.

3.2.3.5 Combining RQ4 and RQ3

Upon combining the data presented for RQ4 in the previous Section3.2.3.4 (For what software engineering activities are UML consistency rules used?) with the data presented in the analysis of RQ3 in Section 3.2.3.3 (What types of UML diagrams are involved in UML consistency rules?), we show in Figure 12 (right-hand side) which UML diagram was covered by rules in which Software Engineering activity.

The bubble plot in Figure 12 (right-hand side) shows that the Class Diagram (CD) is mostly involved in rules used for verification (116 rules) and impact analysis (53 rules) activities. Furthermore, the Interaction Diagram (ID) and State Machine Diagram (SMD) have 100 and 89 rules, respectively, which are used for verification. This is not entirely surprising since these three

UML diagrams are likely the behavioral UML diagrams that are most frequently used in the activities cited above.

Figure 12 (right-hand side) shows a total of 660 rules rather than 424 accurate rules because we grouped all the rules related to each of the seven UML diagram separately, resulting in some rules being repeated.

In Figure 13 we present a bubble plot distribution that was obtained by combining the results of RQ3 and RQ4 with the years of the rules presented in Section 3.1.3.8.

Figure 13. Summary of UML Consistency rules between UML diagrams (see accompanying text for a detailed description of the semantics of the figure).

In Figure 13, we coupled the years of rule's publication, on the right-hand with the UML diagram covered, and on the left-hand with the Software Engineering activity that used the rule. The right-hand side of Figure 13 shows the number of rules presented in the period between 2000 and 2012 and classified by the seven UML diagrams covered by the rules. As in the previous two Sections, we grouped all the rules related to each of the seven UML diagrams, resulting in 660 rules instead of 424 rules, since some rules were repeated. For instance, the 17 rules presented for CD and SMD (combination 11 in Table 10) were included in both the CD count and in the SMD.

However, the left side of Figure 13 shows only 424 rules in the period between 2000 and 2012 (without overlapping) classified in the eight Software Engineering activities presented in Section 3.2.1.3. On the left-hand side of Figure 13, no rules overlap because each of the 95 primary study papers (Section 3.1) was assigned only one of the eight Software Engineering activities presented in Section 3.2.1.3. The rules presented in each paper were consequently classified according to only those seven activities.

The results show (left-hand side of Figure 13) that a great number of rules were proposed in four of the 13 years considered in this study: 49.76% (211 out of 424 rules) of the rules were published between 2002 and 2005. In these four years, 152 out of 259 (58.69%) of the total number of rules related to the Class Diagram and 103 out of 199 (51.76%) of the total number of rules related to the Interaction Diagram were published (right-hand side of Figure 13). Moreover, in these four years we can see that the three peaks of Software Engineering activities use (left-hand side of Figure 13): 1) 57 rules presented in 2003 for impact analysis, 32 rules in 2005 for verification, and finally 25 rules in 2004 for refinement and transformation activities. We can conclude that these four years represent the most prolific period in literature in terms of publishing UML consistency rules to date.

Another important aspect that should be highlighted is that the number of UML consistency rules covering the Verification activity (left side of Figure 13) remained relatively stable in the entire period considered, with a peak of 32 new rules in 2005 and with only seven rules being presented in 2006 and 2012. Furthermore, the right-hand side of Figure 13 shows that class and interaction diagrams remained stable in the period between 2002 and 2009 with a peak of 72 rules in 2003 and 35 rules in 2004, respectively. The fact that researchers consistently continued to focus on class and interaction diagrams during these years, shows the vital importance of these diagrams.

The number of rules decreased in 2012, but this is likely because many papers published in that year were not yet available online when we performed searches [11].

3.2.3.6 Which UML diagram elements are involved in UML consistency rules? (RQ5)

Figure 14 shows the 200 most frequently used words in the plain English specification of the 116 rules found (4761 words in total), in which we can recognize the 60 UML element names already presented in Table 10. This is a weighted word cloud of the 116 rules found in the 95 papers included as primary studies presented in Section 3.1. Figure 14 was created using Tagxedo[6], which uses a streaming algorithm to filter the text input. The aim of Figure 14 is to provide a first impression of the main UML diagram elements referenced in the final set of 116 UML consistency rules. Not surprisingly, the UML elements most involved in the rules are related to the class, sequence, state machine, use case and finally activity diagrams.

Figure 14. Summary of the UML diagram elements involved in UML Consistency rules

These results concerning the model elements most frequently involved in UML consistency issues may be useful as regards helping focus the teaching of UML by concentrating on specific elements.

[6] http://www.tagxedo.com/

Indeed, if an element is not rigorously specified, then there is a risk of many semantics' issues appearing in the model.

3.2.3.7 Additional results

In this Section, we shortly discuss the 66 rules that were excluded from the final set of rules because they had already been presented in previous UML standards. It is important to note that 66 of the 182 (36.26%) non-redundant and accurate UML consistency rules presented by different authors had already been proposed in previous UML standards. We compared the rules with the applicable UML specification version at the time of publication (in which the rule was found): we either found a reference to the applicable version in the publication or inferred it from the time of publication; finally, we also checked whether the rule was contained in the most recent version of the specification (UML 2.4.1 and 2.5). The 66 rules were excluded because they had already been presented in one of the previous UML standards (UML 1.3, 1.5, 2.0, 2.4.1, and 2.5).

3.2.4 Conclusions of Section 3.2

In recent years, a great number of UML consistency rules have been proposed by researchers in order to detect inconsistencies between UML diagrams. However, no previous study has, to the best of our knowledge, summarized and analyzed these UML consistency rules by following a systematic process. Section 3.2 presented the results obtained after following a systematic protocol, whose aim was to identify, present and analyze a consolidated set of 116 UML Consistency rules from literature. No such mapping study or review existed prior to our work. Further conclusions and observations regarding Section 3.2 will be discussed in Chapter 6.

4. IS THIS CONSOLIDATED LIST OF UML DIAGRAM CONSISTENCY RULES RELEVANT? (G2)

The next step from Chapter 3 is to validate the set of rules (Research Goal 2 (G2)). Indeed, even though we very carefully conducted the Systematic Mapping Study (SMS) and collecting of UML consistency rules, there is no complete guarantee that we have not made a mistake of missed rules. In the field of empirical software engineering, this is considered a threat to the validity of our results. We therefore need to corroborate the results, i.e., we need to validate the set of UML consistency rules that we have obtained. For the purpose of this research, we have the following main validation question: how to check whether the UML 116 consistency rules are relevant? To achieve this goal, two types of validation will be presented in the next Sections where we intend to:

1. Rely on the expertise of a number of individuals, experts from academia and from the industry (Sections 4.1 and 4.2);
2. Rely on existing UML models composed of varying numbers of diagrams (Section 4.3).

In the following Sections, we describe the specific objectives for each validation process and how they were finally executed.

4.1 1st International Workshop on UML Consistency Rules (WUCOR)

Despite the fact that there is a need for UML diagram consistency, and even though different ways in which to reason about consistency rules exist, our results in Sections 3.1 and 3.2 [11] show that: 1) there is no well-accepted set (that is as complete as possible) of consistency specification rules, or simply rules, for UML diagrams (beyond the small set of well-formedness rules in the standard specification); 2) many researchers have explicitly or implicitly proposed rules with which to detect inconsistencies, without any effort being made to validate those rules; 3) the majority of the

consistency rules target a small subset of the UML diagrams (mostly, class, sequence, and state machine diagrams); 4) researchers have repeatedly presented non-negligible sets of consistency rules (rather than, for example, referring to an accepted list of such rules); 5) non-negligible sets of consistency rules presented by researchers are actually included in the UML standard itself; 6) the UML standard is a long way from providing a comprehensive set of consistency rules; 7) the vast majority of consistency rules are horizontal and syntactic (the remaining dimensions have barely been considered in those rules).

These observations motivated a proposal for a workshop called "1st International Workshop on UML Consistency Rules" (WUCOR)[7], during which we sought experts' opinions about the set of 116 UML consistency rules we coalesced in Section 3.2 from the systematic mapping study described in Section 3.1. This workshop provided an opportunity for researchers who have been working on UML consistency, or whose (research) activities require consistent diagrams, to work together at a highly interactive venue with the objective of validating previously collected rules [17], and discuss ideas for further research in this area.

The next Sections are structured as follows. In Section 4.1.1, we provide a brief discussion on previous events on UML consistency. This is followed by the specification of the workshop goals and proceedings (Section 4.1.2), and a summary of the papers presented (Section 4.1.3). A detailed description of the activities is presented in Section 4.1.4. A preliminary discussion, along with the main findings and the limitations of the research, are provided in Section 4.1.5. Finally, Section 4.1.6 draws the conclusions.

4.1.1 Previous events on UML consistency

We are not aware of any conference that is specifically dedicated to the issue of consistency among UML diagrams. Our systematic mapping study [11], during which we carried out a rigorous and systematic search, did not find any such event. We are only aware of a workshop entitled "Workshop on Consistency Problems in UML-based Software Development" which took place in conjunction with the UML conference in 2002, 2003, and 2004. These workshops differ from our workshop WUCOR, celebrated within MODELS 2015, in two main ways. First, they were seeking

[7] https://wucor.wordpress.com

contributions from authors on any kind of issue regarding the consistency of UML diagrams (e.g., consistency rule specifications, tooling support with which to check rules, inconsistency repair strategies), whereas we wished to specifically focus on the consistency rules/specifications that seem to be needed by the research community in order for one set of consolidated rules to be defined. Secondly, as will become clearer later in this document, WUCOR was not only seeking paper contributions that would be presented in a way similar to the majority of the workshops that take place concurrently to conferences, in a format akin to a small conference. On the contrary, we primarily encouraged working groups to debate specific issues during the workshop.

4.1.2 Workshop goals

The objective of the workshop was to bring together anyone, from either industry or academia who was interested in consistency rules among the UML diagrams of a given model, and to provide a platform for discussions, interactions and collaborations regarding this topic. The goal was to gather input and feedback on the topic of UML consistency rules from the community. One of the starting points for the discussion groups was our set of 116 unique UML consistency rules that we had coalesced in Section 3.2. We also asked for expert opinion on the subset of those rules that should be deemed paramount, and should therefore always be enforced, and the other rules that can be considered optional.

The papers presented at the workshop were collected in the WUCOR proceedings [19] and were peer-reviewed by three independent reviewers.

4.1.3 Summary of the presented papers

In this Section, we summarize the main results of the two papers that were accepted at WUCOR 2015.

Hoisl and Sobernig [54] analyzed consistency aspects extracted from 84 UML-based domain-specific modeling language (DSML) designs collected via a systematic literature review [55]. They focused exclusively on consistency rules defined at the level of a DSML. For the evaluation of UML consistency aspects, they adopted criteria from closely related work [11, 17]. They then interpreted the consistency-related data extracted in order to discuss frequently identified defects

in UML-based DSML language models. Their study showed that a UML-based DSML is predominantly formalized via the definitions of profiles, which mostly specializes the class, activity, component, and package diagrams. They also noted that constraints, which are specified in natural language or OCL, are most frequently used in combination with these profiles to define consistency rules in a single model for verification purposes. In the majority of cases, they found that the DSML papers do not describe any tool support with which to enforce these rules.

D. Chiorean, Petrascu, and I. Chiorean [56] proposed a change of attitude regarding the definition of the UML's abstract syntax that would improve the quality of the standard specification. They described this improvement as a condition for attaining the value of Model-Driven technologies and paradigms. Their proposal is supported by examples taken from the UML specification [57]. They argue that the first requirement is for the well-formedness constraints in the specifications to have complete, accurate and clear natural language descriptions. Once this requirement is met then the constraints should have associated formal specifications in OCL. Finally, the OCL specifications for those constraints have to be compared to those of the natural language and any synchronization between the two must, if necessary, be carried out.

4.1.4 Description of the activities

In this Section, we describe the three activities that were conducted during WUCOR (see the details of the WUCOR program [19]). These activities consisted of:

- Activity 1 (A1) focused on the definitions of the types and dimensions of UML consistency rules. Its main goal was to ensure that the attendees would work with common definitions during the remaining activities. This activity was divided into two parts, as explained in Section 4.1.4.1, and was allotted 40 minutes.
- Activity 2 (A2) provided an overview of the state of the art in terms of the involvement of UML diagrams in UML consistency rules. This activity was divided into two parts, as explained in Section 4.1.4.2, and was allotted one hour.
- Activity 3 (A3) concerned the validation of some of the 116 UML consistency rules that we had collected in [17]. This activity was allotted one hour and 50 minutes.

The results of these three activities were discussed in the last session of WUCOR in one hour and 25 minutes.

A detailed description of these three activities is provided in the following three subsections.

4.1.4.1 UML Consistency rules and dimensions (A1)

In the first part of activity A1, we provided the attendees with the most frequently used definitions of the three UML consistency dimensions (Horizontal, Vertical, and Evolution consistency) and of the two UML consistency types (Semantic and Syntactic consistency) as described in Section 3.1.1.4. We then asked them the following questions:

1. Is there something that you would like to modify/improve in the wording?
2. Are there any aspects of a dimension or type that are not covered by a definition?
3. Would you please leave your comments in the boxes provided after the definitions?

In the second part of activity A1, we explained that during our systematic mapping study (Section 3.1) we observed that the vast majority of UML consistency rules focus on the Horizontal dimension and Syntactic type, and very few rules are related to Vertical or Semantic, Evolution, Observation, Invocation consistency dimensions. With this introduction, we asked:

1. Does this suggest that dimensions (other than Horizontal) and types (other than Syntactic) of consistency are not relevant to UML or does it suggest that rules are just missing?
2. Would you please leave a comment? Feel free to present examples of UML consistency rules that cover dimensions and types of UML consistency that are not very popular (such as Evolution, Vertical, Invocation, Observation, and Semantic consistency).

4.1.4.2 UML diagrams involved in UML consistency rules (A2)

In the first part of activity A2, we explained that the class diagram is the UML diagram most involved in UML consistency rules presented by researchers to date, followed in importance by the interaction diagram and the state machine diagram as described in Section 3.2. According to another study [58], the activity diagram is the second most frequently used UML diagram after the

class diagram. We, however, found very few rules involving the activity diagram (we presented a list of 28 rules we found that involved the activity diagram [17]). We then asked:

1. Should research into UML consistency focus more on the activity diagram? What would additional consistency rules involving the activity diagram be?
2. Would you please leave your comment on this page? Feel free to present examples of new UML consistency rules that involve UML activity diagrams.

In the second part of activity A2, we showed that in our previous research [11] we did not find a single rule involving the package, profile, component, timing, interaction overview and deployment diagrams. We therefore asked:

1. According to your expertise, why did we obtain those results?
2. Should research into UML consistency focus more on these diagrams?
3. Would you please leave your comment below? Feel free to present examples of new UML consistency rules that involve the previously cited UML diagrams.

4.1.4.3 UML Consistency rules in MDD (A3)

In activity A3, we presented a questionnaire on each of the 116 rules coalesced in Section 3.2. Each rule was presented on a single piece of paper on which we described the rule with an example (unless we considered the rule trivial or easy to understand without an example). For each rule we asked the attendees the following questions:

1. Do you understand the rule?
    a. Yes;
    b. No.

    Would you like to re-phrase it? How?
2. Do you think this is a valid rule? Please check one of the following options and explain your decision:
    a. It is a valid rule and should be enforced in all UML models;
    b. It is a valid rule in some situations but not always;
    c. This rule should not be enforced;
    d. I am not sure.

The expression "valid rule" refers to the fact that a specific rule could be relevant: always (a), in some situations (b), or never (c).

3. UML consistency rule complexity: the complexity of a UML consistency rule is directly related to the complexity of the consistency it attempts to specify. Complex UML consistency rules are generally difficult to understand, making it harder to detect such problems in a diagram or among diagrams.

   According to your expertise, how would you consider this rule? Please choose one of the following options and explain your decision:

   a. Very Complex;
   b. Complex;
   c. Normal Complexity;
   d. Simple;
   e. Very Simple;
   f. Other.

4.1.5 Results

In this Section, we describe only the analysis of the results obtained from the questions in activity A3 and the discussion generated on this activity. A limited number of people attended WUCOR (ten attendees including two of the organizers), and we are therefore of the opinion that the results of this activity (A3), considering the high number of rules to check, are the only ones that may be considered a valid initial point for future improvements and replications for this research. Without a larger number of respondents, the results of activities A1 and A2 would only provide some personal opinions (of the ten participants) instead of general trends.

During activity A3, we were able to implement the questionnaire (presented in Section 4.1.4.3) for 81 out of the total of 116 (69.83%) UML consistency rules [17]. 73 of the 81 (90.12%) UML consistency rules were understood by the attendees. These 73 rules were then used as a basis to define a bubble plot in which to report the frequencies of combining the results from questions 2 and 3 of activity A3 (Section 4.1.4.3). After combining the results for questions 2 and 3 of activity A3, we obtained (see Figure 15) the mapping of the complexity of the rules depending on whether or not the rules were considered valid.

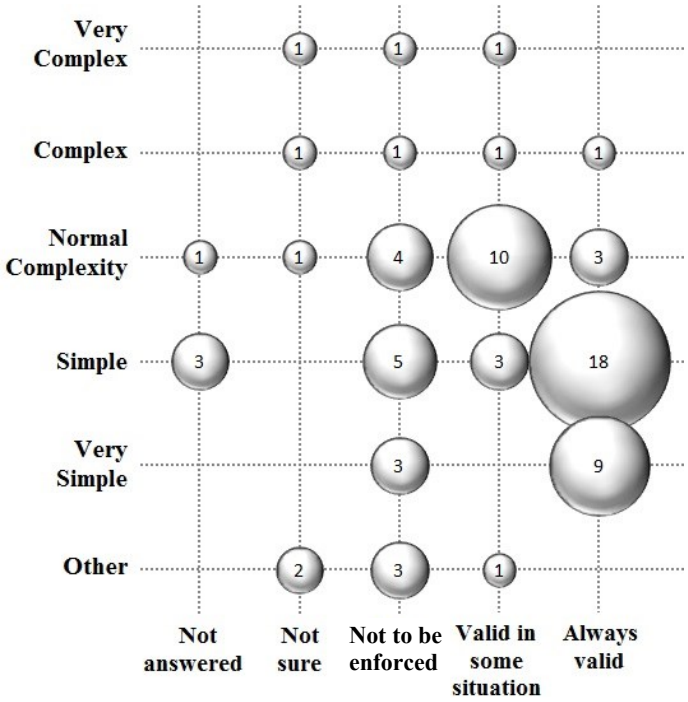

Figure 15. Combining question 1 and 2 of A3 (see accompanying text for a detailed description of the semantics of the figure).

Figure 15 shows that the attendees identified 18 (24.65%) simple rules as always valid. These rules were followed by another ten (13.69%) with a normal complexity that are valid in only some situations. 56.16% (41 out of 73) of the rules checked were considered to be simple or very simple. Another important aspect that should be noted is that 47 of the 73 (64.38%) understood rules were positively considered as valid rules to be enforced (31 always valid, and 16 valid in some situations). The attendees, however, stated that only 23.28% (17 of 73) of the rules checked should not be enforced.

The reasons why eight of the 81 (9.87%) rules were not understood (question 1 of A3) by some of the attendees were:

- one attendee was not familiar with a particular concept presented in one rule;
- one attendee did not understand the specification of one rule and asked for an example scenario involving it;
- two attendees considered two rules to depend on a specific development methodology, which made them difficult to understand;
- one attendee considered the description of one rule to be incomplete;

- three attendees did not provide any justification for three rules, although they were explicitly asked to justify/explain their answers.

4.1.6 Conclusions of Section 4.1

In recent years, researchers have put a great deal of effort into identifying UML consistency rules so as to detect inconsistencies between UML diagrams [11]. However, no previous workshop has been organized to discuss the state of the art in terms of UML consistency rules. WUCOR 2015 has been the first workshop entirely focused on UML consistency rules to gather feedback from the community, either industry or academia, on this topic. The format of WUCOR [19] was a mix of two presentations, three activities developed with working groups, and a final activity dedicated to discussions. We had two presentations from authors who reported their studies on distinct software engineering subjects using different research methods (see Section 4.1.3 for more details). The two presentations were followed by three activities (see Section 4.1.4 for activities and Section 4.1.5 for results), and finally the last Section of WUCOR consisted of a discussion on the three activities during which the participants interacted with each other. We did not change the definition of any of the 81 rules checked at the WUCOR, i.e., how we expressed them, in the hope to be able to combine the feedback of WUCOR with the one obtained by the survey presented in the next section. In the end, we only used the results of the survey and did not use the evaluations collected in WUCOR.

4.2 Online survey: how consistency is handled in MDSE and UML

Due to the limited attendance of the workshop described in Section 4.1 and to complement its results we decided to develop a survey with Model Driven Software Engineering (MDSE) experts from academia and industry with the following main objectives (MO):

MO1) Surveying the diffusion and importance of model consistency issues in order to understand model consistency in the general MDSE context as well as in the specific UML context;

MO2) Validating the 116 UML consistency rules, i.e. identifying the rules that should be always enforced.

The results of this survey will finally contribute to evaluate the potential practical impact of our long-term investigation. By the statement "validate the rules", we mean the process of asking MDSE experts a number of questions regarding the 116 UML consistency rules, i.e. we want to collect their opinion regarding how easy it is to understand the rules, and how relevant it is to enforce those rules in UML models. In order to carry out our two main objectives, the following four research questions (RQs) were formulated:

> ***RQ1*:** What is the demographic information of the survey participants? (MO1)
>
> ***RQ2*:** How do the participants handle the consistency problems in the context of Model Driven Software Engineering? (MO1)
>
> ***RQ3*:** How are UML inconsistencies handled by the participants? (MO1)
>
> ***RQ4*:** Are the 116 UML consistency rules we coalesced from the literature [17] relevant and easy to understand by the participants? (MO2)

We decided to carry out a survey because it is a well-established social science technique that can be used to gather information and opinions from a large population that is known to be representative of a target population [59]. Even though surveys have been used frequently in MDSE, for example to study MDSE in the embedded system domain [60], to study MDSE practices in industry [61], and to study the use of UML documentation in software maintenance [62], MDSE requires more empirical evaluations to support its claimed benefits (e.g. gains in productivity, portability, maintainability and interoperability) [63, 64]. To the best of our knowledge, the survey we discuss and report on in this Section is the first that intends to find empirical evidence about consistency issues in MDSE/UML. For this reason, there will not be any related work for the work presented in Section 4.2.

The next Sections are organized as follows: in Section 4.2.1, we describe the process we followed to carry out the survey. The results of the survey and its implications are presented in Section 4.2.2. Finally, the main findings with some concluding remarks are summarized in Section 4.2.3.

4.2.1 Research Methodology

The survey method is a well-established technique for collecting data about features, behavior or opinions of a specific group of people that is representative of a target population [65]. Thus, a survey aims to produce quantitative data on some aspects of the population under study. There are different surveying methods, each with different advantages and disadvantages [66]. In this study, we chose to use the on-line survey method since we wanted to obtain information from a relatively large number of experts in a quick manner so that we can easily categorize and analyze these data and answer the four research questions mentioned in Section 4.2. We used software engineering survey guidelines [67-69] and other surveys focusing on MDSE [60-62] as guides. Hereafter, we detail the design, execution, and pre-analysis data validation of the survey.

4.2.1.1 Survey Design

The survey was designed to capture the most relevant information to answer our four research questions. For this purpose, the main issues related to the survey design are: the identification of the target audience and respondents, and the design of the questionnaire, which will be explained below.

4.2.1.1.1 Identifying the Target Audience and the Sample of Respondents

The first step to conduct a survey is defining a target population. In our study, the target population is experts in MDSE and UML contexts. Considering that the target population is very specific and of limited availability [67], we need to identify a representative sample. Since we need to make sure that respondents have sufficient knowledge to answer the questions, our sample of respondents is composed of experts from academia and the industry who have been involved and/or have published in conferences and workshops about MDSE/UML topics. To create this sample of the target population we looked at the authors of the primary studies selected in our SMS on UML Consistency presented in Section 3.1 (see references in Appendix D), and from the proceedings of the editions of the following events: The 1st International Workshop on UML Consistency Rules (WUCOR) [18]; The International Conference on Model Driven Engineering Languages and Systems (MoDELS) [70]; The Workshop on Modelling in Software Engineering

(MiSE) [71]. We selected those events, and primarily MoDELS because it is the premier conference series for model-based software and systems engineering with strong, sustained industry participation. We also used other personal networks of collaborators (in industry or academia), colleagues and other contacts. We believe that through this procedure we created a representative sample of the target population. Since we recruited subjects from a sample of the target population, our sampling method was non-probabilistic [67].

We stored the data (first name, last name, email, and affiliation) regarding the target population into an Excel spreadsheet. Our sample consisted of 455 experts: 315 from academia (218 identified from our Systematic Mapping Study [11], 113 from MoDELS [70], one from WUCOR [18], and 18 from MiSE [71]) and 140 from the industry (20 identified from our Systematic Mapping Study [11], and 120 from our networks).

4.2.1.1.2 Designing Survey Questions

Among the many possible instruments to conduct a survey, like paper questions, interviews, phone calls, we opted for an online questionnaire [68] since we estimated it would have more chances of leading to useful results, it would be more convenient to reach respondents around the world, and we would likely have a higher response rate (one can choose when to answer the questions at her/his leisure). To develop a survey that would adequately gather the information needed to answer the four research questions, we developed a questionnaire consisting of four Sections (one Section per research question) with a total of 21 questions (Q1 to Q21):

1) Demographic Information—the first Section of the questionnaire contained eight questions gathering general background data about respondents (answering RQ1).
2) Model Consistency: nine questions—the second Section focused on the importance, familiarity, and frequency of consistency problems' identification, and on the experience of the respondents in regard to consistency management in MDSE (answering RQ2).
3) UML Consistency—the third Section presented two questions specifically on UML consistency (answering RQ3).
4) UML Consistency Rules—the last Section included two questions about the relevance and understanding of the 116 consistency rules coalesced in Section 3.2 (answering RQ4).

A detailed description of the questionnaire is available in Table 19 of Appendix B.

We note that we had two categories of respondents in our sample: academics and practitioners (industry). As a result, we needed to tailor a couple of questions to those main categories. Tailoring however simply meant changing "institution" into "company" in a couple of questions. For instance, Q17 is "What modeling language(s) is/are used in your institution" when asked to an academic and "What modeling language(s) is/are used in your company" when asked to a practitioner. We also note that a respondent may have feedback to provide through the questionnaire as an academic and also as a practitioner. The questionnaire allowed such a respondent to disclose both kinds of experience; in this case the respondent then had to answer institution/company specific questions twice since an answer to a question (e.g., Q17) may be different when the respondent identifies herself/himself as an academic (i.e., it may be UML) and as a practitioner (i.e., it may be Simulink state flow). Last, Q17 was a pivotal question: the survey ended with this question for those respondents who answered to this question that they were not using UML. For the others, i.e., who said they were using the UML, the survey continued with Q18 to Q21 since these specifically focused on UML model consistency (see Figure 36 in Appendix B).

4.2.1.2 Execution

We decided to collect data through an on-line questionnaire created by means of a web-based questionnaire tool (Survey Monkey[8]), since web-based questionnaires have been observed to guarantee high response rates [72]. The survey was conducted from 15 May 2016 until 30 June 2016, and then closed to participants. The hyperlink of the survey was emailed directly to the selected target audience. Our online survey was structured in four Sections of questions as discussed earlier (see Figure 36 and Table 19 in Appendix B) that respondents completed by choosing between different options and filling forms.

To answer research question RQ4, i.e., the two questions in the fourth Section of the questionnaire which were about the UML consistency rules that we coalesced in Section 3.2, we wanted to (1) leverage the respondents' expertise to verify our work, (2) avoid taking too much of their time and

[8] http://www.surveymonkey.com/

facing the risk of respondents dropping out, while (3) covering as much of (ideally all) the list of UML consistency rules with at least two experts evaluating each rule. To achieve the first two objectives, the questionnaire asked questions about the rules (Q20 and Q21) in groups of five: a respondent was presented with five (randomly selected) rules, and, if the respondent was willing to look at more rules, five more (randomly selected) rules were presented, and so on, until the respondent decided s/he had enough or all 116 rules were evaluated (which never happened). To achieve the last objective, we monitored results of answered questionnaires on a daily basis to record progress and specifically record the number of times each rule was evaluated. We then carefully selected the rules that would be presented to the next respondents, making the questionnaire presenting rules with few evaluations thus far. Monitoring was necessary since a priori assigning rules to be evaluated by each respondent was not possible because we could not foresee how much each respondent would be willing to contribute.

4.2.1.2.1 Pre-considerations and data validation

106 respondents completed the questionnaire, resulting in a response rate of 23.3% (106 of 455 participants contacted); 142 respondents started answering the questionnaire but 36 of them did not complete it. Although achieving higher response rates do not necessarily mean more accurate results [73], we were very glad to see such a high response rate especially in light of the response rate of other studies in the field [74, 75], where response rates were low (respectively 12% and 9%). Furthermore, 68 of the 106 respondents identified themselves as academics, 20 identified themselves as practitioners (from industry) and 18 as both academics and practitioners, resulting in a total of 124 (68+20+18*2=124) questionnaires: 68+18=86 questionnaires from academics and 20+18=38 questionnaires from practitioners. (Recall from Section 4.2.1.1.2 that when a respondent identified herself/himself as both academic and practitioner, s/he answered a similar series of questions, resulting in two questionnaire responses.) This number of complete questionnaires is also larger than what was used in other surveys in the field of MDE: 22 questionnaires from industry participants [76], 50 questionnaires from professional software engineers [77], and 113 questionnaires from software practitioners [78].

4.2.2 Survey results

We now report on the results for each of the four-research question from Section 4.2.2.1 to 4.2.2.3.3, respectively. The survey questions presented from 4.2.2.1.1 to 4.2.2.1.3 consider 106 respondents. Starting with Section 4.2.2.1.4 and until Section 4.2.2.3.2, we consider a total of 124 respondents because, as explained below, we obtained answers from respondents working in academia (68), in industry (20), and in both academia and industry (18). The latter 18 respondents completed two different questionnaires: one for academia, and another for industry. For this reason, we consider 124 questionnaires/respondents (68+20+18+18) with only two types of roles: 1) respondents working in academia (86), and others working in industry (38).

4.2.2.1 Demographic Information

This Section answers RQ1 by means of eight questions.

4.2.2.1.1 Respondents' working countries (Q1)

The first question asked respondents about their geographic location. The 106 respondents reported working in 28 different countries: Figure 16. The color legend of Figure 16 shows with different intensity of green the frequency of the participants for each country. The countries without at least one respondent are depicted in grey. The countries, in which the participants work varied: Canada (13), France (10), United States (9), Spain, Italy and Germany (8), Austria and Norway (5), Finland, India, and United Kingdom (4), Brazil and Netherlands (3), Hungary, Australia, Luxembourg, Belgium, Colombia, Japan and Sweden (2), Czech Republic, Algeria, Chile, China, Ecuador, Israel, Romania and Turkey (1). Figure 16 shows that most of the respondents were located in Europe (61.3%), followed by North America (20.7%).

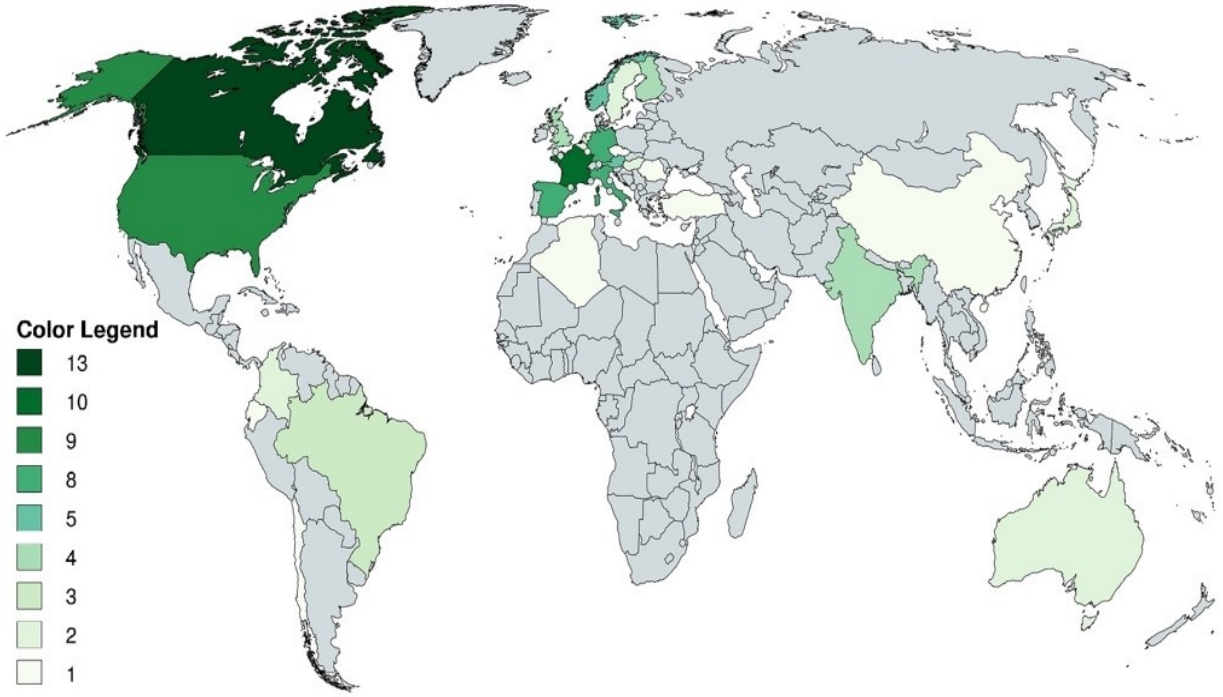

Figure 16. Geographical distribution of respondents

4.2.2.1.2 Respondents' level of education (Q2)

Participants were also asked to provide their highest academic degree. Results show that the majority of respondents (63, 59.4%) have a Ph.D.'s degree. This is followed by 18.8% (20) and 16% (17) of respondents who have a Master's and a Post-Doctoral degree, respectively. Finally, we have five respondents (4.7%) with a Bachelor's degree, and only one respondent reporting having a High school degree.

4.2.2.1.3 Respondents' current working place (Q3)

We learned that 68 respondents (64.1%) work in academia, 20 respondents (18.8%) work in industry, and 18 respondents (16.9%) work in both academia and industry. As mentioned earlier, the latter group, starting from the next Section, completed two different questionnaires: one as "academics" and one as "industry" professionals. In other words, we obtained 124 questionnaires and considered differently answers to other questions as coming from either a respondent from academia (68+18=86 questionnaires out of 124, i.e., 69.3%) or from industry (20+18=38 questionnaires out of 124, i.e., 30.6%).

4.2.2.1.4 Respondents' experience in the context of software engineering (Q4)

In this question, we asked about the working experience of the participants in the context of software engineering. Results are summarized in Figure 17. For each range of experience (e.g., more than 10 years at the bottom of the Figure), the Figure shows two bars: the "Academia" bar shows answers from academics (blue) and practitioners (orange) disclosing academic experience, while the "Industry" bar shows answers from academics (blue) and practitioners (orange) disclosing industry experience; both categories of respondents (academics and practitioners) were able to disclose their own experience in both academia and industry (the two bars for each range of experience level). As a result, the sum of data points for each category of respondents (e.g., blue) is twice the number of respondents in this category (e.g., 86*2=172 when summing up data points in blue bars). For instance, among the 86 academic respondents (blue), 50 disclosed more than 10 years university experience and 13 disclosed more than 10 years of industry experience (note that these two sets of data may overlap, i.e., one respondent may disclose more than 10 years of academic experience and more than 10 years of industry experience); among the 38 practitioner respondents (orange), 12 disclosed more than 10 years of university experience and 23 disclosed more than 10 years of industry experience.

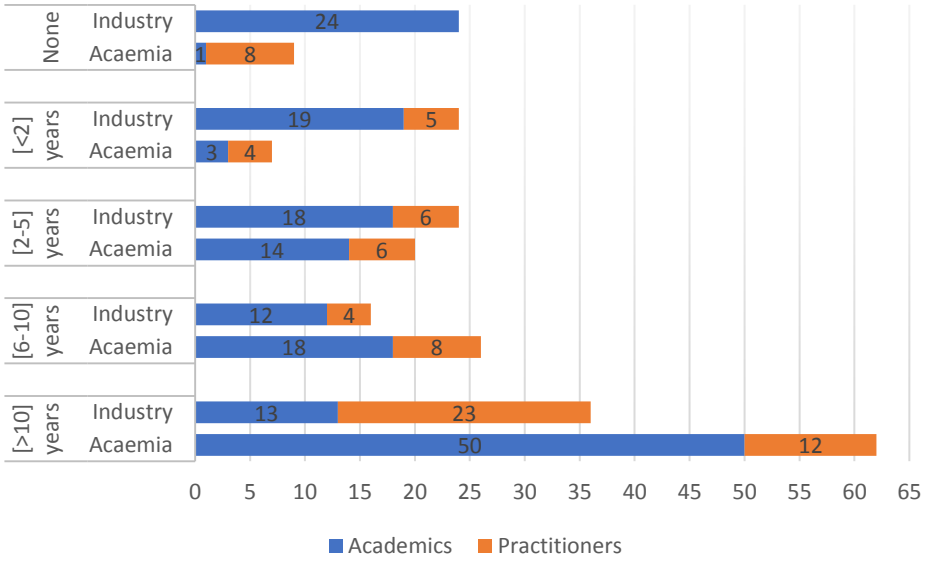

Figure 17. How long have respondents been working in the context of Software Engineering (see accompanying text for a detailed description of the semantics of the figure).

The results reflect that the majority of respondents have 10+ years of experience in their work. 58.1% (50 out of 86) of academics have 10+ years of academic experience and 60.5% (23 out of 38) of practitioners have 10+ years of industry experience. These percentages grow to 79.1% and 71% when looking at 6+ years of experience. Also, a large proportion of respondents, either academics or practitioners, disclose 10+ years of either academic experience (50+12=62, compared to 124, 50%) or industry experience (13+23=36, compared to 124, 29%). We conclude that our sample included a relatively large proportion of senior individuals, which should increase the reliability of their evaluations of UML consistency rules. Another interesting result is that 24 respondents from academia (24 of 86, 28%) have no industrial experience. This justifies the need for both academic and practitioner respondents in our study, for whom we need strong experience levels in software engineering. Note also a data point that appears to be an outlier: one academic reports no academic experience in Software Engineering; this individual is a mature graduate student, who therefore identified himself/herself as an academic, with industry experience, who, being a student, disclosed no academic experience.

4.2.2.1.5 Respondents' experience in the context of MDSE (Q5)

In this question, we asked about the experience of the participants in the context of MDSE. Results appear in Figure 18, which displays data that has similar semantics to that in Figure 17, in terms of bars and colors. From those results, we see that a clear majority of respondents had 10+ years of academic experience in MDSE (40+10=50, out of 124, 40.3%). On the other hand, fewer respondents disclosed 10+ years of industry experience (9+17=26 out of 124, 21%). Overall, both academics and practitioners disclosed a significant amount of MDSE experience in their field: 46.5% of academics (40 out of 86) disclosed 10+ years of MDSE experience in academia and 44.7% of practitioners (17 out of 38) disclosed 10+ years of MDSE industry experience. When also considering the next range (6 to 10 years) of experience, 71% of academics (40+21=61 out of 86) disclosed 6+ years of MDSE experience in academia and 60.5% of practitioners (17+6=23 out of 38) disclosed 6+ years of MDSE industry experience. Another interesting result is that 40 academic respondents (40 of 86, 46.5%), and only one practitioner respondent disclosed no MDSE industrial experience. (The practitioner disclosing no MDSE experience did report software engineering experience though—Section 4.2.2.1.4.) Similarly, to the discussion about question Q4

earlier, this highlights the need to have both academics and practitioners as respondents in a survey like this and that our sample of respondents had significant expertise in the field we want to study, i.e., MDSE, thereby indicating that our sample of respondents is somewhat representative of the overall target population.

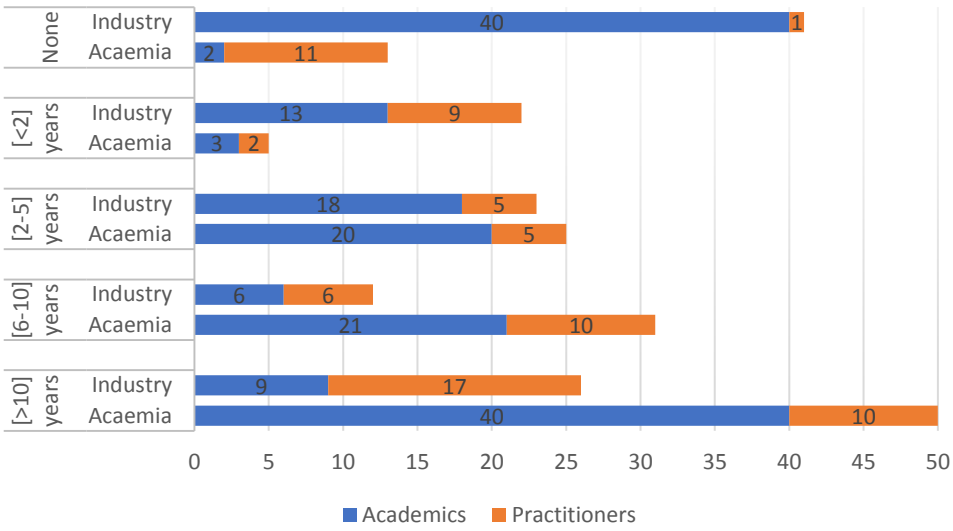

Figure 18. How long have respondents been working in the context of MDSE (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.1.6 Respondents' current position(s) (Q6)

The current positions of respondents are shown in Figure 19. Note that since this was again a multiple-response question, multiple roles could be recorded; in fact, some participants answered to hold two or more roles: e.g., one respondent working in academia and the industry said to be a Professor and a Software Systems/Architect at the same time. As the Figure shows, most of the participants are Professors (51 of 124, 41.1%), followed by Researchers (35 of 124, 28.2%) then Software/Systems Architect (17 of 124, 13.7%). This set of titles described by respondents, especially practitioners, confirm the high level of experience they reported in the previous Section.

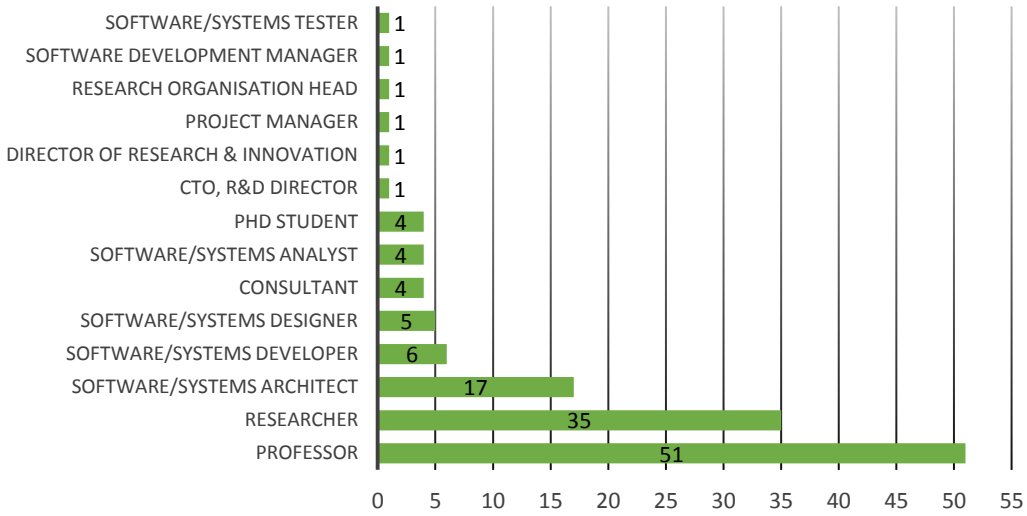

Figure 19. Respondents' current positions

4.2.2.1.7 Number of researchers in respondents' research group – number of employees in respondents' companies (Q7) – Which of them are working in the context of MDSE? (Q8)

For question Q7, we asked about the number of members of the research group, and/or the number of employees of the company that the respondents work in. The bar chart in Figure 20(a) (left hand side) shows that 24 of 38 practitioners (63.2%) reported that the number of employees in their company is larger than 50, and that 57 of 86 academics (66.3%) responded that they work in a research group composed of 5-20 researchers. Only two participants from academia reported to work alone. Following this question, respondents were asked in question Q8 about the number of members in their research groups, or/and the number of employees in their companies, which were working in the context of MDSE. The bar chart of Figure 20(b) (right hand side) shows that most of the industry respondents reported that the number of employees in their company focusing on MDSE are less than 5 and 5-20 (11 out of 38, 28.9% for both of them), and more than 50 (9 out of 38, 23.7%). On the other hand, the majority of participants from academia responded that the number of researchers in their research groups focused on MDSE topics are 5-20 (40 out of 86, 46.5%) and less than 5 (38 out of 86, 44.2%).

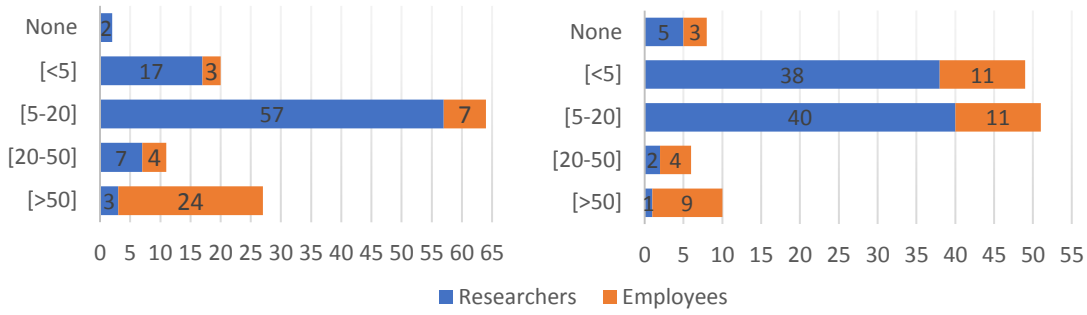

Figure 20. (a) number of researchers in respondents' research group / company (Q7) – (b) which are working in MDSE (Q8)

By comparing academic research groups and employees (size of groups/company and proportion of team members working on MDSE in academia and industry), we conclude that the majority of both academic and industry professional respondents primarily conduct research on MDSE topics.

4.2.2.2 Model Consistency

In this Section, we answer research question RQ2, which is about consistency in the context of MDSE, by means of nine survey questions (Q9 to Q17).

4.2.2.2.1 Frequency of model consistency problems while developing software (Q9)

This question investigates, using a four-point Likert-scale (often, sometimes, rarely, never), how often the participants from academia and industry have detected, studied, fixed, or handled model consistency problems while developing software. We obtained the following results:

- 56 of 124 respondents (45.1%), either from academia (39 of 86, 45.3%) or industry (17 of 38, 44.7%), disclosed that they were often dealing with model consistency problems during software development;
- 48 of 124 (38.7%), either from academia (32, 37.2%) or industry (16, 42.1%) said that they had sometimes faced model consistency problems;
- 15 of 124 (12.1%), either from academia (12, 13.9%) or industry (3, 7.8%), said that they had rarely faced model consistency problems;

- five of 124 (4.03%), three from academia and two from industry, said that they did not have any experience in dealing with model consistency problems. We chose to not exclude those five participants from the survey because they previously answered to have experience in the field of MDSE.

The vast majority (95.97%) of the respondents (119 of 124), both academics (83 of 86, 96.51%) and practitioners (36 of 38, 94.74%), reported that they had faced model consistency issues (at least once) during the process of software development. Moreover, the vast majority of them (104 of 124, 83.8%) reported to have faced consistency issues more than once. We conclude that research on the topic of consistency in MDSE is of practical relevance.

4.2.2.2.2 Importance of the model consistency topic (Q10)

For Q10, we asked respondents how important the topic of model consistency was, on a three-point Likert scale: very important, partially important, not important. For academics, the importance of the topic in their institution was related to whether consistency was a topic being taught. For practitioners, the importance of the topic was related to the degree with which consistency was considered during their MDSE process. Results can be summarized as follows:

- 48 of 124 participants (38.7%), either from academia (32 of 86, 37.2%) or industry (16 of 38, 42.1%) stated that the topic of model consistency in their institution/company is very important;
- 52 of 124 participants (41.9%), either from academia (37 of 86, 43%) or industry (15 of 38, 39.4%) said that the topic is partially important in their institution/company;
- 24 of 124 participants (19.3%), either from academia (17 of 86, 19.7%) or industry (seven of 38, 18.4%) replied that the topic is not important in their institution/company.

Most (80.6%) of the respondents (100 of 124) agreed that the topic of model consistency is at least partially important in their university (69 of 86, 80.2%) or company (31 of 38, 81.5%), respectively.

4.2.2.2.3 MDSE activities where model consistency constraints are used (Q11)

In this question, we asked respondents what MDSE activities, in a predefined list of nine MDSE activities (see the activities in Appendix B, notice that those activities are a superset of the ones presented in Section 3.1.1.4), require model consistency constraints: Figure 21. Q11 was a multiple-response question, where each respondent could record more than one choice. The majority of participants (83 of 124, 66.9%), regardless of whether they are academics or practitioners, said that model transformation is the MDSE activity where they mostly used model consistency constraints. This is closely followed by the verification of consistency and model comprehension, with 82 (66.1%) and 74 (59.6%) out of the 124 participants, respectively. These observations also apply when focusing on either academics or practitioners: the three activities mostly requiring consistency are the process of consistency verification (by checking and detecting consistency issues), model transformation and model comprehension for academics (59, 56, 52, respectively), as well as for practitioners (23, 27, 22, respectively).

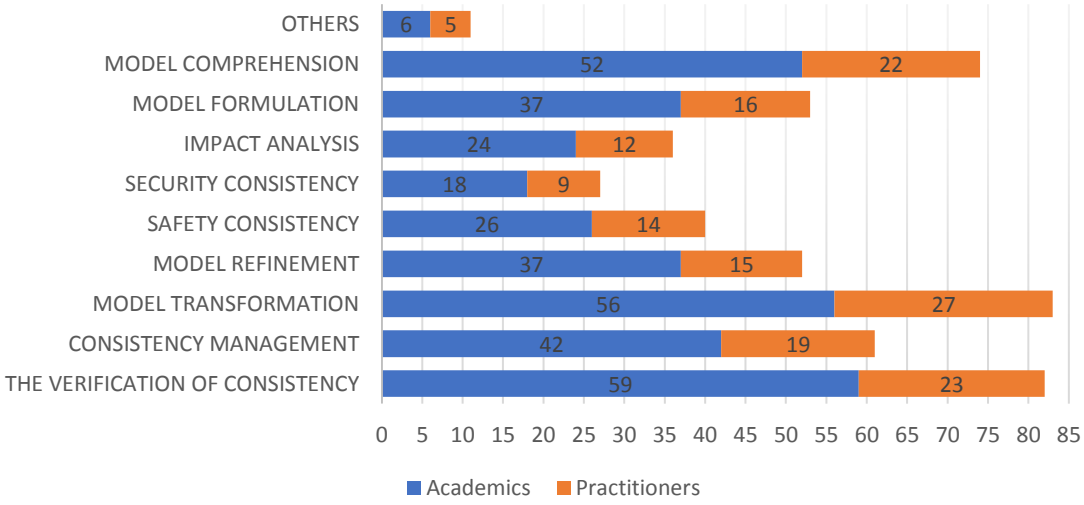

Figure 21. MDSE activities in which models consistency constraints are used (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.2.4 *Respondents' familiarity with the different dimensions and types of Consistency (Q12) – Respondents' experience with consistency issues involving different dimensions and types of consistency (Q13)*

We asked the participants about their familiarity with different dimensions and types of consistency (Q12), and their experience with consistency issues involving different dimensions and types of consistency (Q13). The dimensions and types of consistency used to answer Q12 and Q13 were already presented in Section 3.1.1.4 and were presented to participants along with the questions. Figure 22 shows the vertical mapping of the seven model consistency dimensions, with respondents' familiarity with these notions (right - Q12) using a four-point Likert-scale, and respondents experience with these notions (left - Q13) using a four-point Likert-scale plus N/A. We use a bubble plot to represent multiple dimensions of the results in Figure 22.

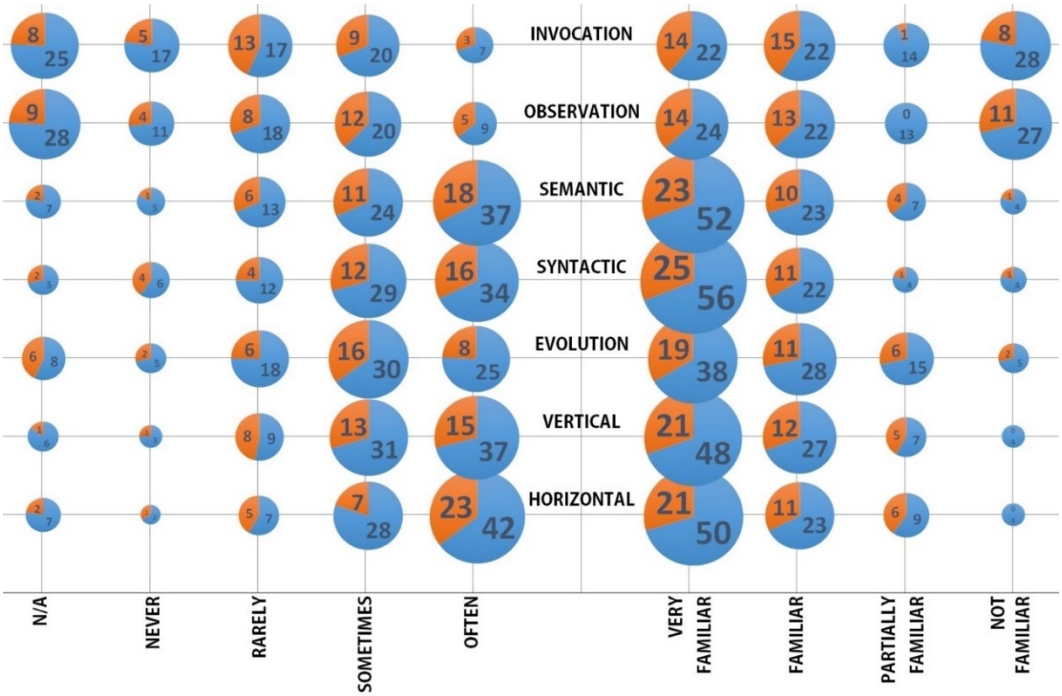

Figure 22. Results about familiarity and experience of respondents regarding to different types and dimensions of model consistency (see accompanying text for a detailed description of the semantics of the figure).

We combined pie charts with the bubble plot (each bubble is a pie chart) to help us describe the two different types of participants involved in this survey: academics (in blue) and practitioners (in orange). The Figure shows, for instance, that 50, i.e., 58.1% (resp. 21, i.e., 55.2%), academics (resp. practitioners) indicated that they were very familiar with the notion of horizontal consistency (right side of Figure) and that 42, i.e., 48.8% (resp. 23, i.e., 60.5%) academics (resp. practitioners)

have often experienced horizontal consistency issues (left side of Figure). The results show that the majority of respondents from both academia and the industry are very familiar (55.6% of them for vertical consistency to 65.3% for syntactic consistency) and have often experienced consistency issues (40.3% of them for syntactic consistency to 52.4% for horizontal consistency) with the following four model consistency dimensions: horizontal, vertical, syntactic, and semantic. The dimensions that the participants are least familiar with are the observation and invocation dimensions.

4.2.2.2.5 How model consistency constraints are specified (Q14)

Q14 was about how model consistency constraints were usually specified in academia and industry: respondents were able to provide multiple answers to the question. In Figure 23 we show that 77 of 124 participants (62.1%), either from academia (48 of 86, 55.8%) or industry (29 of 38, 76.3%), said that the most popular language used to specify model constraints is plain language.

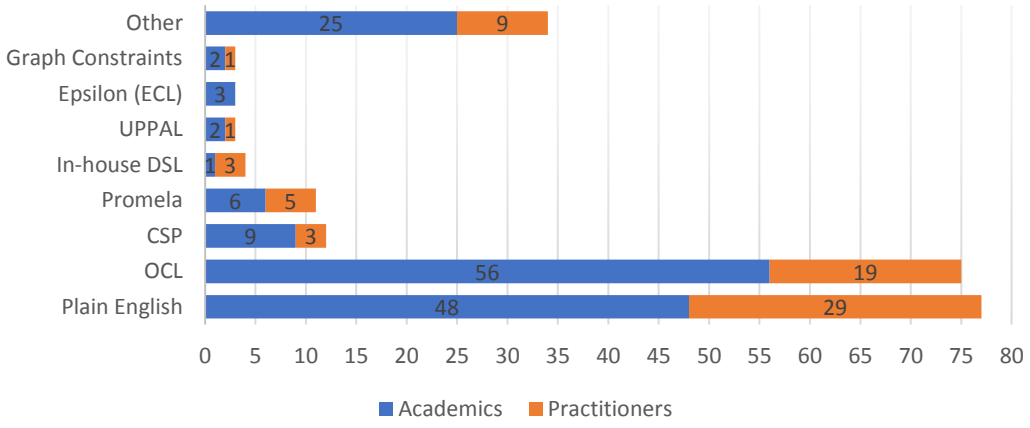

Figure 23. How models consistency constraints are specified (see accompanying text for a detailed description of the semantics of the figure).

This is closely followed by the Object Constraints Language (OCL) [79] with 60.5% of the respondents (75 of 124), either from academia (56 of 86, 65.1%) or industry (19 of 38, 50%). 34 respondents reported other ways to specify model constraints, i.e., Architecture Analysis and Design Language (AADL), Acceleo (an extension of OCL), Alloy, Clock Constraint Specification Language (CCSL), Paradigm (McPal), Prolog, SPIN, Structured Query Language (SQL), Temporal logic, Umple, VIATRA/IncQuery Pattern Language, W3C Shape Expression, Xtext,

SPASQL (an extension of SQL). These results suggest that OCL is the language mostly used by academics, whereas plain language is the most used language by practitioners.

4.2.2.2.6 Tools used to check model consistency (Q15)

Participants were asked which tools they used to check model consistency in academia and the industry. This was again a multiple-response question; so multiple tools could be recorded. As presented in Figure 24, 52.4% of respondents (65 of 124) said they used tools that had been developed in-house: either in academia (43 of 86, 50%) or industry (22 of 38, 57.9%). This is closely followed by the use of open source tools (50%, 62 of 124), manual checking (42.7%, 53 of 124), and then commercial tools (33.1%, 41 out of 124). Finally, 11 respondents (8.8%) indicated other tools to check model consistency, such as QVT, ATL, Epsilon, Prolog, Symbolic Model Verification (SMV), Failures-Divergences Refinement (FDR), SPIN, Epsilon Validation Language (EVL), and VIATRA/EMF-IncQuery. Overall, the vast majority of respondents rely on automated verification of consistency rules.

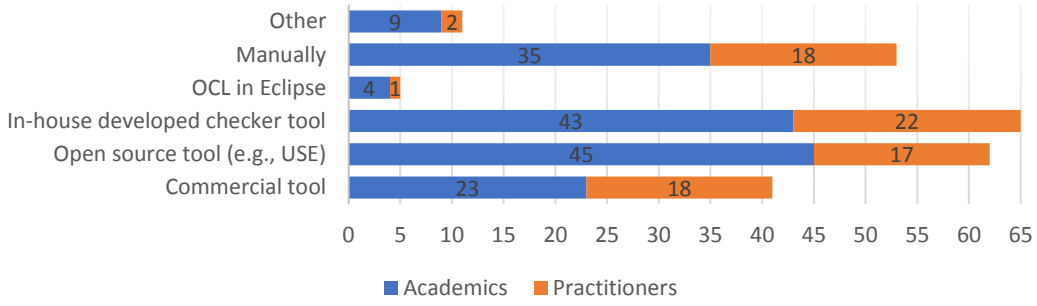

Figure 24. Tools that are used to check models consistency (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.2.7 Tools that are used to represent models (Q16)

In this Section, we present the answers of respondents regarding the tools they used to represent models. Q16 is again a multiple-response question where multiple modeling tools could be recorded. Figure 25 shows that the majority of respondents (83 of 124, 66.9%), either in academia (61 of 86, 70.9%) or industry (22 of 38, 57.8%), use Eclipse-based tools. This is followed by tools developed in-house (37.1%, 46 of 124). Moreover, 24 out of 124 respondents indicated the following tools in answering this question: ADOXX, INTEGRANOVA, Kevoree Modelling

Framework, UMLet, MS PowerPoint (only for sketch), Latex, Academic Signavio, AADL, Protege for OWL, RDF, MetaEDIT, Simulink, UPPAAL, Lucid Charts, MEGA HOPEX, OmniGraffle, Google Docs, CAPELLA, and Genmymodel.

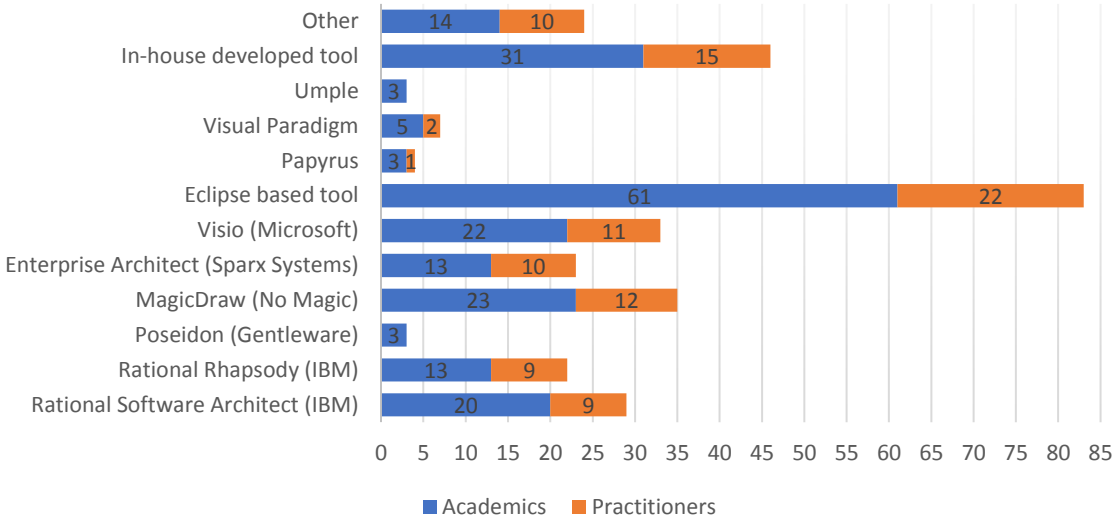

Figure 25. Tools that are used to represent models (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.2.8 Modeling languages that are used to develop models of software systems (Q17)

Q17 asked what modeling languages the respondents used to develop models of software systems. This was again a multiple-response question. Not surprisingly (Figure 26), the majority of respondents (106 of 124, 85.4%), either in academia (76 of 86, 88.3%) or industry (30 of 38, 78.9%), said they used UML. The second most selected response was "Other DSL (domain specific language)" with 61.2% of the respondents (76 of 124), either from academia (48 of 86, 55.8%) or industry (28 of 38, 73.6%).

Apart from the specific choices the question provided (Figure 26), respondents indicated 21 other modeling languages, including Layered Queuing Networks, Stochastic Petri Nets, Fault Trees, Capella CTT, MB-UID, ADLs OO-Method Kevoree Modelling Framework, KAOS goal diagram, CDM process diagram, Umple, MERODE, Acme, OWL2 ER, OPM, AUTOSAR, and CVL. Only 24 of 124 respondents (19.3%) indicated that they do not use any specific modeling language, and they model informally through sketches. Although some of these "Other" responses might be included in the set of provided choices (i.e., AADL can be under "Domain Specific Language

(DSL)"), we chose to report them because participants specifically named the usage of these modeling languages.

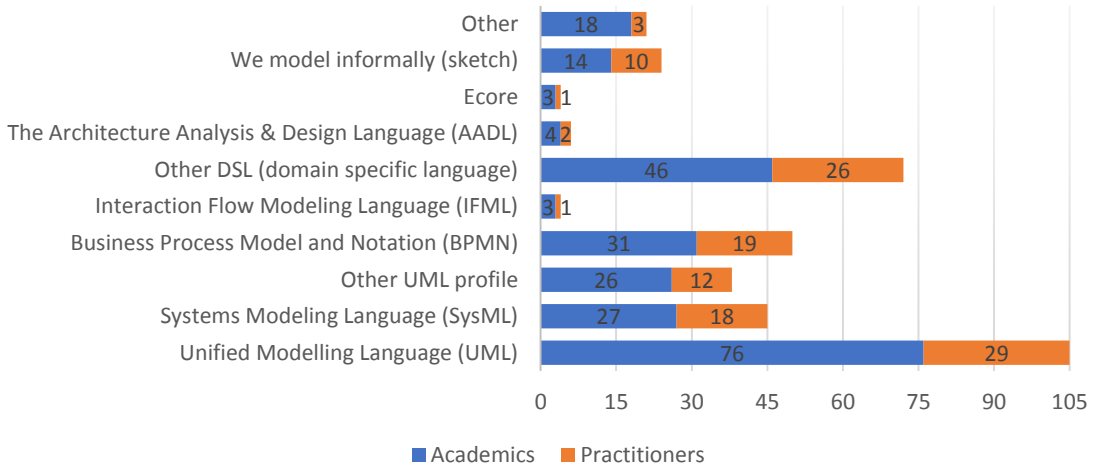

Figure 26. Modeling languages that are used to represent models (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.2.9 Combining Q8 with Q14-Q17

In order to show an overview (Figure 27) of how MDSE experts specify, check, and represent model consistency, we combine the results of Q8 (number of researchers in respondents' research group/company who are working in MDSE) with the most chosen answers of Q14 (most selected way to specify model consistency constraint, i.e. plain language and OCL), Q15 (most used tools to check model consistency issues, i.e. commercial, open source, in-house developed, manually), Q17 (most used languages to represent models, i.e. UML, SysML, other UML profile, BPMN, and DSL), and Q16 (most used tools to represent models, i.e. Rational Software, Rational Rhapsody, Magic Draw, Enterprise Architect, Visio, Eclipse, and in-house developed). Q14-Q17 were multiple-response questions, therefore, in Figure 27 for each of these four questions, the reader should not expect percentages to sum up to 100%. For instance, the 29 responses from academics with a MDSE team between 5 and 20 members has a percentage of 73% because it is calculated on the 29 responses out of 40 total respondents for this category (see also Section 4.2.2.1.7). Figure 27 presents the data of respondents from academia in light blue (upper part), and from industry in light orange (bottom part), respectively.

Figure 27 points to a number of interesting and perhaps not expected observations. Regarding Q14 (first two columns – reading from left to right), regardless of the number of MDSE researchers,

OCL and plain English are equally used (except for [5-20] responses with 73%) in academia, while, industry prefers to specify constraints in plain English.

In Q15 (column three to six in Figure 27), we can see that in academia the majority of respondents with a small (less than 5) or a medium (between 5 and 20) number of MDSE researchers in their group said to check model consistency with open source and in-house developed tools; respondents from the industry (especially the ones working in company with more than 50 MDSE experts) prefer to check model consistency with in-house developed tools. Also, industry tends to use commercial tools more so than academia, which is to be expected (commercial tools are perceived as more reliable but can be expensive). Another observation with Q14 and Q15 is that since industry tends to rely more on plain language than OCL, it is not surprising that industry relies also on manual verification.

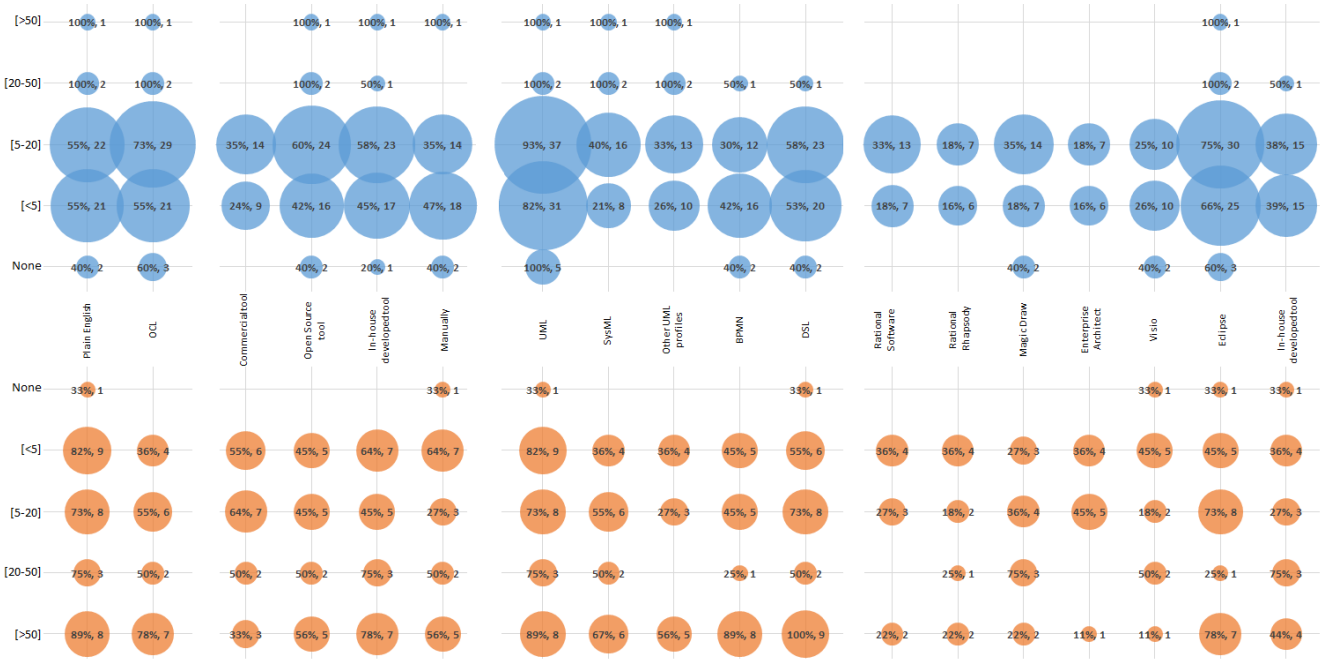

Figure 27. Combining Q8 with Q14, Q15, Q17 and Q16 (see accompanying text for a detailed description of the semantics of the figure).

From column seven to eleven, we observe that DSLs are far more present than we anticipated. UML is still universally accepted as the modeling language of choice, but an interesting finding of the survey was a perhaps surprisingly high use of DSLs in the industry and especially by respondents working in company with more than 50 MDSE experts. This suggests that those

companies prefer to develop DSLs for well-understood domains instead of applying the whole OMG standard with UML, SysML, and BPMN.

Regarding Q17 (column 12 to 18), respondents generally did not stick to only one tool to represent models but rather pick and choose a combination of tools. Nevertheless, the majority of respondents from academia prefer Eclipse tools, similarly to respondents from the industry who work in companies with 5-20, and with more than 50 MDSE experts.

The wide diversity of languages and tools reported in our survey suggests that, more than ten years after the OMG brought out its MDA specification, practitioners do not feel there is a need for one-fit-all MDSE modelling language. Nevertheless, the data presented in this Section suggest some differences of preferences between academia and the industry. In academia respondents prefer to use 1) OCL to define consistency constraints, 2) in-house/open source tools to check model consistency, 3) UML as model language, and 4) Eclipse as tool to implement the models; on the other hand, in industry the choices are 1) Plain English, 2) in-house/commercial tools, 3) UML/DSL, and 4) Eclipse.

4.2.2.3 UML Consistency

In this Section, we discuss results about UML consistency to answer RQ3. Recall (Section 4.2.1.1.2) that Q17 was a decision point in the survey. The survey ended with Q17 for those who said they were not using UML: we collected 18 out of 124 such responses (14.5%). For the others (106 respondents, 76, i.e. 88.4%, from academia and 30, i.e. 78.9%, from the industry), who said they were using the UML, the survey continued with questions 18 to 21.

4.2.2.3.1 UML diagrams mostly involved in consistency issues (Q18)

Not surprisingly, since these are the most used UML diagram [58], Figure 28 shows that the UML diagram that was most involved in consistency issues according to the 106 respondents was the class diagram (98, 92.4%), both in academia (70 of 76, 92.1%) and industry (28 of 30, 93.3%). This is followed by the sequence diagram, the state machine diagram, and the activity diagram, respectively at 71.7%, 63.2%, and 43.4% of the respondents' responses. Figure 28 illustrates that each of the 14 different kinds of UML diagrams was selected at least once by the participants. The

diagram with the least reported involvement with consistency issues was the Interaction Overview diagram with four out of 106 respondents choosing it (3.7%).

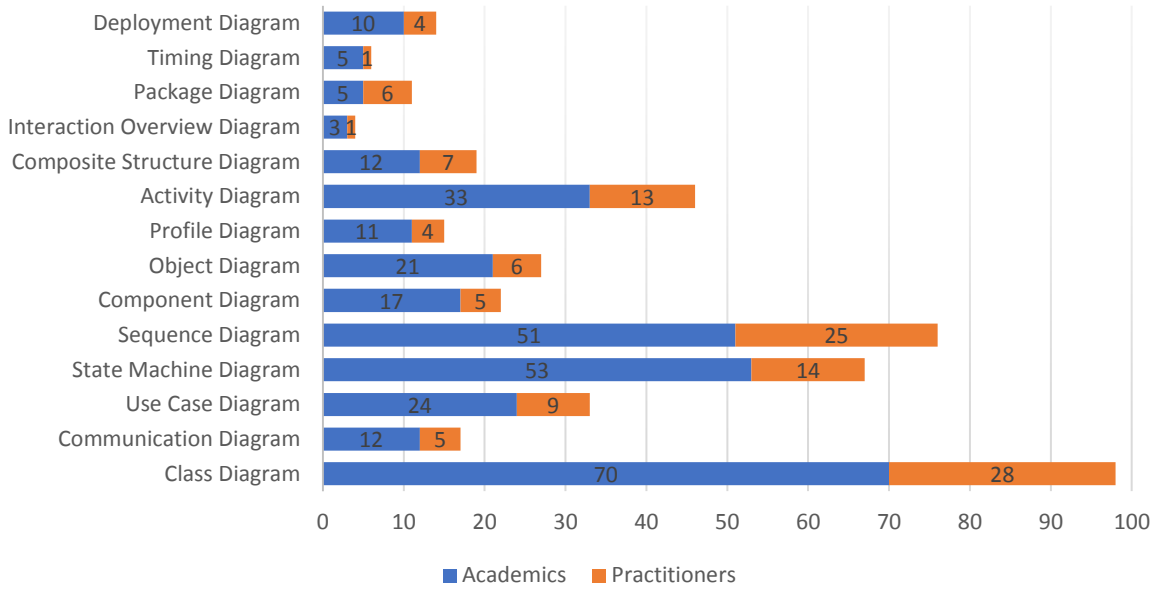

Figure 28. UML diagrams that are mostly involved in consistency issues (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.3.2 Studying, detecting and handling UML model inconsistencies (Q19)

Q19 asked the 106 participants (identified in Q17) how they detect, study, and handle UML model inconsistencies. This is again a question where multiple answers could be recorded. Answers did not show any specific trend (Figure 29); 62 respondents (58.4%) said to check consistency constraints in addition (extra) to what is supported by the tool they are using to create their models, 61 (57.5%) said to specify their own UML consistency constrains for their UML models, and another 53 (50%) said to check the UML well-formedness constrains presented by the Object Management Group (OMG) in the UML specification standard. Moreover, 19 participants suggested other ways to detect, study, and handle UML models consistency, including: EVL, GEMOC Studio, Papyrus, OCL in Ecore, Paradigm (McPal), Umple, JMERMAID, Alloy, Magic Draw, MEGA HOPEX, and Rational Studio Architect.

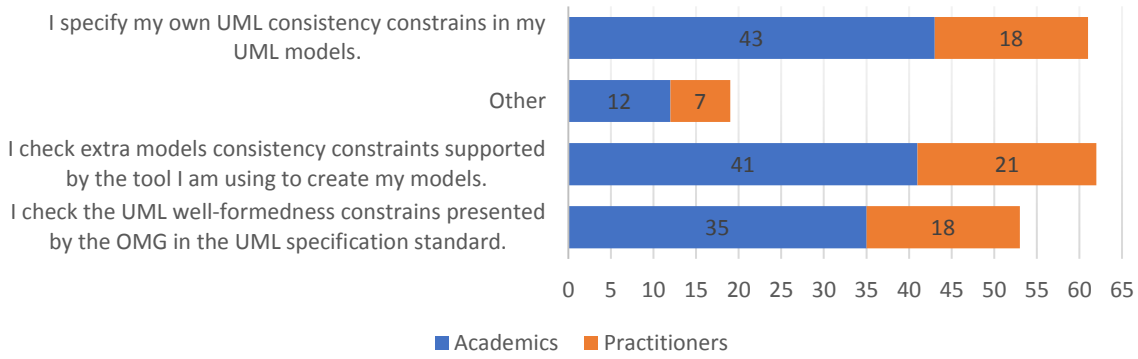

Figure 29. How UML model inconsistencies are studied, detected, and handled (see accompanying text for a detailed description of the semantics of the figure).

4.2.2.3.3 UML Consistency Rules (Q20 and Q21)

In this Section, we present the answer to our fourth research question (Section 4.2), which is about how relevant our set of 116 UML consistency rules identified in Section 3.2 is.

In order to answer this question, we asked the 106 respondents whether they felt that they understood the definition of each consistency rule presented to them (Q20) and whether they believed the rule was to be enforced (Q21). Since we assumed answers to Q20 and Q21 should not depend on whether the respondent is an academic or a practitioner, we asked Q20 and Q21 only once to participants that identified themselves as both an academic and a practitioner. This means that Q20 and Q21 were asked to 61 respondents from academia, 15 from the industry and 15 from academia and the industry, resulting in a total of 91 responses (106-15) to Q20 and Q21. As reported in Section 4.2.1.2, each of the respondents was asked to answer the two questions for at least five of the 116 UML consistency rules we discussed in Section 3.2. As discussed earlier, once a respondent answered on a group of five UML consistency rules, s/he could choose to answer on another five rules and so on or end the questionnaire.

Table 11 presents the results of Q20 and Q21: the first column represents the rule ID# that can be used as a reference to access the rule description in the Appendix A; the second column shows the total number of respondents who checked a specific rule; the third column indicates whether the respondents felt s/he understood the rule (Q20); the fourth column indicates how often the rule should be enforced according to the respondent (Q21).

In the end, for Q20 we collected 751 answers (sum of the "Total" columns) for the 116 UML consistency rules. At least six respondents evaluated each rule. In 80.8% of the answers (607 out of 751, sum of the "Yes" columns), the respondents said that they understood the rules, while in the remaining 19.2% of the answers (144 out of 751, sum of the "No" columns), the respondents said they did not understand the rules. The understandability of each of the 116 rules was checked by six to eight different respondents (recall our execution procedure). We observed that each of the 116 rules was understood, on average, by more than five different respondents, 39 out of 116 rules, i.e. 33.6%, were understood by 100% of the respondents that checked a given rule (see cells colored in blue in Table 11), while 12 rules, i.e. 10.3%, were not understood by the majority of the respondents (see cells colored in orange in Table 11). 89.7% of the rules (104 out of 116) were understood by the majority of the respondents.

Table 11. Q20 and Q21 results for each of the 116 consistency rules

Rule #	Total	Understood the rule? (Q20)		Rule to be enforced? (Q21)			Rule #	Total	Understood the rule? (Q20)		Rule to be enforced? (Q21)		
		Yes	No	Always	Sometimes	Never			Yes	No	Always	Sometimes	Never
1	6	5	1	2	2	1	59	6	5	1	1	3	1
2	6	5	1	2	2	1	60	6	4	2	1	1	2
3	6	6	0	2	2	2	61	7	6	1	2	3	1
4	6	4	2	0	1	3	62	7	5	2	1	3	1
5	6	4	2	1	1	2	63	7	6	1	4	2	0
6	7	6	1	2	2	2	64	7	7	0	6	1	0
7	7	6	1	2	2	2	65	6	6	0	2	3	1
8	7	4	3	0	2	2	66	6	6	0	2	4	0
9	7	5	2	3	1	1	67	6	6	0	5	1	0
10	7	6	1	2	2	2	68	6	5	1	2	3	0
11	7	4	3	1	3	0	69	6	5	1	1	4	0
12	7	7	0	4	3	0	70	6	6	0	3	3	0
13	7	6	1	4	2	0	71	6	4	2	1	3	0
14	7	5	2	2	2	1	72	6	6	0	2	4	0
15	7	6	1	3	3	0	73	7	6	1	5	1	0
16	7	6	1	2	3	1	74	7	4	3	3	1	0
17	7	6	1	1	5	0	75	7	7	0	7	0	0
18	7	6	1	3	3	0	76	7	7	0	7	0	0
19	7	6	1	3	0	3	77	7	5	2	5	0	0
20	7	6	1	2	3	1	78	6	6	0	5	1	0
21	6	5	1	2	3	0	79	6	6	0	3	3	0
22	6	4	2	2	2	0	80	6	5	1	3	2	0
23	6	2	4	0	2	0	81	6	6	0	5	1	0
24	6	6	0	3	2	1	82	6	6	0	6	0	0

Rule #	Total	Understood the rule? (Q20)		Rule to be enforced? (Q21)			Rule #	Total	Understood the rule? (Q20)		Rule to be enforced? (Q21)		
		Yes	No	Always	Sometimes	Never			Yes	No	Always	Sometimes	Never
25	6	4	2	0	4	0	83	6	6	0	4	1	1
26	6	4	2	0	4	0	84	6	6	0	4	1	1
27	6	6	0	5	1	0	85	6	6	0	4	1	1
28	6	5	1	3	1	1	86	6	6	0	5	1	0
29	6	6	0	2	3	1	87	6	6	0	5	1	0
30	6	5	1	1	4	0	88	6	4	2	3	0	1
31	6	4	2	1	3	0	89	6	5	1	5	0	0
32	7	7	0	3	4	0	90	6	5	1	4	1	0
33	7	7	0	3	4	0	91	6	3	3	1	2	0
34	7	4	3	1	3	0	92	6	3	3	1	2	0
35	7	6	1	3	3	0	93	6	5	1	4	1	0
36	7	7	0	3	4	0	94	6	5	1	5	0	0
37	7	6	1	5	1	0	95	6	5	1	2	1	2
38	7	6	1	3	3	0	96	6	5	1	2	2	1
39	7	6	1	4	2	0	97	6	4	2	2	2	0
40	7	7	0	6	1	0	98	6	5	1	4	1	0
41	7	5	2	2	3	0	99	6	5	1	4	1	0
42	8	5	3	2	3	0	100	6	2	4	1	1	0
43	8	3	5	0	3	0	101	6	1	5	1	0	0
44	8	6	2	2	4	0	102	6	4	2	3	1	0
45	8	3	5	0	3	0	103	6	2	4	1	1	0
46	8	5	3	3	2	0	104	6	3	3	1	2	0
47	6	5	1	4	1	0	105	6	2	4	2	0	0
48	7	7	0	5	2	0	106	6	1	5	1	0	0
49	7	7	0	6	1	0	107	6	4	2	2	2	0
50	7	6	1	3	2	1	108	6	5	1	3	1	1
51	7	7	0	3	3	1	109	6	6	0	5	0	1
52	7	3	4	2	0	1	110	6	6	0	6	0	0
53	7	7	0	3	3	1	111	6	5	1	3	1	1
54	7	7	0	6	1	0	112	6	5	1	4	1	0
55	7	7	0	4	3	0	113	6	6	0	2	2	2
56	7	7	0	3	3	1	114	7	7	0	3	3	1
57	6	5	1	1	3	1	115	7	7	0	4	1	2
58	6	5	1	0	3	2	116	6	6	0	4	1	1

To answer Q21, participants who understood the rules (in Q20) were asked if each rule were relevant to be: a) enforced always in any UML development process, b) enforced in some situations, c) never enforced. Following Q20, we therefore collected 607 answers regarding Q21. Results of Q21 can be summarized as follows (Table 11):

a) 53.9% of the time (327 out of 607), respondents considered the rules to be enforced in all UML/model systems ("Always" in Table 11);

b) 36.7% of the time (223 out of 607) respondents considered the rules to be enforced only in specific situations ("Sometimes" in Table 11);

c) 9.4% of the time (57 out of 607) respondents considered the rules not to be enforced ("Never" in Table 11).

So, 90.6% of the time, rules were considered to be relevant to enforce in at least some (or all) situations (327 times always relevant, 223 times relevant in some situation). The majority of respondents considered that (Table 11): 52 rules, i.e. 44.8%, should always be enforced in all UML/model systems (cells colored in green); 34 rules, i.e. 29.3%, were to be enforced in specific situations (cells colored in red); three rules, i.e. 2.6%, were not to be enforced (cells colored in grey). In addition, Table 11 highlights that 74 rules, i.e. 63.8%, were not consider by any one as never to be enforced (cells colored in yellow).

To summarize Q20 and Q21, an average of five different respondents checked each of the 116 rules, and said that 1) they understood the rule, 2) 90.6% of them were to be enforced in UML/model system development.

Starting from the results collected in Table 11, we present in Figure 30 a Venn diagram where the 116 rules are divided into three major sets along with their intersections according to the most chosen answers for each rule in Q21.

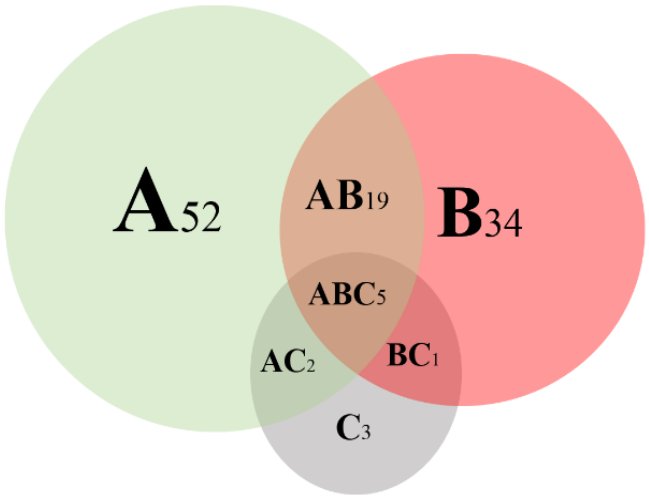

Figure 30. The set of 116 UML consistency rules most selected in Q21

Set A in Figure 30 is made of rules for which a majority of answers indicated that the rule should always be enforced, set B is made of rules for which a majority of answers indicated that the rule should sometimes be enforced, and set C is made of rules for which a majority of answers indicated that the rule should never be enforced. The intersections between the sets, i.e. AB, ABC, AC, and BC, indicate an equal split of opinions: e.g., AB is the set of rules that are equally considered to be always and sometimes enforceable. The abovementioned four sets represent the 27 out of 116 (23.2%) rules for which respondents had contradicting opinions (i.e. we were not able to find a majority consensus in their responses). Some interesting findings from this set of 27 rules are that 1) seven of the 27 rules involved only SMD which represent the 50% of the total rules (i.e. 14 rules) identified for this diagram (see Table 10); and 2) eight of the 27 rules were semantic rules, which is 38.1% of the total 21 semantic rules identified in this work (see Table 10).

105 (52+34+19) of the 116 UML consistency rules were considered to be enforceable in UML models, either always or in some situations, by the majority of respondents. Set A represents the 52 rules out of the 116 rules (44.8%) for which the majority of respondents said the rule should always be enforced. Most of those rules are Syntactic (48 out of 52, 92.3%) and Horizontal (49 out of 52, 94.2%) and involve the Class Diagram (38 out of 52, 73.1%) and the Sequence Diagram (11 out of 52, 21.1%). This last finding was not surprising considering the results obtained on questions on consistency dimensions/types (Section 4.2.2.2.4) and UML diagrams most involved in consistency issues (Section 4.2.2.3.1).

4.2.2.3.4 Combining Q4-Q5 with Q20-Q21

According to the survey results, 80.8% of the respondents said they understand the rules and 90.6% of the times respondents considered the rules should be enforced. In this Section, we analyze the distribution of the responses of Q20 and Q21 according to the experience of the respondents in SE and MDSE (respectively represented in yellow and green in Figure 31). In order to do that we built the bubble pie chart plot. Figure 31 is structured as follow: 1) the left side of the Figure shows data of respondents with experience in industry, and the right side presents the data of respondents from academia; 2) the first two rows (i.e. YES and NO) of the Figure present the data obtained for Q20, the last three rows refers to Q21 where respondents considered the rules a) to be enforced in all UML/model systems (“A” in Figure 31); b), to be enforced in a specific situation (“S” in Figure

31), c) not to be enforced ("N" in Figure 31); 3) the experience of the respondents in SE and MDSE is represented in each pie chart of Figure 31 respectively with the colors yellow and green.

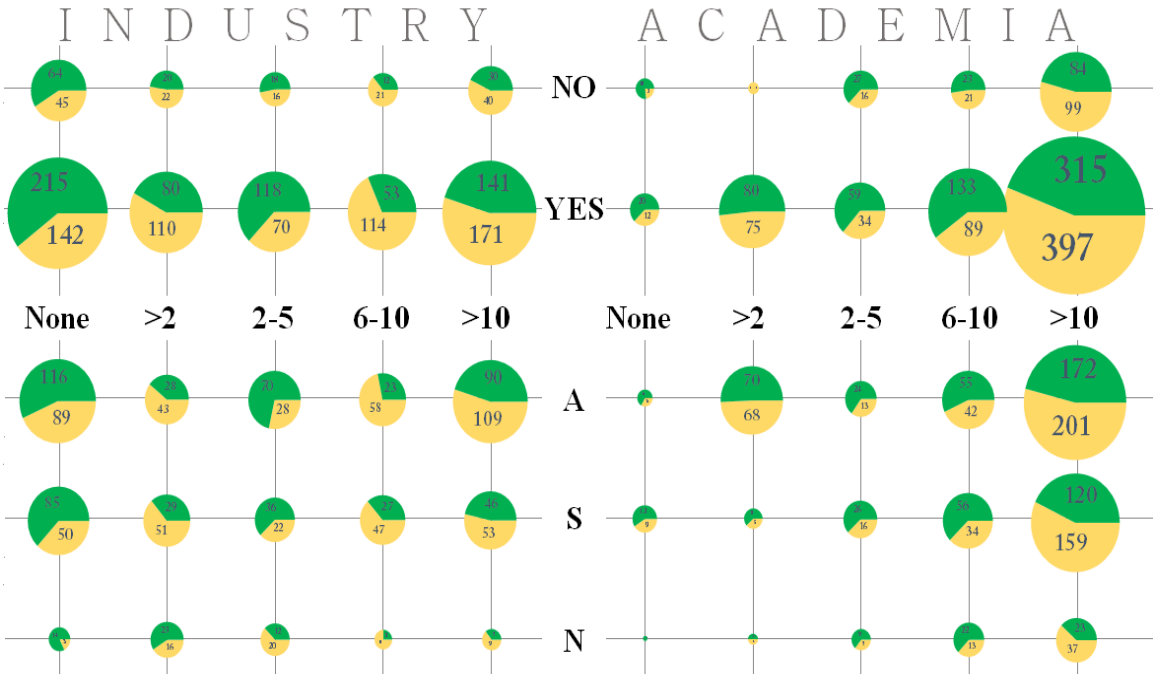

Figure 31. Combining Q4-Q5 with Q20-Q21 (see accompanying text for a detailed description of the semantics of the figure).

We confirm that most of the respondents had more experience in SE (see Section 4.2.2.1.4) than MDSE (more yellow than green), which is not surprising given that the topic of SE goes back to the end of the 60s [80], while UML, Model Driven Architecture, and MDSE are relatively more recent topics [57]. The 116 rules were understood (315 in MDSE and 397 in SE out of 607, with 51.9% and 65.4% respectively) and identified as rules to enforce in all UML/model systems (172 in MDSE and 201 in SE out of 327, with 52.6% and 61.5% respectively) by respondents with more than 10 years of experience in academia. On the other hand, the 116 rules were understood (312 in MDSE and 355 in SE out of 607, with 51.4% and 58.5% respectively), and were identified as rules to enforce in all UML/model systems (183 in MDSE and 195 in SE out of 327, with 56% and 59.6% respectively) by respondents with at least 2 years of experience (results of "2-5", "6-10", and ">10" categories were summed up) in the industry.

Figure 31 also shows that some rules were checked by some practitioners with no experience in MDSE (215 out of 607, 35.4%) or in SE (142 out of 607, 23.4%). We checked and confirmed that all those respondents had at least 2 years of academic experience in SE and/or MDSE.

4.2.3 Conclusions of Section 4.2

Section 4.2 provided the results of a survey that investigated how model consistency is handled in the context of MDSE and UML with experts from academia and industry. The survey intended to (1) evaluate the diffusion and importance of model consistency issues; (2) validate the 116 UML consistency rules identified in Sections 3.1 and 3.2. In order to achieve our two main objectives, we formulated four research questions. The survey was conducted online, from 15 May 2016 to 30 June 2016, and a total of 124 complete responses were collected. The findings presented in the previous Sections provided information on different ways to reason about consistency in the context of MDSE. The analysis of the questionnaire data has involved different descriptive statistics and graphs to bring some overall awareness to the topic of MDSE/UML consistency. In the following Sections (4.2.3.1-4.2.3.4), we present the main findings grouped in the four research questions introduced at the beginning of Section 4.2. Other final comments and observations regarding Section 4.2 will be presented in Chapter 6.

4.2.3.1 RQ1: Demographic information of the survey participants.

The survey collected 124 valid questionnaires where the majority of respondents had a post graduate degree. All the respondents had at least 2-5 years of experience working in the context of Software Engineering (SE) and Model Driven Software Engineering (MDSE) in academia and/or in industry. Most of the participants indicated that they were professors and researchers.

We believe that these results indicate that respondents were competent to answer our survey and that they are somewhat representative of our target population, i.e. software engineers concerned with MDSE consistency issues.

4.2.3.2 RQ2: How consistency problems in the context of MDSE are handled and detected

The majority of the respondents said that they had faced model consistency issues during the process of software development, and they agreed that the topic of model consistency was important in their university or company. The majority of respondents said that model transformation, the verification of consistency, and model comprehension were the MDSE activities where they mostly used model consistency constraints. Participants said that they were

familiar with consistency issues with the following four model consistency dimensions: horizontal, vertical, syntactic, and semantic. Syntactic consistency was the model consistency dimension with which the majority of respondents said they were familiar. The results of this survey showed that respondents: a) develop model systems mostly using Eclipse-based tools and "in house developed tools"; b) specify model constraints in plain English and the OCL [79] ; c) check model consistency developed in-house and open source tools. Not surprisingly, the majority of respondents said that they used the UML as modelling language, and DSL (domain specific language) as well.

4.2.3.3 RQ3: How UML inconsistencies are handled

Participants reported that the UML diagrams most involved in consistency issues was the class diagram followed by the sequence and state machine diagram. When participants were asked how they dealt with UML inconsistencies, they did not show a clear preference, and each of the following options were chosen at least once by half of the respondents: 1) checking model consistency constraints that were supported by the tool they were using, 2) specifying their own UML consistency constrains, and 3) checking the UML well-formedness constrains presented by the OMG in the UML specification standard.

4.2.3.4 RQ4: Relevancy and understandability of the 116 UML consistency rules

The 116 UML consistency rules identified in Section 3.2 were selected and analyzed in depth by following a precise selection protocol. We asked participants if they understood those 116 rules, and 80.8% of the times, respondents said they understood the specific rule. The rules understood were used to ask respondents about the relevance of the entire set of 116 rules. The set of 116 UML consistency rules was checked by 91 different respondents and they were considered to be enforced 90.6% of the times. An average of five different respondents checked each of the 116 rules, and said: 1) they understood the rules, and 2) the rules were relevant to be enforced in UML models. We finally identified a subset of 52 UML consistency rules which should always be enforced in all UML/model systems. This set of 52 UML consistency rules will be used to carry out the last validation step of this Ph.D. thesis presented in the next section.

4.3 UML consistency rules: a case study with Eclipse Papyrus models

Starting with the set of 52 UML consistency rules identified in Section 4.2, the main research goal of this Section is to investigate to what extent existing UML models satisfy those rules. To do that, in this Section we describe an exploratory flexible indirect case study [22] to evaluate the 52 UML consistency rules on UML models where we report on: 1) the encoding of the 52 UML consistency rules [17] as invariants in the Object Constraint Language (OCL) [79]. From here on in this Section, the OCL consistency invariants will be referred to simply as the OCL rules; 2) the selection of the UML models to use; 3) the methodology used to import the UML models into the Eclipse Papyrus tool[9] to verify them with the OCL rules; and 4) the results obtained by verifying the UML models with the OCL rules.

In this case study we conduct a type of verification of UML consistency (see definition of *Verification* in Section 3.2.1.3) where we focus on the detection of inconsistencies between (or in a single) UML diagram(s) of a UML model.

The next Sections are structured as follows: Section 4.3.1 provides a summary of the related work; the case study along with its results is presented in Section 4.3.2; finally, our conclusions are shown in Section 4.3.3.

4.3.1 Related work – Section 4.3

To the best of our knowledge, we are not aware of any previous work that present a case study involving UML consistency rules, which were systematically collected in the literature (Sections 3.1 and 3.2), evaluated by MDSE experts from the industry and academia (Sections 4.1 and 4.2), and encoded in OCL and checked systematically on a relatively large number of open-source UML models. Nevertheless, as related work, we decided to collect papers that discuss OCL rules used to check consistency between UML diagrams. Note that we therefore omit related work that discusses consistency (rules) of (UML) models. For a more complete discussion of this latter topic, we refer the reader to the previous Sections 3.1 and 3.2. In Table 12, we present a summary of the five pieces of work that fit the description of related works for the work presented in this Section. For each one, the Table indicates the year when the work was published, whether the paper

[9] https://eclipse.org/papyrus/

involves any case study, the number of OCL rules discussed in the paper, and finally the number of model projects involved.

Table 12: Summary of related work

Authors	Year	Case Study	OCL rules	Models
Reder and Egyed [81]	2013	Yes	20	29
Gogolla et al. [82]	2015	Yes	10	18
Czarnecki and Pietroszek [83]	2006	Yes	1	1
Liu et al.[84]	2002	No	1	none
Sapna and Mohanty [85]	2007	No	1	none

Reder and Egyed [81] presented an approach for identifying how design rules cause a given inconsistency and what model elements are involved. They demonstrated that their approach identifies causes correctly by validating 29 UML models against a set of 20 OCL design rules, where three out of 20 rules were considered in the set of 52 consistency rules (see Section 4.2) as well. The rest of 17 rules were collected in our previous work presented in Appendix A as UML consistency rules, but the MDSE experts of our survey described in Section 4.2 did not consider those rules should be enforced in every UML model. For this reason, those 17 rules were not included in this work.

Gogolla et al. [82] report on a middle-sized case study, in which the consistency of a UML class model is checked. The class model restrictions are expressed by UML multiplicity constraints and explicit, non-trivial OCL rules. Their approach automatically constructs a valid system state that shows whether a class model can be instantiated and failure to instantiate the model while satisfying OCL rules is an indication of inconsistency.

Czarnecki and Pietroszek [83] use OCL to define well-formedness rules for the verification of feature-based model templates that are analyzed by a SAT solver. The rules can be evaluated against a template using a template interpretation for OCL. They presented a tool prototype which was tested on a business model for an e-commerce platform.

Liu et al. [84] describe a rule-based solution to detect software design inconsistency problem. They characterized classes of inconsistency that occur in software design. They defined a production system language and rules specific to software designs modeled in UML. Using this approach, they detect inconsistencies, notify the users, recommend resolutions, and automatically fix the inconsistency during the design process.

Sapna and Mohanty [85] deal with structural inconsistencies between use case, activity, collaboration, state machine and class diagrams by using OCL rules converted to SQL triggers applicable across Tables that store the UML diagrams.

The work presented in this Section 4.3 differs from the above ones in several ways. First of all, we use a larger set of consistency rules than any other previously published study. Our set of rules has been systematically identified and validated by MDSE experts. We also evaluated the rules on a larger set of models. Also, our objective—of studying the extent of inconsistency—is different from, though complementary to, previously published work: e.g., Reder and Egyed [81] focused on fast evaluations of rules, and Liu et al. [84] focused on resolutions of inconsistencies.

4.3.2 Case study

The research process of a case study can be characterized as fixed or flexible [86, 87]. In a fixed design process, all parameters are defined at the launch of the study, while in a flexible design process key parameters of the study may be changed during the course of the study [88]. In this Section, we describe an exploratory flexible indirect case study for the following reasons: a) *exploratory* because we want to find out what is happening in this context (i.e. how our OCL rules are checked on UML models), seek new insights regarding inconsistencies in different UML diagrams types, and generate ideas and hypotheses for new research [22]; b) *flexible* because our design process key parameters were modified during the course of the study [22], i.e. we excluded some of the OCL rules, UML models, and inconsistencies (see Sections 4.3.2.1.1, 4.3.2.1.2, and 4.3.2.1.3 respectively for more details); c) *indirect* because we carry out the case study by using a tool instrumentation [89] such as Eclipse Papyrus[10]. In this Section, we present the five major process steps to carry out our case study by following the guidelines for conducting and reporting

[10] https://www.eclipse.org/papyrus/

on case study research in software engineering [88]: 1) Case study design, where the main objective is defined and the case study is planned (Section 4.3.2.1); 2) Preparation for data collection, which defines the procedures and protocols for data collection (Section 4.3.2.2); 3) Collecting evidence (i.e. execution with data collection on the studied case) and results (Section 4.3.2.3); and 5) Analysis of the collected data (Section 4.3.2.4). Dedicating a significant amount of space to the discussion is important [88] as this facilitates interpretation of results and is paramount to allow replications.

4.3.2.1 Design

For the purpose of this case study, we present in this Section the case study planning that includes the elements (the UML models, the UML consistency rules and how to implement them, and the UML tool) that are involved in running the case study.

We initially present the research objective of this case study, which is to investigate to what extent the most relevant UML consistency rules that we found in the literature are satisfied in actual UML models. This will show how a specific concept, as covered by a specific OCL rule, is already (or not) implicitly considered by designers during a typical UML-based development. For instance, is rule # 74, which checks if the Liskov's substitution principle is satisfied in a class diagram, is considered when a class diagram is created.

In Figure 32, we briefly present the three main steps we followed to carry out the case study reported in this Ph.D. thesis: 1) Selecting and encoding of UML consistency rules in OCL (Section 4.3.2.1.1), 2) Collecting UML model projects (Section 4.3.2.1.2), and 3) Checking the OCL rules on the UML model projects (Section 4.3.2.1.3).

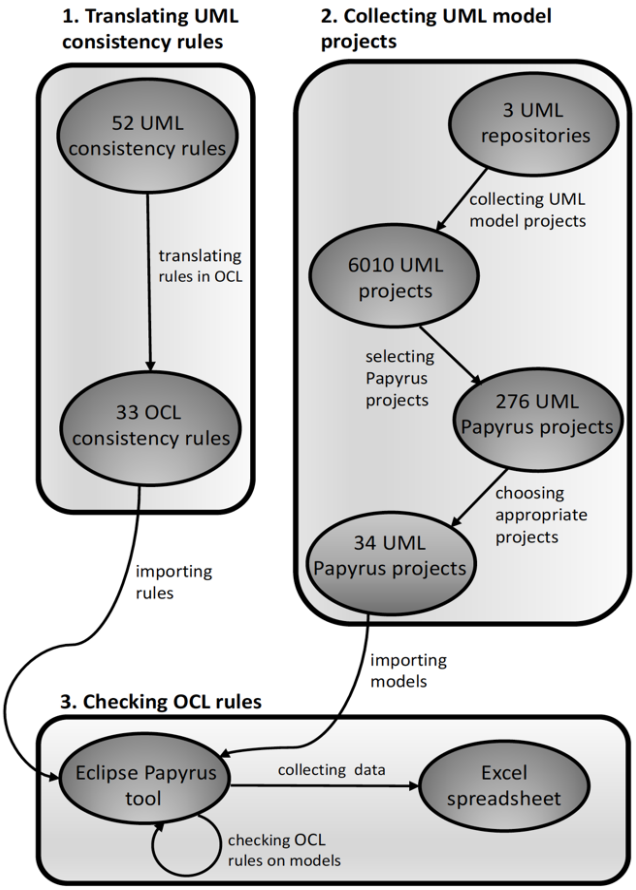

Figure 32. Case study plan (see accompanying text for a detailed description of the semantics of the figure).

4.3.2.1.1 Encoding the UML consistency rules in OCL.

The consolidated set of 116 UML consistency rules we discovered in Sections 3.1 and 3.2 was evaluated by MDSE experts from academia and the industry (Sections 4.1 and 4.2). The majority of respondents considered that 52 out of the 116 rules, i.e. 44.8%, should always be enforced in all UML models. We therefore tried to encode all 52 rules in OCL for automated evaluation. We were able to encode only 33 (see yellow ID# rules of Table 18 in Appendix A) out of those 52 rules in OCL (see project in GitHub[11] to access OCL rules) for different reasons: a) OCL is not expressive enough to encode all those rules (e.g., need a lot of processing); b) in some cases the UML standard does not include the necessary information to check the rules (e.g., requires specific information in use case descriptions). Some of the rules we excluded were: a) a state machine should be deadlock-free; b) for a send action, there should be a reception within the classifier of

[11] https://github.com/yvanlabiche/UML-model-consistency

the receiver instance that corresponds to the signal of the send action describing the expected behavior response to the signal; c) entity classes correspond to data manipulated by the system as described in a use case description; d) the name of a use case must include a verb and a noun (for instance "Validate User"). Moreover, we had to

The OCL specification of ten out of the 33 OCL rules was obtained from other authors who presented OCL rules, in particular, rules numbers 112, 55, and 110 [81], rule numbers 101, 105, 102, and 106 [90], and rule numbers 84, 74, and 75 [79].

Below, we present one example of the 33 rules (see GitHub for the full list of rules) encoded in OCL. Rule 84 specifies that a class cannot be a part of more than one composition (no composite part may be shared by two composite classes), which was initially presented by others [91] and be implemented in Eclipse Papyrus:

context UML::Property

inv Rule84:

(self.association <> null)
and (self.aggregation = AggregationKind::composite)

implies

(self.upper>=0 and self.upper <=1)

This step of selecting and encoding rules finally provided 33 OCL rules to check consistency among different UML diagrams. The OCL rules were saved in a library called "*UML consistency rules.ocl*", which was included in the Eclipse Papyrus directory "*Rules Validation*" that we dedicated to this work (see Figure 33).

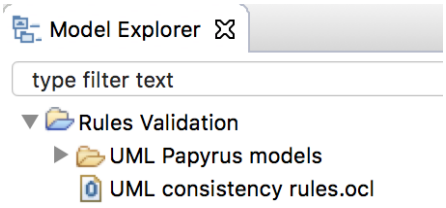

Figure 33. UML consistency rules file

Some of the 33 rules (see yellow ID# rules of Table 18 in Appendix A) were merged in a unique OCL rule when they were encoded in OCL. For instance, rule 54 (objects involved in a communication diagram should be instances of classes of the class diagram) and rule 115 (each class in the class diagram must be instantiated in a sequence diagram), involve the Communication Diagram (COMD) and Class Diagram (CD), and the Class Diagram (CD) and the Sequence Diagram (SD) respectively. In Eclipse Papyrus, those two rules both focus on *UML::Class* (for CD) and *UML::InteractionFragment* (for COMD and SD). Producing two separate OCL rules would have led to two identical rules. For this reason, the two rules were instead merged into one OCL rule.

In addition, for some of the 33 rules (see yellow ID# rules of Table 18 in Appendix A) to be encoded in OCL, we had to abstract (i.e. simplify) their original meaning provided in plain English when we encoded them in OCL because of OCL as a constraint language. Although new operations and even completely new kinds of types can be added to OCL, the mechanisms have their limitations. One cannot change the underlying fundamentals of the language. It is not possible to make OCL expressions have side effect, or to dictate an algorithm to apply when a certain expression is evaluated [201]. A simpler example of the abstraction/simplification procedure we applied is for rule 115 that specifies that each class in the class diagram must be instantiated in a sequence diagram. To be able to encode rule 115 we simplified it (i.e. we do not take care of the qualified names) in OCL as follows:

```
context uml::InteractionFragment
inv Rule115: (
        self->exists(self.name = UML::Class.name)
                )
```

Note that rule 115 is meant to hold for complete models; on the other hand, the models we collected in this case study may not be complete. We believe that the rule also assumes one is checking a specific kind of models. Specifically, it is not realistic for a design model, that shows in the class diagram how design patterns are used, to indicate how classes that are specific to design patterns interact in a sequence diagram: a design model is not meant to satisfy the rule. On the other hand, we believe it makes more sense for an analysis model to satisfy the rule.

4.3.2.1.2 Collecting UML models.

Due to the current lack of shareable, real (i.e. industry) UML models, which impairs the ability to perform extensive empirical studies [92], we initially considered three different, open-source UML repositories to collect UML models: the UML Repository[12] (R1), the Repository for Model-Driven Development (ReMoDD)[13] (R2) and the Lindholmen Dataset[14] [93] (R3). These three repositories are resources that aim to support the work of researchers and educators in the Model Driven Development community. Researchers and practitioners can use the repositories as a vehicle for sharing exemplar UML models.

The UML projects were identified in three steps (see Table 13):

S1) We downloaded all the UML model projects from the three repositories' websites, i.e. R1, R2 and R3. The following number of UML model projects were downloaded from each repository: 32 from the UML Repository, four from ReMoDD, and 5974 from the Lindholmen Dataset, for a total of 6010 UML model projects (see Figure 32).

S2) Since the UML model projects available on these three repositories were developed with different UML tools, meaning that they present different extensions (i.e. "*uml*", "*mdl*", "*xmi*", "*di*", "*notation*", etc.) we decided to select only UML projects that were developed by using Eclipse Papyrus. We chose to select only Papyrus models because this has been a way to trim the list of UML models and obtain a decent and manageable number of models. Nevertheless, since we only rely on Eclipse technology for defining and evaluating the rules (i.e. OCL constraints for Eclipse UML metamodel instances), this case study can be replicated on non-Papyrus models as well. This further selection returned a set of 276 UML Papyrus models, which were found on GiHub through the Lindholmen Dataset. This phase conducted manually, i.e. by checking the extension of each project. To include a UML project, we searched for the ones that contained the following Papyrus extension files: "*.di*" and "*.notation*" because each UML Papyrus model is composed by the model.*uml* that is the actual model and the model.*di* and model.*notation* files that contain the information used by the editor to depict diagrams.

[12] http://models-db.com/

[13] http://www.remodd.org/

[14] http://oss.models-db.com

S3) Out of the 276 Papyrus projects, 242 were not considered appropriate and were excluded for the following reasons:

- 193 were considered too simple. With the word "simple" we mean that the model didn't present any diagram that involved at least a total of ten of the following elements: classes, interactions, actions and decisions, states, and use cases;
- Ten were duplicates of the same project;
- 15 were made of corrupt files (i.e. Eclipse Papyrus was not able to open them, probably because they were created by older versions of Papyrus);
- 14 were UML profiles, which are not the type of UML models targeted in this work;
- Seven were empty models;
- Three models presented diagrams not covered by our 33 consistency rules (i.e. Component, Package, Deployment, diagrams).

Finally, the step S3) (Figure 32) returned 34 UML Papyrus models that focus on different domains in software development (for instance, aerospace, logistics, gaming, electronics, service, robotics). The 34 selected projects were eventually imported into the Eclipse Papyrus tool in the folder "UML Papyrus models" as shown in Figure 33. The characteristics of the people (i.e. if they were student, professional model developer, etc.) that created the 34 UML Papyrus models were not always reported in the UML projects, and for this reason were not collected.

Table 13: Summary of UML models collection

#	R1	R2	R3	Total
S1	32	4	5974	6010
S2	0	0	276	276
S3	0	0	34	34

4.3.2.1.3 Checking OCL rules

The UML tool that we used is Papyrus 2.0.X (developed on Eclipse Neon[15]), which was the recommended Eclipse version to use with Papyrus until this work was completed (April 2018)[16]. Moreover, the installation of the following additional plug-in was required: OCL Classic 2 SDK, Ecore/UML Parsers, Evaluator, and Edit (version 5.2).

This tool stack (Eclipse Neon + Papyrus + Plug-in) was chosen because it offers a specific validation view which allows to check the results of OCL rules. Users can also generate a plug-in from a profile that embeds the OCL rules created.

Having said that, the following three steps summarize and describe how we checked the 34 UML model projects with our 33 OCL rules:

1) we open the "*.uml*" file of each UML Papyrus models (see Figure 34-A);
2) then from the root node of the "*.uml*" model, we selected in the context menu OCL->Load Document (see Figure 34-B), and selected our "*UML consistency rules.ocl*" library (see Figure 34-C). This step added the OCL rules defined in our library to the set of UML well-formedness rules already implemented in Eclipse (based on the UML specification).
3) finally, to execute the OCL rules, i.e. validate the loaded model, from the root node of the "*.uml*" model again, we selected Validate (see Figure 34-D). This last step ran all the well-formedness rules of the UML standard as well as our 33 OCL rules.

4.3.2.2 Data Collection

The data collected in an empirical study may be quantitative or qualitative. Quantitative data involves numbers and classes, while qualitative data involves words, descriptions, pictures, diagrams etc. Quantitative data is analyzed using statistics, while qualitative data is analyzed using categorization and sorting [88]. For this case study, we try to report by using a combination of qualitative and quantitative data to provide a better understanding of the studied phenomenon. This approach is called "mixed methods"[87]. According to Lethbridge et al [89] our data collection

[15] https://www.eclipse.org/neon/

[16] https://www.eclipse.org/papyrus/download.html

technique uses indirect methods where we collected raw data without actually interacting with the UML Papyrus models during the data collection.

We answered the main research objective (see introduction of Section 4.3.2) by extracting data from the execution of the OCL rules in each UML Papyrus model, and recording that data in an Excel spreadsheet (our data collection form) presented in Table 14.

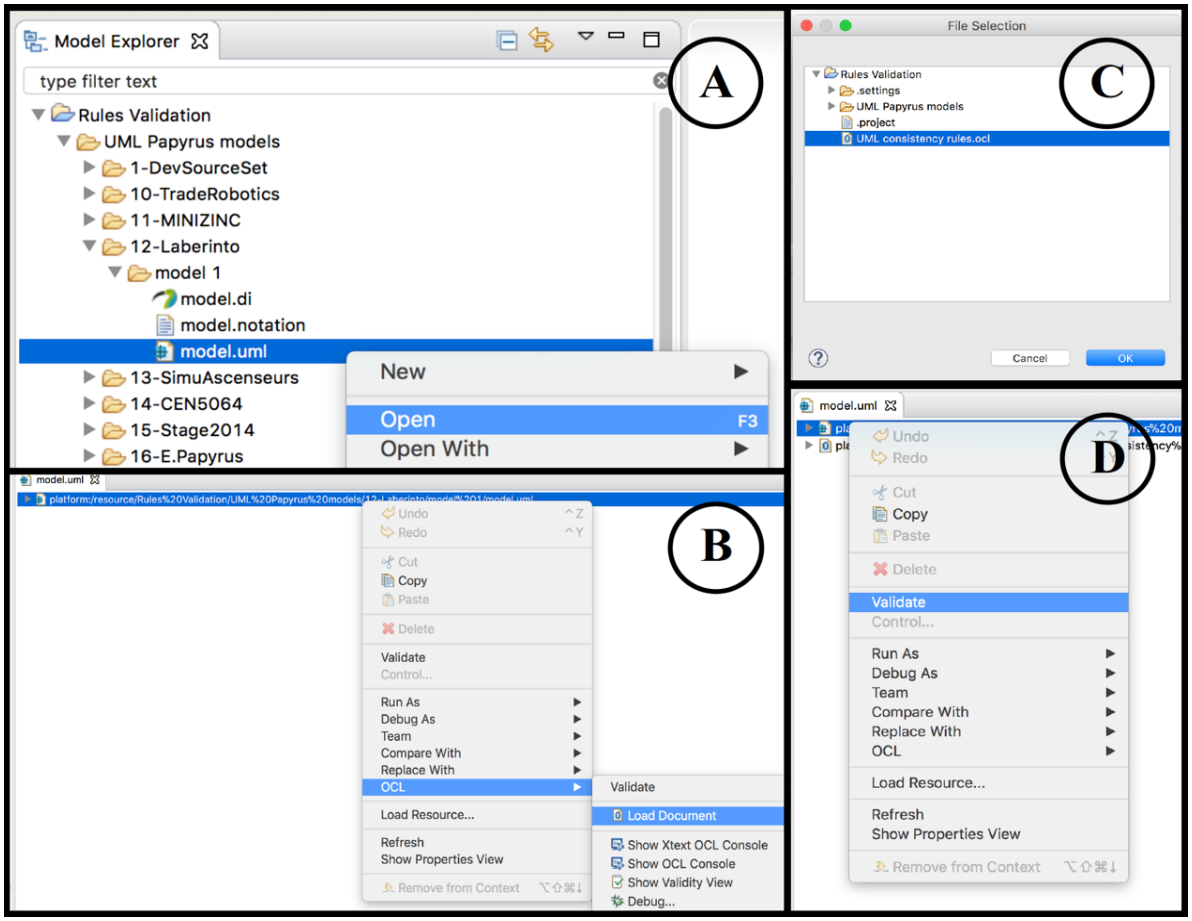

Figure 34. Eclipse Papyrus steps to check OCL rules (see accompanying text for a detailed description of the semantics of the figure).

We extracted the data from the case study models according to a number of criteria, which were directly derived from the main research goal presented at the beginning of Section 4.3.2.

Table 14: Sample of the data collection form

Data Collection Form – Case Study						
#	Name	Link	Diagram	UDI	Elements	33 rules

By using each data extraction criterion required, we evaluate the 33 OCL rules on each of the 34 UML models in Eclipse Papyrus. The following information was collected from each UML model (Table 14):

- A univocal code, Name, online link, and number of diagrams for each UML Papyrus models (see respectively first to fourth columns in Table 14).
- The type of the UML diagrams involved (UDI) in each UML model project: Class (CD), Use Case (UCD), Communication (COMD), Composite Structure (CSD), State Machine (SMD), Sequence (SD), Object (OD), Activity (AD), Diagram (see fifth column in Table 14);
- The elements involved in the UML diagrams: classes (CD, CSD and OD), interactions (SD and COMD), actions and decisions (AD), states (SMD), use cases (UCD) (see sixth column, "Elements" in Table 14);
- The number of times each of the 33 rules identified inconsistency in each UML model project (see seventh column in Table 14;

4.3.2.3 Execution and results

To reach the goal of this case study, i.e., to what extent the most relevant UML consistency rules we found in the literature are satisfied by actual UML models, the 33 OCL rules were executed on the 34 UML Papyrus models. We first noted that many well-formedness rules specified in the standard are violated by the UML models. What we focus and report in this Ph.D. thesis are additional violations that are specific to the 33 OCL rules.

Before we report on the results of the OCL rules execution it is important to mention that the 33 OCL rules focus on different UML diagrams in the following way: CD (19 rules), UCD (two rules), COMD (three rules), CSD (four rules), SMD (four rules), SD (11 rules), OD (one rule), ID (one rule), and AD (three rules). In addition, 18 OCL rules involve only one UML diagram, and the other 15 ones two different UML diagrams. No rule involves more than two diagrams. Having said that, after the execution of the OCL rules on the UML Papyrus models, the following results were identified:

- Only 30 out of 33 OCL rules could have actually found inconsistencies in the UML Papyrus models, because three rules specifically focus on the communication diagram (rules 46, 54 and 55 in Appendix A) and the set of UML Papyrus models did not include any communication diagrams. Another way to discuss this is that the 34 models are consistent with respect to these three rules, but this is not due to some correct use of the communication diagram but rather to the absence of those diagrams;
- Three of the 34 UML Papyrus models do not show any inconsistency. Similar to the previous case, this is not due to some correctness of the models. Indeed, those models only involve ADs (two models) and UCDs (one model), which are not covered by any of our 33 OCL rules: we do not have rules that only involve (or check the correctness of) either one or the other of these two diagrams or that involve both diagrams. More specifically, we do have four rules that involve AD and UCD (i.e. rules 39, 40, 50, and 98—see Appendix A), but none of them focus only on UCD or AD, or check that the two diagrams are consistent with one another. They rather relate some elements of AD or UCD with other diagrams;
- 31 (i.e. 34 minus the three abovementioned) UML Papyrus models involved a total of 206 UML diagrams: Sequence (27), Class (82), State Machine (12), Use Case (12), Activity (54), and Object (19) diagrams. The 206 UML diagrams present a total of 1722 "elements" including 898 classes (CD, OD, and CSD), 369 actions and decisions (AD), 341 interactions (SD), 57 use cases (UCD), and 59 states (SMD);
- 12 of the 31 UML Papyrus models did not trigger any inconsistency according to the 30 OCL rules. Those 12 models presented 20 out of 206 UML diagrams which are not involved in any inconsistency identified by our 30 OCL rules: 4 (SD), 10 (CD), and 6 (OD).
- The 30 OCL rules found 2731 inconsistencies among 206 UML diagrams.

A quantitative summary of the results according to our main research goal and the data collection form (Section 4.3.2) is presented Table 15. More details about the results are provided in the next Section.

Table 15 presents the raw results obtained after the execution of the 30 OCL rules on the 206 UML diagrams. We omitted the three rules, discussed earlier, that cannot identify inconsistencies, as well as the three abovementioned models that do not have diagrams involved in our rules. In the

analysis of results, we therefore refer to 30 rules and 31 models. Table 15(a) presents the number of inconsistencies we found with each of the 30 OCL rules: the consistency rule number appears in bold face (see yellow ID# rules of Table 18 in Appendix A) with the number of inconsistencies found for this rule appearing immediately below; data appear in four rows. For example, we found a total of 58 inconsistencies across the 31 UML projects for rule 82. Table 15(b) describes the number of times each UML diagram type was found involved in at least one inconsistency, i.e. according to any of the 30 OCL rules. For example, the OCL rules found 1670 inconsistencies that involved class diagrams (CD). The sum of the inconsistencies' frequencies for each UML diagram is 5282, which is greater than 2731 (the total number of inconsistencies identified), because the rules that involved two UML diagrams were counted twice. For instance, rule 98, which describes some consistency between the use case diagram (UCD) and the sequence diagram (CD), identified a total of 596 inconsistencies and we therefore counted 596 inconsistencies for both UCD and SD in the Table. Table 15(c) presents the number of inconsistencies we found in each of the 31 UML Papyrus models. In particular, the first raw of each Section presents the code (in bold face) assigned to each UML Papyrus model (see project in GitHub[17]), and the second raw presents the number of inconsistencies found in each model: for example, the UML Papyrus model 29.1 shows 21 inconsistencies. The fact that some models are identified with a number of the form x.x means that the same UML project had more than one model version.

As discussed earlier, 15 of the 33 rules check the consistency between two different diagrams of the same UML model (e.g. consistency between a class diagram and a sequence diagram). A reader familiar with UML, OCL and Papyrus will notice that, given our OCL implementation of those 15 rules (please refer to our Github project), if the UML model being checked does not include one of the two diagrams involved in any of those rules (e.g. a rule checks the consistency between a class diagram and a state machine diagram but the model being checked does not include the latter), then the model verification will identify an inconsistency.

[17] https://github.com/yvanlabiche/UML-model-consistency

Table 15: Raw results after the execution of the case studies. Rules, Diagrams and Model IDs are in bold; number of inconsistencies are below these.

***(a)** 30 UML Consistency rules*								
Rule ID	**9**	**13**	**24**	**27**	**28**	**39**	**40**	**48**
Inconsistencies	0	62	0	0	63	0	298	103
Rule ID	**50**	**74**	**75**	**76**	**77**	**78**	**80**	**81**
Inconsistencies	62	0	0	58	0	0	0	0
Rule ID	**82**	**84**	**85**	**98**	**101**	**102**	**105**	**106**
Inconsistencies	58	0	0	596	1	0	1	0
Rule ID	**108**	**109**	**110**	**112**	**115**	**116**		
Inconsistencies	0	182	139	55	899	154		
***(b)** UML diagrams involved*								
Diagram ID	**CD**	**OD**	**SD**	**SMD**	**UCD**	**AD**	**ID**	**CSD**
Inconsistencies	1670	0	1830	165	658	361	596	2
***(c)** 31 UML Papyrus models*								
Model ID	**1.1**	**1.1**	**2**	**6**	**8.2**	**9.2**	**9.3**	**9.4**
Inconsistencies	46	4	75	12	0	0	2	0
Model ID	**9.5**	**9.6**	**9.7**	**12**	**13.1**	**13.2**	**14**	**18**
Inconsistencies	0	0	2	30	68	44	1924	10
Model ID	**22**	**26.1**	**26.2**	**28.1**	**28.2**	**29.1**	**29.2**	**30**
Inconsistencies	2	4	5	0	0	21	24	156
Model ID	**31**	**36**	**37**	**38**	**39**	**40**	**42**	
Inconsistencies	36	264	0	0	0	0	0	

This is simply a side effect of our procedure to check rules. A different, more effective procedure would check any of those rules only if the model being checked contains the two diagrams involved in the rule. In the results presented below, we do not count inconsistencies that are simply due to the absence of a diagram in a model.

4.3.2.4 Analysis of the results

In this Section, we analyze the data obtained previously, by following a qualitative quasi-statistical approach [22], which includes calculation of frequencies of the identified inconsistencies according to different UML Papyrus models and diagrams.

Table 15(a) shows that 15 out of 30 OCL rules did not find any inconsistency in the 31 UML Papyrus models. 12 of these 15 OCL rules involved only one UML diagram (eight CD, two CSD, one SD, and one SMD).

Recall that in our set of 33 rules, 18 rules involve only one diagram whereas 15 involve two diagrams. 12 of those 18 OCL rules (66.7%) did not find any inconsistency whereas 13 of those 15 OCL rules (86.7%) found inconsistencies. This suggests that designers were comfortable working on individual diagrams but had difficulties designing several diagrams at the same time in a consistent way.

Seven of the 15 OCL rules that found inconsistencies, specifically rules 40, 48, 98, 109, 110, 115, and 116, each found more than 100 inconsistencies. The reader will notice (see yellow ID# rules of Table 18 in Appendix A) that all these rules involve two diagrams and one of those two diagrams is always the sequence diagram. 13 models (out of 31) have the right diagram types so we can evaluate whether they contain inconsistencies that are specified by these seven rules; 11 of these 13 models contained such inconsistencies. This suggests that a) the majority of the designers of the models we selected may not fully understand how the sequence diagram ought to be consistent with other diagrams, b) some rules involving SDs with another diagram may be intended to hold only on complete UML models while the models we collected may not be complete (this may have increased inconsistencies related to SDs).

The OCL rules number 98 and 115 detected the highest number of inconsistencies across the UML Papyrus models, respectively 596 and 899 inconsistencies. Most of these inconsistencies were found in UML Papyrus model number 14 (see Table 15(b)) that contains the largest number of diagrams (9 CDs, 12 SDs, and 3 UCDs), which are the diagrams that those two rules focus on. Seven of the 31 models have the right diagram types, so we can evaluate whether they contain inconsistencies that are specified by these two rules; Each one of these seven models contained such inconsistencies.

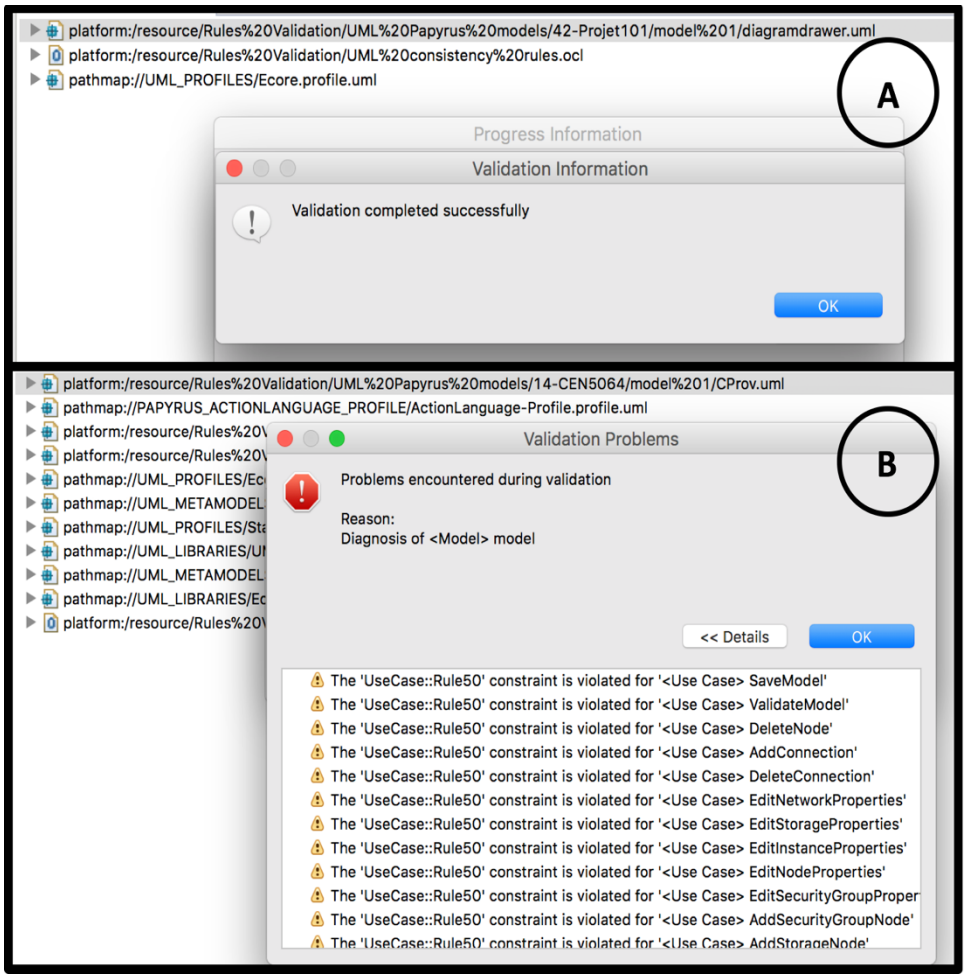

Figure 35. Triggering OCL rules

Table 15(b) shows that CD and SD were by far the UML diagrams having the highest number of inconsistencies. This is not surprising since these are likely the most frequently used UML diagram kinds [44]. In addition, the majority of our OCL rules are related to CDs and SDs, respectively with 19 (57.6%) and 11 (33.3%) rules out of 33 OCL rules, and the 31 UML Papyrus models contained 82 CDs (with 898 classes) and 27 SDs (with 341 interactions) out of a total of 206 UML diagrams. We looked for models that provide opportunities for detecting inconsistencies by computing the cumulative number of models that have the right diagram types so that the OCL rules that involve the class diagram (either only the class diagram or the class diagram and another diagram type, which therefore needs to be in the model to have the right diagram types) can have a chance to discover an inconsistency; as well as the cumulative number of models that have the right diagram types so that the OCL rules that involve the class diagram have a chance to discover an inconsistency and that actually contain an inconsistency as specified by those rules. We did the

same for the sequence diagram. The ratio, as a percentage, between model opportunities and actual inconsistencies for the two types of diagrams are: 15.5% for the class diagram and 58.5% for the sequence diagram; showing that the prevalence of inconsistencies involving SDs is much higher than that for CDs.

It is important to mention that the 54 ADs (with a total of 369 actions and decisions), which is the second UML diagram most involved in the 31 UML Papyrus models after the CD, triggered only 361 inconsistencies. The three rules that involved ADs were triggered seven out of ten times (70%).

Table 15(c) indicates that 12 UML Papyrus models are consistent according to our 30 OCL rules. For instance, UML Papyrus model 42 (Table 15(c) and Figure 35-A) passed consistency checks. Those 12 models include 20 UML diagrams, four SDs, ten CDs, and six ODs. This result shows that the 12 models are much simpler than the remaining 19, and moreover, it suggests that checking consistency is especially important when diagram size and complexity increases.

On the other hand, UML Papyrus model 14 (Table 15(c) and Figure 35-B), which is the biggest model of the set of models we considered, with a total of 48 UML diagrams (i.e. 12 SDs, nine CDs, six SMDs, three UCDs, six ADs, and 12 ODs), triggered 1,924 different inconsistencies. With this model, the entire set of 30 OCL rules is valid (i.e. the model has all the diagram types involved in the rules), 13 of which (43.3%) revealed at least one inconsistency.

In Table 16 we present the summary of the results obtained by execution of the 30 OCL rules on the 31 UML Papyrus models based on the different types of UML diagrams. The first column “Type of Diagrams” refers to types of diagrams involved in the UML Papyrus models. Because CSD and CD use the same constructs (i.e. Structured Classifiers) in the UML standard [57], data for CSD and CD are merged and presented only as CD; similarly, data for ID and SD are only presented as SD. Column “Diagrams” provides the total number of diagrams of a given type considered in this study. Column “Elements” indicates the total number of elements involved in each diagram, i.e. classes for CDs and ODs, actions and decisions for ADs, interactions for SDs, states for SMDs, and use cases for UCDs.

Table 16: Summary of the results according UML diagrams

	1	2	3	4	5	6
Type of Diagrams	**Diagrams**	**Elements**	**Rule Count**	**Opportunities**	**Inconsistencies**	**Total inconsistencies**
CD	82	755	23	247	53	1672
AD	54	369	3	10	7	361
SD	27	341	12	51	24	2426
OD	19	143	1	1	0	0
SMD	12	59	4	8	5	165
UCD	12	57	2	4	4	658

Column "Rule Count" shows the total number of OCL rules that involve each diagram type: for instance, 12 rules involve SD, three rules involves ADs, etc.; of course, when a rule involves two diagrams, the rule counts one for each diagram type so the sum of the numbers in this column is greater than the total number of rules (30). Column "Opportunities" is a cumulative count of the models that have the required diagrams to check the OCL rules. For instance, rule 109, which involves CD and SD, can be checked on model number 30 since this model has three SDs and two CDs, which contributed +1 for both CD and SD to this column. Column "Inconsistencies" is the cumulative number of times a model appears to have at least one inconsistency as specified by a rule involving a specific diagram type (row). For instance, rule 115 that involves CD and SD found 91 inconsistencies in model number 30, and this contributes +1 to this column for both CD and SD. Column "Total inconsistencies" reports on the total number of inconsistencies discovered by the OCL rules and that involve each type of diagram.

Table 17 shows different ratios of column values from Table 16. The first column of Table 17 is the same as that of Table 16: Types of Diagrams (TD in Table 17). Column "R 6/1" is the ratio of the 6^{th} column of Table 16 over its 1^{st} column; this gives the average number of inconsistencies for each UML diagram type. For instance, each of the 82 (Table 16) class diagrams (CD) has an average of 20.4 inconsistencies.

Table 17: Analysis of the results according UML diagrams

TD	R 6/1	R 4/1	R 6/3	R 4/5	R 6/5
CD	20.4	3	72.7	21.5%	31.5
AD	6.7	0.2	120.3	70%	51.6
SD	89.9	1.9	202.2	47.1%	101.1
OD	0	0.1	0	0%	0
SMD	13.8	0.7	41.3	62.5%	33
UCD	54.8	0.3	329	100%	164.5

The highest number of inconsistencies (on average) is found in sequence diagrams (SD): 89.9 inconsistencies on average involving each sequence diagram. Column "R 4/1" is the ratio of the 4th column of Table 16 over its 1st column; this gives the average number of UML diagrams checked by each one of the 30 OCL rules. CDs were the UML diagrams most checked by our OCL rules. This is not a surprise because the CDs are the UML diagrams most involved in our set of OCL rules. Column "R 6/3" is the ratio of the 6th column of Table 16 over its 3rd column; this gives an average number of inconsistencies that were found by OCL rule in each UML diagram. For instance, each one of the 12 OCL rules that involved SD triggered an average of 202.2 inconsistencies. Column "R 4/5" is the ratio of the 4th column of Table 16 over its 5th column; this gives the percentage of times each OCL rule that was checked found an inconsistency. For example, in 21.5% of the times that the rules that involved CDs were checked, they identified at least one inconsistency, on the other hand, inconsistencies in SDs were identified 47.1% of the times an OCL rules involving this diagram was used. Column "R 6/5" is the ratio of the 6th column of Table 16 over its 5th column; this gives the average number of inconsistencies that were identified each time an OCL rule triggered an inconsistency. For instance, each one of the seven times and OCL rule was triggered to check ADs, an average of 51.6 inconsistencies were identified.

In sum, the analysis of the results presented in this Section shows that, despite the fact that the experts in the field considered that the OCL rules we used should be enforced in any UML model, we observed that inconsistencies are numerous in the UML models we used, especially the ones that involve SDs.

In addition, we realized that most of the inconsistencies identified by the OCL rules involve rules that focus two UML diagrams. This may indicate that the UML Papyrus designers may be more comfortable in developing and keep consistency within a single UML diagram and may find it hard to keep consistency between two different UML diagram types, especially when a sequence diagram is one of the two diagrams involved.

4.3.3 Conclusions of Section 4.3

This work presents the results of a case study that involved a set of 33 OCL rules that were used to check the consistency of diagrams in 34 UML Papyrus models. Starting with the set of 52 UML consistency rules identified in Section 4.2, the goal of this case study was to investigate to what extent existing UML models satisfy those rules. To do that, this work described an exploratory flexible indirect case study [22] to verify the UML consistency rules on the UML models. After the execution of the OCL rules on the models, we realized that only 30 OCL rules and 31 UML Papyrus models were finally used (see Section 4.3.2.1 for more details), and we identified a total of 2731 inconsistencies among the 206 UML diagrams that constitute the 31 UML models. A first conclusion we can draw is that, despite the fact that the OCL rules we used were validated by experts in the field who indicated the rules should always be enforced (Section 4.2,), i.e. any UML model ought to satisfy the rules, we observe that inconsistencies are plentiful in the open-source models we used. This may represent a concern to anyone who intends to use those models to experiment with some kind of UML-based, Model-Driven activity since such activities typically rely on consistent diagrams [11]; inconsistent diagrams will likely lead to invalid experimental results.

The main contributions of this Section are the description of the process to check OCL rules on UML models in Papyrus and the evaluation of 33 of those rules to detect different inconsistencies between diagrams of UML models. This work can thus be considered a contribution to defining the best practices of UML-based Model-Driven software engineering. Designers can use the information provided by this work to better understand UML inconsistencies. Other comments and observations regarding Section 4.3 will be described in the final Chapter 6.

5. THREATS TO VALIDITY

The main threats to the validity of this Ph.D. thesis are related to the empirical research studies that have been conducted, publication bias, selection bias, inaccuracy in data extraction, and misclassification [94].

In the following Sections, we analyze the potential threats to validity that could affect the steps that were carried out along this Ph.D. thesis, i.e. 1) in Section 5.1, 5.2, 5.3 and 5.4 we discuss the threats to validity to the experimental results discussed in Sections 3.1, 3.2, 4.2, and 4.3, respectively. In Sections 5.2, 5.3 and 5.4 we divided the threats to validity according to the four categories suggested by Wohlin et al. [95]. In Section 5.4 we adapted those previously cited four categories for case study in software engineering according to Runeson et al. [22].

5.1 Threat to Validity – A Systematic Mapping Study of UML consistency (Section 3.1)

We used seven search engines to collect journals, conferences and workshops proceedings that are relevant to UML consistency rules; we did not consider grey literature [31] (e.g., Ph.D. theses, books) or unpublished results (e.g., technical reports) because these might affect the validity of our results because they were not peer-reviewed. Selection bias refers to the distortion of a statistical analysis owing to the criteria used to select publications. As it is impossible to completely cover every publication written on our topic, we acknowledge that some relevant papers might not have been included. Finally, when considering the limited number of studies in this work, we know that the coverage of this work might be the most inherent threat to validity, even though we followed rigorous guidelines that are well-known in the empirical software engineering community [31].

We did not carry out any quality assessment in this SMS. Quality assessment is more essential in systematic literature reviews to determine the rigor and relevance of the primary studies. In a SMS no quality assessment needs to be performed [202]. We attempted to solve this problem by defining our inclusion and exclusion criteria in order to gather the most relevant papers regarding UML

consistency rules. To help ensure an unbiased selection process, we defined seven research questions in advance, organized the selection of articles and finally created and followed a multi-phase process to execute the SMS. We would also like to mention that during the data extraction process, there was the possibility of subjectivity when we decided what was (and what was not) related to our topic. This interpretation might have affected the results. The entire systematic procedure was performed by Damiano Torre beforehand and supervised and evaluated by the Prof. Yvan Labiche and Prof. Marcela Genero.

5.2 Threat to Validity – A systematic identification of UML consistency (Section 3.2)

Construct validity concerns the relation between the measures used in the study and the theories on which the research questions are grounded [96]. In the work presented in Section 3.2, this concerns the systematic identification of UML consistency rules, which is inherently dependent on the coherence of the terminology in the field. In order to mitigate these threats, we described the terminology and concepts used to measure the frequency of UML consistency rules regarding the different UML consistency issues. We collected the concepts and terminology that were used to carry out those measurements from our previous systematic study presented in Section 3.1, which focused on UML consistency. We observed that the terminology used by researchers is not always consensual and that authors in this field sometimes discuss UML consistency issues using different terms such as “Horizontal Consistency” and "Intra-model Consistency" to refer to the same concept. We used these synonyms when searching for the primary studies. Among other issues, researchers may use a different terminology for a particular topic [97], such as rules concerning diagram (model) synthesis (see Appendix C), rules in standard UML-driven software development processes discussed in textbooks, traceability rules, and rules for domain specific languages; these types of rules are not covered in this study. Another threat to construct validity is represented by the fact that UML can be seen as a mechanism with which to define the notation and semantics for several kinds of software development phases and the exact software development phase under which a particular UML diagram falls may, therefore, be a matter of discussion [98]. Finally, by following a systematic research method, we developed a set of terms

with the intention of providing a useful and accessible set of definitions for the terms that are used within the UML consistency domain.

Threats to *internal validity* are influences that can affect the independent variable with respect to causality, without the researcher's knowledge [95]. The main idea behind conducting the study in Section 3.2 was to collect UML consistency rules in literature without, as far as possible, introducing any researcher bias since otherwise internal validity would appear to be a major challenge for the study. Since in this study we relied solely on rules that have been published in literature and identified in Section 3.1, the fact that we did not find rules involving certain diagram(s) does not mean that none have been created. Moreover, we acknowledge that our set of rules has been collected from (primarily) academic literature, and that other rules from the industry were not included. This is indeed a threat to the validity of our work, especially in the attempts of generalizing our results. With this being said, one could argue that, although the rules have been collected from academic literature, they have been described in support of some software engineering tasks that would theoretically be of use by practitioners. In addition, our set of rules has been validated by some experts from industry. So, we think that there is no reason to believe that rules for standard UML development would be drastically different in industry. Moreover, the idea of industry experts contributing with other rules may increase the risk that those rules would be project or domain specific, or that such rules would likely be representative of their development process and considered a critical asset that they would not be willing to share. Finally, we know that there is a lot of tool support for UML, such as IBM Rational products[18], and since these tools support some kinds of Model-Driven development activities, they must rely on, and likely enforce, UML diagram consistency. However, the rules they rely on are not necessarily in the public domain, or at least are not discussed in published literature. In addition, some rules might also be for specific domains (industry, organization, project, team specific, etc.) Moreover, it is important to say that some of our 116 UML consistency rules a) are difficult, and potentially impossible to enforce (e.g. specifying that if an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, the different diagrams should specify the same scenarios, i.e. rule 39), b) cannot be captured in OCL (e.g. specifying that a state machine should be deadlock-free, i.e. rule 12), c) are relatively simplistic (e.g. specifying that an instance is required in a

[18] https://www-01.ibm.com/software/rational/uml/products/

sequence diagram for each class, i.e. rule 115), d) do not apply to partial/incomplete models and may be specific to a particular modeling approach (e.g. focusing on a one-to-one mapping between use cases and sequence diagrams, i.e. rule 98), e) should not always be enforced (e.g. saying that the type of a lifeline, i.e. the type of the connectable element of the lifeline, in a sequence diagram must not be an interface nor an abstract class, i.e. rule 108), f) are only slightly different but not equivalent to each other (e.g. rule 76: an abstract operation can only belong to an abstract class, and rule 82: a class that contains an abstract operation must be abstract).

External validity refers to generalization from this study. In general, the external validity of a systematic study is strong, as the key idea is to aggregate as much of the available literature as possible [96]. Moreover, our specific research goal focused on UML consistency rules (see the inclusion/exclusion criteria in Section 3.2.1.2), and the fact that we do not claim that our map covers other types of UML rules such as traceability rules, synthesis rules (Appendix C), rules presented in books [25] and domain specific rules [54], helps to reduce the threats to external validity. We used seven search engines to search journals, conference and workshop proceedings that are relevant to UML consistency rules (Section 3.1). The limited number of rules in this research could be an inherent external threat to validity, despite the fact that we attempted to mitigate this by following rigorous guidelines that are well-known in the empirical software engineering community [31-33].

The *reliability* is concerned with the extent to which the data and the analysis are dependent on the specific researchers [95]. In order to achieve reliability, the research steps must be repeatable, i.e., other researchers have to be able to replicate our results [99]. While conducting this type of study, there is a possibility of missing relevant papers and consequently UML consistency rules. It is impossible to thoroughly search for every rule published on UML consistency, and we acknowledge that some rules might not have been included. The main issues that may constitute a threat to the reliability of our research are related to publication bias, selection bias, inaccuracy in data extraction, and misclassification [94]. Selection bias refers to the distortion of a statistical analysis owing to the criteria used to select the rules. We attempted to mitigate this risk by carefully defining (1) our inclusion and exclusion criteria in order to select primary studies (see Section 3.1.1.3), and (2) our exclusion criteria in order to select rules in Section 3.2.1.2, based on our predefined research questions. In order to increase reliability, two researchers (Damiano Torre and Prof. Yvan Labiche) were responsible for applying these criteria, with the help of Dr. Maged

Elaasar who was contacted in the case of a disagreement regarding the inclusion or exclusion of a rule.

5.3 Threat to Validity – Online survey: how consistency is handled in MDSE and UML (Section 4.2)

Construct validity refers to the extent to which the measurement instruments used in the study actually reflect the construct under study [100]. To mitigate these threats, we described the terminology and concepts used to measure the frequency, relevance, familiarity, and understandability of respondents regarding the different model consistency issues, and UML consistency rules. We collected concepts and terminology that were used to carry out those measurements from our previous systematic studies described in Sections 3.1 and 3.2, which focused on UML consistency rules. Respondents may have misunderstood our online questions. To reduce ambiguity, we reviewed the survey with colleagues, and tried to make the questions as simple and straightforward as possible. Our survey was a balanced mix of closed (without an option for participant comments) and open questions. Another possible bias to consider is that people tend to drive their answers when they feel they are being evaluated according to what they think is the goal of a study. In order to mitigate this bias, at the beginning of the survey, we notified participants that the complete questionnaire was anonymous. We acknowledge that the fact that the survey asked respondents whether they understood the rules (Q20 in Section 4.2.2.3.3) represents a threat to the validity: an alternative and better version of this question would have asked comprehension questions about the rules to effectively evaluate respondents understanding.

Threats to *internal validity* are influences that can affect the independent variable with respect to causality, without the researcher's knowledge [95]. In terms of internal validity, we did not carry out probabilistic sampling for the selection of respondents. Our recruitment strategy could have incurred a possible selection bias (for example, a high probability of profile similarity among the respondents, such are participants which are working in the context of MDSE). It would, of course, be interesting to conduct a related survey with those who are not working in this area. We attempted to address this issue when we defined the protocol of the survey: we explicitly required the survey to be filled in by respondents from academia and industry who are working in the context of MDSE. Also, we especially wanted expert opinion about UML consistency rules to identify which of our set of 116 UML consistency rules identified in Sections 3.1 and 3.2 should

always or often be enforced so as to implement them in OCL (Section 4.3) to be used by an average UML user. We believe that asking average UML users the same questions would have introduced a much high threat to validity.

External validity threats concern the generalization of the results [95]. In our case, the results are limited to those who have experience in the area of MDSE. However once again, it was important we obtained expert opinion rather than the opinion of the average UML users. Although the demographics of our sample are fairly diverse, we are aware that the size of our sample is not large enough (especially in the industry) for generalization purposes. We make no claims that the results generalize to model based software development more broadly.

Conclusion validity threats regard issues affecting the ability to draw accurate study conclusions [95]. Most of our analysis was based on simple descriptive statistics, we did not carry out a quantitative statistical analysis of our data. This decision was made due to the limited sample size of our study: in this situation, a quantitative analysis may overstate some of the relationships among our variables, leading to possible misinterpretation of the results [101]. The entire survey was run by Damiano Torre and supervised and evaluated by the Prof. Marcela Genero, Prof. Yvan Labiche and Dr. Maged Elasaar.

5.4 Threat to Validity – UML consistency rules: a case study with Eclipse Papyrus models (Section 4.3)

Construct validity reflects to what extent the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions [22]. In the study presented in Section 4.3, this is related to the ability to encode the 33 consistency rules presented in Appendix A (see yellow ID# rules of Table 18) as OCL rules according to the Eclipse MDT UML2 5.0[19], which is compliant with the UML 2.5 specification. The encoding of the UML consistency rules in OCL was performed by Damiano Torre and tested on various simple UML diagrams to evaluate that they check the right properties of the UML diagrams i.e., classes, attributes, connectors, relationships, names, etc. Another threat to construct validity is whether the open source UML Papyrus models we used represent a set of models as representative as possible.

[19] https://projects.eclipse.org/projects/modeling.mdt.uml2/releases/5.0.0

We do not claim that the selected UML Papyrus models constitute a complete set or the best set of UML models possible (if such a set actually exists). Finding realistic UML models is no easy task and this has driven several initiatives, which we relied on, to collect models. We did not cherry-pick model and we described a systematic procedure (see Section 4.3.2) that led us to use a number of models that we believe is not trivial.

Internal validity is of concern when causal relations are examined [22]. The most important internal validity threat to our study is related to the collection process of inconsistencies that was manually performed. In order to minimize this threat, the inconsistencies collection for each UML Papyrus model was performed twice. In addition, the sampling of UML models only considered Papyrus models. Our collection strategy could have incurred a possible selection bias (for example, a high probability of similar UML models). It would, of course, be interesting to conduct the same case study with models from the industry and/or that not necessarily use Papyrus. We attempted to address this issue when we defined the protocol to collect UML models in Section 4.3.2.1.2. Finally, we acknowledge that the purpose and the level of completeness (in a development process) of the UML models we used in this case study could have an impact on the results. Specifically, some of the rules we used, such as the one requiring that each use case diagram be mapped to a collaboration diagram, seem to only apply on complete models; If instead some of the models we evaluated are not meant to be complete, then checking this rule will fail but this will be a false negative.

External validity is concerned with the extent to which it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [22]. The first threat to external validity is whether the conclusion can be generalized to other UML models. We admit that different UML models may incur different inconsistencies. So, the results derived from this case study may not be applicable to others. We nevertheless believe that, given the extent of the models we used, given the extent of the inconsistencies we revealed, the results of this study are still meaningful and valuable. The second threat is that the analysis we performed on the identified inconsistencies is only limited to the models discussed and does not represent the inconsistencies that may be triggered in an average UML model. In practice, it is very difficult (if not impossible) to find open source UML models that may be considered complete, complex, or representative enough to be able to generalize the results obtained. Only replications of our study

will help. The fact that we provide the analyzed models and the OCL consistency rules on a GitHub will facilitate replications.

Reliability validity is concerned with the extent to which the data and the analysis are dependent on the specific researchers [22]. To minimize this threat, we decided to collect the highest possible number of open source UML Papyrus models (involving at least a specific number of UML elements, see Section 4.3.2.1.2 for more details), which were developed in different software domains (i.e. aerospace, logistics, gaming, electronics, service, robotics, etc.).

6. CONCLUSION AND FUTURE WORK

This Ph.D. thesis is supervised by Yvan Labiche (Associate professor, Systems and Computer Engineering, Carleton University), Marcela Genero (Full Professor at the University of Castilla-La Mancha in Spain) and Maged Elaasar (Adjunct professor, Systems and Computer Engineering, Carleton University) under the umbrella of a Cotutelle agreement between Carleton University (Ottawa, Canada) and University of Castilla-La Mancha (Ciudad Real, Spain). This work has been developed within the SEQUOIA Project, (TIN2015-63502-C3-1-R) (MINECO/FEDER) funded by "Fondo Europeo de Desarrollo Regional and Ministerio de Economía y Competitividad" (Spain), the GEMA project funded by "Consejería de Educación, Cultura y Deportes de la Dirección General de Universidades, Investigación e Innovación de la JCCM (Spain) and a NSERC Discovery Grant (Canada). The main goal of this Ph.D. thesis is to carry out the systematic identification and the validation of a set, as complete as possible, of well-accepted consistency rules for UML diagrams. To achieve this objective, we presented two main research goals in Section 1.3, respectively to collect and validate the UML consistency rules. To answer these two main research goals, the following work has been done:

- **G1** (What are the UML consistency rules proposed in the literature?) – Chapter 3.
    - Systematic Mapping Study (SMS) to collect the current state of the art in terms of UML consistency rules (Section 3.1). This SMS and accompanying results have been published and presented at the 18th ACM International Conference on Evaluation and Assessment in Software Engineering (EASE) in 2014, a premier conference in the field [11];
    - A systematic identification of UML consistency rules (Section 3.2). This work presented the results obtained after following a systematic protocol, whose aim was to identify, present and analyze a consolidated set of 116 UML Consistency rules obtained from the work presented in Section 3.2. The updated version of the work presented in Section 3.2 was accepted for publication at the International Journal of Systems and Software [17]. The updated version [17], with a search for primary studies up to 2018, revealed ten more papers (i.e. a total of 105 primary studies

papers), from which we obtained an initial set of 687 UML consistency rules (621 in Section 3.2) which were finally coalesced into 119 UML consistency rules (116 in Section 3.2). The general conclusions from Section 3.2 (about the primary studies, the rules, etc.) are the same as those presented in our journal paper, with more up to date data [17].

- **G2** (Is this list of UML diagram consistency rules relevant?) – Chapter 4.
    - 1st International Workshop on UML Consistency Rules (WUCOR) (Section 4.1). We organized WUCOR during the 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015) (Section 4.1) [18-20]. The goal of this workshop was to gather input and feedback on the 116 UML consistency rules (identified in Sections 3.1 and 3.2) from the MoDELS conference community. Due to limited attendance at the workshop, we were only able to receive feedback from experts in the field on 81 out of 116 UML consistency rules.
    - How consistency is handled in Model Driven Software Engineering and UML: a survey of experts in academia and industry (Section 4.2). This work provided the results of a survey that investigated how model consistency is handled in the context of MDSE and UML with experts from academia and industry. We developed a survey with MDSE experts from academia and industry to: (1) survey the diffusion and importance of model consistency issues; (2) validate the 116 UML consistency rules identified in Sections 3.1 and 3.2. The results of the work presented in Section 4.2 (i.e. [21]) will be submitted to an international journal in the near future.
    - UML consistency rules: a case study with Eclipse Papyrus models (Section 4.3). Starting with the set of 52 UML consistency rules identified in Section 4.2, the goal of this case study was to investigate to what extent existing UML models satisfy those rules. To do that, we described an exploratory flexible indirect case study [22] to verify the UML consistency rules on the UML models. The results of the work presented in Section 4.3 will be submitted to an international conference in the near future.

- Additional publications carried out in the context of this Ph.D. thesis:
    - On collecting and validating UML consistency rules: a research proposal, that has been published and presented the Doctoral Symposium at 18th International Conference on Evaluation and Assessment in Software Engineering (EASE) in 2014 [23].
    - On validating UML consistency rules, that has been published and presented at the 26th IEEE International Symposium on Software Reliability Engineering, Doctoral Symposium (ISSRE) in 2015 [24].
    - UML consistency rules in technical books, that has been published and presented at the 25th IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE) in 2015 [25].
    - Verifying the consistency of UML models, that has been published and presented at the 27th IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE) in 2016 [26].
    - Systematic Mapping Study (SMS) to collect the current state of the art in terms of consistency rules identified in UML synthesis techniques (see Appendix C). This work presents the results obtained after applying the SMS protocol with the aim of identifying and evaluating the current approaches for synthesizing UML diagrams from other UML diagrams. This work has been published and presented at the 10th Workshop on Modelling in Software Engineering (MiSE) in 2018 [27].

As follows, we present our results and observations in regard to G1 and G2 (in no particular order of importance):

- **G1** (What are the UML consistency rules proposed in the literature?)
    - From an initial set of 621 UML consistency rules, published in literature and extracted from seven scientific research databases, a total of 116 rules were eventually selected and analyzed in depth, by following a precise selection protocol driven by four research questions (see Appendix A). The UML diagram that is most involved in the final set of the 116 rules is the Class Diagram (62 rules, 53.45%), followed by the Interaction Diagram

(37 rules, 31.90%) and the State Machine Diagram (24 rules, 15.52%). This is not entirely surprising since these are likely the most frequently used UML diagrams [44]. However, another research [58] suggests that the Activity Diagram is the second most frequently used UML diagram after the Class Diagram. Considering the very scarce number of rules we found for the Activity Diagram (17 of 116 rules, 14.66%), we believe that future research should focus more on this diagram. However, if we compare the last UML 2.5 specification (released in 2015) [57] with the most frequently used UML 2.0 specification (released in 2005) [102], we can see that the UML has put more effort into the definition of activity diagrams, which consequently leads to (explicitly or otherwise) consistency rules concerning that diagram. Another diagram to which research on UML consistency rules has paid much less attention is the Use Case Diagram (22 rules, 15.52%). We believe that one of the reasons for these results is that the semantics of the use case diagram is simple and the semantics generally presents the include, extend and generalization, even though the semantics for this diagram is not, however, very clear to everyone: Cockburn argues [103] for instance that generalization between use cases is not clear and necessary, while other [104-106] argue that there are consistency rules between use case, and more so use case descriptions and other diagrams such as class and sequence diagrams. In addition, the mapping between semantics in the use case diagram and other diagrams is not clear either: for instance, if we have a generalization between use cases and use cases do map to two sequence diagrams, how does the generalization (in the use case) translate into relations between the two sequence diagrams? Besides the Class, Interaction, and State Machine diagrams, there is a need for much additional research on consistency rules involving other UML diagrams. For example, no rule was found for the Package, Component, Timing, Profile, Interaction Overview and Deployment Diagrams. One consideration that could partially explain the lack of rules for these diagrams is their overlap with the Class diagram syntax/semantics.

- The results show that the vast majority of the rules are horizontal and syntactic rules, and very few address vertical, semantic, invocation, observation and evolution consistency. One reason for this could be the fact that the UML syntax is more formally described (by the UML metamodel) in the specification than semantics, and creating syntactic rules may therefore prove more feasible. Another reason for this lack could be explained by the fact

that users mostly employ a few types of behavior diagrams that they understand or find suitable. Despite the fact that the topic of UML consistency is mature, it still needs to evolve to include more definitions of UML consistency rules in all dimensions. Very few rules address the issue of vertical and evolution consistency. Even though the UML consistency topic is mature, it still needs to evolve to include definitions of UML consistency rules in all dimensions. Our SMS therefore shows areas where future work is needed.

- We found that 49.76% of the rules were published between the years 2002 and 2005, and this period represents the most prolific period in literature in terms of publishing UML consistency rules. Not surprisingly, the UML version 2.0 is the most used standard to present UML consistency rules. Additionally, 38.97% (242 of 621) of the collected rules that have been proposed by researchers over the years describe redundant (similar or even identical) rules. This highlights the need for a central repository for such rules, such as this manuscript and our GitHub. Another adequate place in which to record those rules would be the UML specification itself. Moreover, we found that 36.26% (66 of 182) of non-redundant and accurate UML consistency rules presented by different authors had already been presented in one of the previous UML standards.
- The main software development activity that justified the description of UML diagram consistency rules is UML consistency verification, 51.89%, followed by impact analysis, 13.44%, UML model refinement & transformation, 12.74%, and management of consistency, 12.03%.

➢ **G2** (Is this list of UML diagram consistency rules relevant?)

- Regarding the 116 UML consistency rules we identified in Section 3.2, we conclude that 105 of the 116 (90.5%) UML consistency rules were considered important to check in UML models (always or at least in some situations) by the majority of the respondents (see Section 4.2.2.3.3). Based on our analysis of the rules' responses, we identified 52 out of 116 rules (see Section 4.2.2.3.3) that the vast majority (76.6%) of respondents considered to be relevant to enforce in every UML model.

- We believe that this subset of 52 rules can be valuable in the MDSE/UML contexts in many ways, such as: (a) it could be considered to be added to the UML standard, (b) it can be used as a reference to researchers who study UML and model driven technologies in general, and as a result, they will simply need to refer to this list rather than re-inventing the wheel [11], c) it can be used as a practical example in the classroom to better convey to students that UML is more than lines and boxes [28]. We are confident that the relatively large proportion of senior experts among the survey participants increases the reliability of our evaluations of UML consistency rules. Moreover, the results highlight the need to have both academics and practitioners as respondents in a survey like this and that our sample of respondents had significant expertise in the field we want to study, i.e., respondents primarily focus on MDSE topics, thereby indicating that our sample of respondents is somewhat representative of the overall target population.
- Starting with the set of 52 UML consistency rules identified in Section 4.2, in Section 4.3 we described an exploratory flexible indirect case study [22] to verify the UML consistency rules on the UML models. After the execution of the OCL rules on the models, we realized that only 30 OCL rules and 31 UML Papyrus models were finally used (see Section 4.3.2.1 for more details), and we identified a total of 2731 inconsistencies among the 206 UML diagrams that constitute the 31 UML models. 15 out of 30 OCL rules (50%) did not find any inconsistencies across the 31 UML Papyrus models. 12 out of these 15 OCL rules involved only one UML diagram. 12 out of the 18 (66.7%) OCL rules which involve only one UML diagram did not find any inconsistency. On the other hand, 13 out of 15 OCL rules (86.7%) which involve two UML diagrams revealed inconsistencies. This may indicate that designers are more comfortable designing single diagrams and find it hard to model consistent diagrams of different types (e.g., class diagram and sequence diagram); Seven out of 15 OCL rules (46.7%) found more than 100 inconsistencies. All those rules involved sequence diagrams. This may suggest that more training needs to be performed to better explain to designers how a sequence diagram relates to, i.e. should be consistent with, diagrams of other types;
- The OCL rules number 98 and 115 (see Appendix A), detected the highest number of inconsistencies, respectively 596 and 899 inconsistencies each. These two rules involve a

sequence diagram and a diagram of another type. On the other hand, 12 out of the 31 UML Papyrus models (38.7%) never triggered any of the 30 OCL rules.

- The class and sequence diagrams were the UML diagrams that were involved in the OCL rules that triggered the highest number of inconsistencies, respectively 1670 (61.2%) and 1830 (67.1%). In addition, an average of 89.9 inconsistencies were identified for each one of the 24 sequence diagrams involved in this study. Also, each one of the 12 OCL rules that involved a sequence diagram, triggered an average of 202.3 inconsistencies.

We have identified in our previous work [11] that there is still a small number of evaluations of consistency rules reported in the literature. For this reason, we believe that the works presented in Sections 4.1, 4.2, and 4.3 will contribute to the description of the best practices of model driven software engineering, where designers can use the information provided by this work to better understand UML inconsistencies. Moreover, the Sections 4.1, 4.2, and 4.3 represent the first research works that intends to find empirical evidence on how consistency issues are handled in the MDSE/UML contexts, and for that reason we expect that these results will have a solid impact in the field. We think that the detailed results presented in this Section will help academics and practitioners to identify the most relevant UML consistency rules, MDSE consistency terminology such as types of consistency, MDSE activities related to consistency, modeling languages, tools and technologies in the field.

It is important to note that some of our 116 UML consistency rules a) are difficult, and potentially impossible to enforce, b) cannot be captured in OCL, c) are relatively simplistic and do not apply to partial/incomplete models, and d) may be specific to a particular modeling approach. However, we believe that the final consolidated and validated set of 116 UML consistency rules resulting from this Ph.D. thesis will be used as a reference in the future by researchers in the field of UML model consistency.

As future work, we plan to submit the set of 52 rules described above to the UML revision task force[20] for inclusion in a forthcoming revision of the UML standard. Moreover, we consider consolidating further the list of OCL rules we have specified in this Ph.D. thesis. First, we intend to encode in OCL the remining 63 UML consistency rules (116-52) that we collected in a

[20] https://issues.omg.org/issues/lists/uml2-rtf

systematic way, and the 47 synthesis techniques (Appendix C) to check the UML consistency presented. Second, we will try to approach industry stakeholders that work in the field of MDSE, encouraging them to check our set of OCL rules directly on large industry models. In addition, since the OCL rules and the UML models are already freely available online[21], we will try to collect more data regarding models checked with our OCL rules in the following way: 1) we will add a note on GitHub to ask users to share with us basic info (e.g. who they are and what they are using the OCL rules for); 2) we will ask users to share with us some non-attributable metadata about the (in)consistencies they found in their models; 3) we will provide a script, to be run in Papyrus, that returns automatically some anonymized statistics about their model (e.g. different measures of size) and returns the (anonymized) results of the consistency verification (only number of rules triggered, UML diagrams cheeked, etc., similar to Table 15); 4) in return, we will provide to the users some statistical graphs via GitHub so users can compare their models with others. Another interesting future work would be the study of the result of the survey (presented in Section 4.2) by only considering the rules' evaluations from industry participants. An additional future work may use the set of rules identified in this thesis to detect inconsistencies, notify the users, recommend resolutions, and automatically fix the UML model similarly to what Liu and colleagues have done [84].

As final comment, we acknowledge that specific formalizations of subsets of the UML such as fUML[22], AFL[23], PSCS[24], PSSM[25], could facilitate the evaluation of some of our rules, and/or lead to the identification of new rules, and/or ultimately lead to making some of our rules obsolete.

[21] https://github.com/yvanlabiche/UML-model-consistency

[22] https://www.omg.org/spec/FUML/

[23] https://www.omg.org/spec/ALF/

[24] https://www.omg.org/spec/PSCS/

[25] https://www.omg.org/spec/PSSM/

References

[1] Y. Wang, "The cognitive Processes of Formal Inferences," *Int'l Journal of Cognitive Informatics and Natural Intelligence,* vol. 1, pp. 75-86, 2007.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice* 2nd ed.: Morgan & Claypool Publishers, 2016.

[3] L. Castiglioni and S. Mariotti, *Il Vocabolario della Lingua Latina. Latino - Italiano/Italiano*, 4th ed., 2008.

[4] J. Mukerji and J. Miller, "Overview and guide to OMG's architecture," Object Management Group http://www.omg.org/mda/, 2003.

[5] D. Thomas, "MDA: Revenge of the modelers or UML utopia?," *IEEE Software,* vol. 21, pp. 15–17, 2004.

[6] F. J. Lucas, F. Molina, and A. Toval, "A systematic review of UML model consistency management," *Information and Software Technology,* vol. 51, pp. 1631-1645, 2009.

[7] M. Genero, A. M. Fernández-Saez, H. J. Nelson, G. Poels, and M. Piattini, "A Systematic Literature Review on the Quality of UML Models," *Journal of Database Management,* vol. 22, pp. 46-70, July-September 2011 2011.

[8] M. Usman, A. Nadeem, K. Tai-hoon, and C. Eun-suk, "A Survey of Consistency Checking Techniques for UML Models," presented at the Advanced Software Engineering and Its Applications, Hainan Island, China, 2008.

[9] OMG, "OMG Unified Modeling LanguageTM - Superstructure Version 2.4.1," 2011.

[10] T. Pender, "UML Bible " 2003.

[11] D. Torre, Y. Labiche, and M. Genero, "UML consistency rules: a systematic mapping study," presented at the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014), London, UK, 2014.

[12] N. Ibrahim, R. Ibrahim, M. Z. Saringat, D. Mansor, and T. Herawan, "Consistency rules between UML use case and activity diagrams using logical approach," *International Journal of Software Engineering and its Applications,* vol. 5, pp. 119-134, 2011.

[13] J. Simmonds, R. V. Straeten, V. Jonkers, and T. Mens, "Maintaining Consistency between UML Models using Description LogicZ," *RSTI – L'Object LMO'04,* vol. 10, pp. 231-244, 2004.

[14] J. Muskens, R. J. Bril, and M. R. V. Chaudron, "Generalizing Consistency Checking between Software Views," presented at the 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, Pennsylvania, USA, 2005.

[15] Z. Huzar, L. Kuzniarz, G. Reggio, and J. L. Sourrouille, "Consistency problems in UML-based software development," presented at the International Conference on UML Modeling Languages and Applications, Lisbon, Portugal, 2005.

[16] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," in *Handbook of Software Engineering and Knowledge Engineering*, S. K. Chang, Ed., ed Singapore: World Scientific Publishing Co., 2001, pp. 329-380.

[17] D. Torre, Y. Labiche, M. Genero, and M. Elaasar, "A systematic identification of consistency rules for UML diagrams," *Journal of Systems and Software,* vol. 144, pp. 121-142, 2018.

[18] D. Torre, Y. Labiche, M. Genero, M. Elaasar, T. K. Das, B. Hoisl*, et al.*, "1st International Workshop on UML Consistency Rules (WUCOR 2015): Post workshop report," *SIGSOFT Softw. Eng. Notes,* vol. 41, pp. 34-37, 2016.

[19] D. Torre, Y. Labiche, M. Genero, and M. Elaasar, "Introduction to WUCOR 2015," in *1st International Workshop on UML Consistency Rules (WUCOR 2015) co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015)*, Ottawa, 2015.

[20] I. Dragomir, S. Graf, G. Karsai, F. Noyrit, I. Ober, D. Torre*, et al.*, "Joint Proceedings of the 8th International Workshop on Model-based Architecting of Cyber-physical and Embedded Systems and 1st International Workshop on UML Consistency Rules (ACES-MB 2015 & WUCOR 2015) co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015)," Ottawa, Canada, 2015.

[21] D. Torre, M. Genero, Y. Labiche, and M. Elaasar, "How consistency is handled in Model Driven Software Engineering: a survey of experts in academia and industry," Carleton University https://goo.gl/QVXTxc, 2018.

[22] P. Runeson, M. Host, A. Rainer, and B. Regnell., *Case Study Research in Software Engineering: Guidelines and Examples* 1st ed.: Wiley Publishing, 2012.

[23] D. Torre, "On collecting and validating UML consistency rules: a research proposal," presented at the Doctoral Symposium at 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014), London, UK, 2014.

[24] D. Torre, "On validating UML consistency rules," presented at the 26th IEEE International Symposium on Software Reliability Engineering, Doctoral Symposium (ISSRE 2015) Gaithersburg, MD, USA, 2015.

[25] D. Torre, Y. Labiche, M. Genero, and M. Elaasar, "UML consistency rules in technical books," presented at the IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE 2015), Gaithersburg, MD, USA, 2015.

[26] D. Torre, "Verifying the consistency of UML models," presented at the 27th IEEE International Symposium on Software Reliability Engineering, Fast Abstract (ISSRE 2016) Ottawa, Canada, 2016.

[27] D. Torre, Y. Labiche, M. Genero, M. T. Baldassarre, and M. Elaasar, "UML diagram synthesis techniques: a systematic mapping study," presented at the the 10th Workshop on Modelling in Software Engineering (MiSE'2018), 2018.

[28] Y. Labiche, "The UML is more than boxes and lines," presented at the Workshops and Symposia at MODELS 2008, Toulouse, France, 2008.

[29] D. Kalibatiene, O. Vasilecas, and R. Dubauskaite, "Rule Based Approach for Ensuring Consistency in Different UML Models," presented at the 6th SIGSAND/PLAIS EuroSymposium 2013, Gdańsk, Poland, 2013.

[30] M. A. Ahmad and A. Nadeem, "Consistency checking of UML models using Description Logics: A critical review," presented at the 6th International Conference on Emerging Technologies Islamabad, Pakistan, 2010.

[31] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University2007.

[32] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information & Software Technology,* vol. 64, pp. 1-18, 2015.

[33] B. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*: Chapman & Hall/CRC., 2015.
[34] R. Wendler, "The maturity of maturity model research: A systematic mapping study," *Inf. Softw. Technol.,* vol. 54, pp. 1317-1339, December 2012 2012.
[35] R. Giuffrida and Y. Dittrich, "Empirical studies on the use of social software in global software development - A systematic mapping study," *Information & Software Technology,* vol. 55, pp. 1143-1164, 2013.
[36] A. Fernandez, E. Insfran, and S. Abrahão, "Usability evaluation methods for the web: A systematic mapping study," *Inf. Softw. Technol.,* vol. 53, pp. 789-817, August 2011 2011.
[37] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using mapping studies in software engineering," presented at the Proceedings of the Psychology of Programming Interest Group Workshop, Lancaster University, 2008.
[38] P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software,* vol. 80, pp. 571–583, 2007.
[39] D. Torre, Y. Labiche, and M. Genero, "UML consistency rules: a systematic mapping study," Carleton University, Ottawa http://squall.sce.carleton.ca/mediawiki/index.php/Technical_reports, 2014.
[40] T. Mens, R. Van der Straeten, and J. Simmonds, "A framework for managing consistency of evolving UML models," in *Software Evolution with UML and XML*, ed: IGI Publishing, 2005, pp. 1-30.
[41] G. Engels, J. H. Hausmann, and R. Heckel, "Testing the consistency of dynamic UML diagrams," presented at the Integrated Design and Process Technology, Pasadena, California, 2002.
[42] G. Engels, J. M. Küster, R. Heckel, and L. Groenewegen, "A methodology for specifying and analyzing consistency of object-oriented behavioral models," *Sigsoft Software Engineering Notes,* vol. 26, pp. 186-195, September 2001 2001.
[43] R. Wieringa, N. A. M. Maiden, N. R. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements Eng.,* vol. 11, pp. 102–107, 2006.
[44] B. Dobing and J. Parsons, "How UML is used," *ACM,* vol. 49, pp. 109-113, 2006.
[45] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE 2008), Bari, Italy, 2008.
[46] Z. Manna and R. J. Waldinger, "Toward automatic program synthesis. Commun.," *ACM* vol. 14, pp. 151-165, 1971.
[47] IEEE, "IEEE Standard for Software VeriÞcation and Validation Plans (IEEE Std 1012-1986)," 1992.
[48] R. F. Paige, D. S. Kolovos, and F. A. C. Polack, "Refinement via Consistency Checking in MDA," *Electronic Notes in Theoretical Computer Science,* vol. 137, pp. 151-161, 2005.
[49] Z. Pap, I. Majzik, A. Pataricza, and A. Szegi, "Methods of checking general safety criteria in UML statechart specifications," *Reliability Engineering & System Safety,* vol. 87, pp. 89-107, 2005.
[50] O. Pilskalns, D. Williams, D. Aracic, and A. Andrews, "Security Consistency in UML Designs," presented at the 30th Annual International Computer Software and Applications Conference, Chicago, USA, 2006.

[51] L. C. Briand, Y. Labiche, and L. O'Sullivan, "Impact analysis and change management of UML models," presented at the International Conference on Software Maintenance, Amsterdam, The Netherlands, 2003.

[52] I.-Y. Song, R. Khare, Y. An, and M. Hilsbos, "A Multi-level Methodology for Developing UML Sequence Diagrams," presented at the 27th International Conference on Conceptual Modeling, Barcelona, Spain, 2008.

[53] J. Yang, "A framework for formalizing UML models with formal language rCOS," presented at the 4th International Conference on Frontier of Computer Science and Technology, Shanghai, China, 2009.

[54] B. Hoisl and S. Sobernig, "Consistency Rules for UML-based Domain-specific Language Models: A Literature Review," in *1st International Workshop on UML Consistency Rules (WUCOR 2015) co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015)*, Ottawa, Canada, 2015.

[55] S. Sobernig, B. Hoisl, and M. Strembeck, "Extracting reusable design decisions in UML-based domain-specific languages: A multi-method study," *Journal of Systems and Software,* vol. 113, pp. 140-172, 2016.

[56] D. Chiorean, V. Petraşcu, and I. Chiorean, "Proposal for Improving the UML Abstract Syntax," in *1st International Workshop on UML Consistency Rules (WUCOR 2015) co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015)*, Ottawa, Canada, 2015.

[57] OMG, "OMG Unified Modeling LanguageTM - Superstructure Version 2.5," 2015.

[58] G. Reggio, M. Leotta, F. Ricca, and D. Clerissi, "What are the used UML diagrams? A Preliminary Survey," presented at the In Proceedings of 3rd International Workshop on Experiences and Empirical Studies in Software Modeling - CEUR Workshop Proceedings (EESSMod 2013), Miami, Florida - USA, 2013.

[59] A. Pinsonneault and K. L. Kraemer, "Survey research methodology in management information systems: an assessment," *Journal of Management Information Systems - Special Section: Strategic and competitive information systems,* vol. 10, pp. 75–105, 1993.

[60] D. Akdur, O. Demirörs, and V. Garousi, "The results of a world-wide survey on software modeling and Model-Driven engineering in the embedded system domain," *Technical Report METU/II-2015-55,* 2015.

[61] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-Driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming,* vol. 89, pp. 144-161, 2014.

[62] A. M. Fernández-Sáez, M. Genero, D. Caivano, and M. R. V. Chaudron, "On the Use of UML Documentation in Software Maintenance: Results from a Survey in Industry," presented at the 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE, 2015.

[63] E. Arisholm, L. C. Briand, and B. C. D. Anda, "First workshop on empirical studies of Model-Driven engineering at MODELS 2008," 2008.

[64] D. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*: John Wiley & Sons, 2003.

[65] A. Pinsonneault and K. L. Kraemer, "Survey research methodology in management information systems: an assessment," *Journal of Management Information Systems,* vol. 10, pp. 75–105, 1993.

[66] R. M. Groves, F. J. Fowler, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau, *Survey Methodology*, Second ed.: John Wiley & Sons, 2009.
[67] B. A. Kitchenham and S. L. Pfleeger, "Personal Opinion Surveys," in *Guide to Advanced Empirical Software Engineering*, ed: Springer London, 2008, pp. 63--92.
[68] J. Linaker, S. M. Sulaman, R. M. d. Mello, and M. Höst, "Guidelines for Conducting Surveys in Software Engineering, TR 5366801," Lund University 2015.
[69] T. Punter, M. Ciolkowski, B. Freimut, and I. John, "Conducting on-line surveys in software engineering," presented at the International Symposium on Empirical Software Engineering, 2003.
[70] (last access on November 2016). *The International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Available: www.modelsconference.org
[71] (last access on November 2016). *The Workshop on Modelling in Software Engineering (MiSE)*. Available: https://sselab.de/lab2/public/wiki/MiSE/
[72] A. Jedlitschka, M. Ciolkowski, C. Denger, B. Freimut, and A. Schlichting, "Relevant information sources for successful technology transfer: a survey using inspections as an example," presented at the the International Symposium on Empirical Software Engineering and Measurement (ESEM), 2007.
[73] J. A. Krosnick, "Survey research," *Annual Review of Psychology,* vol. 50, pp. 537–567, 1990.
[74] L. Telinski, W. Agner, I. W. Soares, P. C. Stadzisz, and J. M. Simão, "A Brazilian survey on UML and Model-Driven practices for embedded software development," *J. Syst. Softw. ,* vol. 86, pp. 997-1005, 2013.
[75] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula, "Survey on agile and lean usage in finnish software industry," presented at the the ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '12), 2012.
[76] G. Scanniello, C. Gravino, and G. Tortora, "Investigating the Role of UML in the Software Modeling and Maintenance - A Preliminary Industrial Survey. ," presented at the the 12th International Conference on Enterprise Information Systems (ICEIS'2010), Funchal, Madeira, Portugal, 2010.
[77] M. Petre, "UML in practice," presented at the 35th International Conference on Software Engineering, San Francisco, CA, USA, 2013.
[78] A. Forward, O. Badreddin, and T. C. Lethbridge, "Perceptions of Software Modeling: A Survey of Software Practitioners," presented at the 5th Workshop From code centric to model centric: Evaluating the effectiveness of MDD (C2M:EEMDD), Paris, France, 2010.
[79] OMG. (2016). *Object Management Group - Object Constraint Language (OCL)*.
[80] NATO, "SOFTWARE ENGINEERING," presented at the Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 1968.
[81] A. Reder and A. Egyed, "Determining the Cause of a Design Model Inconsistency," *IEEE Trans. Softw. Eng. ,* vol. 39, pp. 1531-1548, November 2013 2013.
[82] M. Gogolla, L. Hamann, F. Hilken, and M. Sedlmeier, "Checking UML and OCL Model Consistency: An Experience Report on a Middle-Sized Case Study," presented at the International Conference on Tests and Proofs, 2015.
[83] K. Czarnecki and K. Pietroszek, "Verifying feature-based model templates against well-formedness OCL constraints," presented at the 5th international conference on Generative programming and component engineering (GPCE '06), 2006.

[84] W. Liu, S. Easterbrook, and J. Mylopoulos, "Rule-Based Detection of Inconsistency in UML Models," presented at the Workshop on Consistency Problems in UML-Based Software Development, 2002.
[85] P. G. Sapna and H. Mohanty, "Ensuring consistency in relational repository of UML models," presented at the 10th International Conference on Information Technology, Orissa, India, 2007.
[86] J. W. Anastas and M. L. MacDonald, *Research design for the social work and the human services*: Lexington, 1994.
[87] C. Robson, *Real World Research*, 2nd ed.: Blackwell, 2002.
[88] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering,* vol. 14, pp. 131-164, 2009.
[89] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: data collection techniques for software field studies. ," *Empirical Softwtware Engineering* vol. 10, pp. 311–341, 2005.
[90] I. Dragomir and I. Ober, "Well-formedness and typing rules for UML Composite Structures," 2010.
[91] A. Egyed and A. Reder. (2016, last access on August 2016). *Examples of Design Rules*. Available: http://goo.gl/MJkJnB
[92] (2016, last access on August 2016). *UML Repository*. Available: http://models-db.com/
[93] R. Hebig, T. H. Quang, M. R. V. Chaudron, G. Robles, and M. A. Fernandez, "The Quest for Open Source Projects that Use UML: Mining GitHub," presented at the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, 2016.
[94] D. I. K. Sjoberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg*, et al.*, "A Survey of Controlled Experiments in Software Engineering," *IEEE Trans. Softw. Eng.,* vol. 31, pp. 733-753, 2005.
[95] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*: Springer Publishing Company, Incorporated, 2012.
[96] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empirical Software Enginering,* vol. 19, pp. 1565–1616, 2014.
[97] C. Wohlin, "Writing for synthesis of evidence in empirical software engineering," presented at the the 8th ACM/IEEE International Symposium on Empirical Software Engieneering and Measurement, 2014.
[98] B. Dathan and S. Ramnath, *Object-Oriented Analysis, Design and Implementation: An Integrated Approach*, 2nd ed.: Springer Publishing Company, Incorporated, 2015.
[99] D. Badampudi, C. Wohlin, and K. Petersen, "Software component decision-making: In-house, OSS, COTS or outsourcing - A systematic literature review," *The Journal of Systems and Software,* vol. 121, pp. 105–124, 2016.
[100] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction.* Norwell, MA, USA: Kluwer Academic Publishers, 2000.
[101] J. A. Maxwell, "Using numbers in qualitative research," *Qualitative Inquiry,* vol. 16, pp. 475–482, 2010.
[102] OMG, "OMG Unified Modeling LanguageTM - Superstructure Version 2.0," 2005.

[103] A. Cockburn, *Writing Effective Use Cases*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.
[104] N. Ibrahim, R. Ibrahim, and M. Z. Saringat, "Definition of Consistency Rules between UML Use Case and Activity Diagram," presented at the 2nd International Conference, Daejeon, South Korea, 2011.
[105] J. Chanda, A. Kanjilal, S. Sengupta, and S. Bhattacharya, "Traceability of Requirements and Consistency Verification of UML Use case, Activity and Class Diagram: A Formal Approach," presented at the IEEE ICM2CS New Delhi, 2009.
[106] L. Fryz and L. Kotulski, "Assurance of System Consistency During Independent Creation of UML Diagrams," presented at the 2nd International Conference on Dependability of Computer Systems (DEPCOS-RELCOMEX '07), 2007.
[107] M. Stumptner and M. Schrefl, "Behavior consistent inheritance in UML," presented at the 19th international conference on Conceptual modeling, Salt Lake City, Utah, USA, 2000.
[108] J. Seiter, R. Wille, U. Kühne, and R. Drechsler, "Automatic refinement checking for formal system models," presented at the Forum on Specification and Design Languages (FDL), Munich, Germany, 2014.
[109] L. A. Campbell, B. H. C. Cheng, W. E. McUmber, and R. E. K. Stirewalt, "Automatically Detecting and Visualising Errors in UML Diagrams," *Requirements Engineering,* vol. 7, pp. 264-287, 2002.
[110] C. Schwarzl and B. Peischl, "Static- and dynamic consistency analysis of UML state chart models," presented at the 13th international conference on Model driven engineering languages and systems: Part I, Oslo, Norway, 2010.
[111] H. Qiu, "Consistency Checking of UML-LIGHT with CSP," WS 2004/2005: Seminar : Modellbasierte Softwareentwichklung Department of Computer Science, University of Paderborn, Paderborn, Germany 2004.
[112] G. Costagliola, V. Deufemia, F. Ferrucci, and C. Gravino, "Exploiting Visual Languages Generation and UML Meta Modeling to Construct Meta-CASE Workbenches," *Electronic Notes in Theoretical Computer Science,* vol. 72, pp. 25-35, 2003.
[113] E. Astesiano and G. Reggio, "An attempt at analysing the consistency problems in the UML from a classical algebraic viewpoint," presented at the 16th International Workshop, Frauenchiemsee, Germany, 2003.
[114] C. F. Borba and A. E. A. Da SIlva, "Knowledge-based system for the maintenance registration and consistency among UML diagrams," presented at the 20th Brazilian Conference on Advances in Artificial Intelligence, São Bernardo do Campo, Brazil, 2010.
[115] M. Hilsbos and I.-Y. Song, "Use of Tabular Analysis Method to Construct UML Sequence Diagrams," presented at the 23th International Conference on Conceptual Modeling, Shanghai, China, 2004.
[116] J. Yang, Q. Long, Z. Liu, and X. Li, "A predicative semantic model for integrating UML models," presented at the 1st international Conference on Theoretical Aspects of Computing, Guiyang, China, 2004.
[117] L. Quan, L. Zhiming, L. Xiaoshan, and J. He, "Consistent code generation from UML models," presented at the Australian Conference on Software Engineering, Brisbane, Australia, 2005.
[118] J. H. Hausmann, R. Heckel, and S. Sauer, "Extended model relations with graphical consistency conditions," presented at the UML 2002 Workshop on Consistency Problems in UML-based Software Development, Blekinge Institute of Technology, 2002.

[119] A. Ohnishi, "A supporting system for verification among models of the UML," *Systems and Computers in Japan,* vol. 33, pp. 1-3, April 2002 2002.
[120] J. Chanda, A. Kanjilal, S. Sengupta, and S. Bhattacharya, "Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach," presented at the International Conference on Methods and Models in Computer Science, New Delhi, India, 2009.
[121] A. Kanjilal, S. Sengupta, and S. Bhattacharya, "Analysis of object-oriented design: A metrics based approach," presented at the IEEE Region 10 Annual International Conference, Singapore; Singapore, 2009.
[122] X. Liu, "Identification and check of inconsistencies between UML diagrams," presented at the 2013 International Conference on Computer Sciences and Applications, 2013.
[123] H. Il-Kyu and K. Byung-Wook, "Meta-validation of UML structural diagrams and behavioral diagrams with consistency rules," presented at the IEEE Pacific Rim Conference on Communications, Computers and signal Processing, Victoria, B.C., Canada, 2003.
[124] S. K. Kim and D. Carrington, "A formal object-oriented approach to defining consistency constraints for UML models," presented at the Australian Conference on Software Engineering, Melbourne, Australia, 2004.
[125] X. Xia, J. Shi, Z. Fan, Z. Ai, and Y. Dong, "A Consistency Checking Approach for System Architecture," presented at the Annual IEEE Systems Conference (SysCon), Orlando, FL, USA, 2016
[126] J. Kienzle, W. A. Abed, and J. Klein, "Aspect-oriented multi-view modeling," presented at the 8th ACM international conference on Aspect-oriented software development, Charlottesville, Virginia, USA, 2009.
[127] O. Vasilecas, R. Dubauskaitė, and R. Rupnik, "Consistency checking of UML business model," *Technological and Economic Development of Economy,* vol. 17, pp. 133-150, 2011.
[128] I. Zaretska, O. Kulankhina, and H. Mykhailenko, "Cross-Diagram UML Design Verification," presented at the ICT in Education, Research, and Industrial Applications (ICTERI), 2013.
[129] N. Pattanasri, V. Wuwongse, and K. Akama, "XET as a Rule Language for Consistency Maintenance in UML," presented at the 3rd International Workshop, Hiroshima, Japan, 2004.
[130] A. Zisman and A. Kozlenkov, "Knowledge Base Approach to Consistency Management of UML Specifications," presented at the 16th IEEE international conference on Automated software engineering, Coronado Island, San Diego, CA, USA, 2001.
[131] F. J. L. Martínez and A. T. Álvarez, "A precise approach for the analysis of the UML models consistency," presented at the 24th international conference on Perspectives in Conceptual Modeling, Klagenfurt, Austria, 2005.
[132] B. Bjørn, "Consistency checking UML interactions and state machines," Master Thesis, Department of Informatics, University of Oslo, Oslo, Norway, 2008.
[133] D. Du, J. Liu, H. Cao, and M. Zhang, "BAS: A Case Study for Modeling and Verification in Trustable Model Driven Development," *Electronic Notes in Theoretical Computer Science,* vol. 243, pp. 69-87, July 2009 2009.
[134] D. Ruta and V. Olegas, "The approach of ensuring consistency of UML model based on rules," presented at the 11th International Conference on Computer Systems and

Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies, Sofia, Bulgaria, 2010.

[135] A. Egyed, "Fixing Inconsistencies in UML Design Models," presented at the 29th international conference on Software Engineering Minneapolis, MN, USA, 2007.

[136] H. Wang, T. Feng, J. Zhang, and K. Zhang, "Consistency check between behaviour models," presented at the International Symposium on Communications and Information Technology, Beijing, China, 2005.

[137] P. Pelliccione, P. Inverardi, and H. Muccini, "CHARMY: A Framework for Designing and Verifying Architectural Specifications," *IEEE Transactions on Software Engineering,* vol. 35, pp. 325-346, 2009.

[138] J. Kuster and J. Stroop, "Consistent design of embedded real-time systems with UML-RT," presented at the 4th International Symposium on Object-Oriented Real-Time Distributed Computing, Magdeburg, Germany, 2001.

[139] W. Shengjun, J. Longfei, and J. Chengzhi, "Ontology Definition Metamodel based Consistency Checking of UML Models," presented at the 10th International Conference on Computer Supported Cooperative Work in Design, Nanjing, China, 2006.

[140] K. N. Chang, "Model checking consistency between sequence and state diagrams," presented at the International Conference on Software Engineering Research and Practice, Las Vegas, NV, USA, 2008.

[141] A. Egyed, E. Letier, and A. Finkelstein, "Generating and evaluating choices for fixing inconsistencies in UML design models," presented at the 23rd IEEE/ACM International Conference on Automated Software Engineering, L'Aquila, Italy, 2008.

[142] A. Nimiya, T. Yokogawa, H. Miyazaki, S. Amasaki, Y. Sato, and M. Hayase, "Model checking consistency of UML diagrams using alloy," *World Academy of Science, Engineering and Technology,* vol. 71, pp. 547-550, 2010.

[143] X. Zhao, Q. Long, and Z. Qiu, "Model checking dynamic UML consistency," presented at the 8th International Conference on Formal Engineering Methods, Macao, China, 2006.

[144] Y. Hammal, "A modular state exploration and compatibility checking of UML dynamic diagrams," presented at the 6th IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, 2008.

[145] K. Lano, "Formal specification using interaction diagrams," presented at the 5th IEEE International Conference on Software Engineering and Formal Methods, London; United Kingdom, 2007.

[146] M. N. Alanazi and D. A. Gustafson, "Super state analysis for UML state diagrams," presented at the 2009 WRI World Congress on Computer Science and Information Engineering Los Angeles, California, USA, 2009.

[147] T. Yokogawa, S. Amasaki, K. Okazaki, Y. Sato, K. Arimoto, and H. Miyazaki, "Consistency Verification of UML Diagrams Based on Process Bisimulation," presented at the IEEE 19th Pacific Rim International Symposium on Dependable Computing (PRDC), 2013.

[148] A. Knapp, T. Mossakowski, and M. Roggenbach, "Towards an Institutional Framework for Heterogeneous Formal Development in UML," *Software, Services, and Systems,* pp. 215-230, 2015.

[149] H. Nakanishi, T. Miura, and I. Shioya, "Formalizing UML collaborations by using description logics," presented at the 2nd IEEE International Conference on Computational Cybernetics, Vienna, Austria, 2004.

[150] C. M. Zapata, G. González, and A. Gelbukh, "A rule-based system for assessing consistency between UML models," presented at the 6th Mexican International Conference on Advances in Artificial Intelligence, Aguascalientes, Mexico, 2007.
[151] R. g. Laleau and F. Polack, "Using formal metamodels to check consistency of functional views in information systems specification," *Information and Software Technology,* vol. 50, pp. 797-814, 2008.
[152] R. F. Paige, P. J. Brooke, and J. S. Ostroff, "Metamodel-based model conformance and multiview consistency checking," *ACM Transactions Software Engineering Methodology,* vol. 16, p. 11, July 2007 2007.
[153] H. Il-kyu and B. Kang, "Cross Checking Rules to Improve Consistency between UML Static Diagram and Dynamic Diagram," presented at the 9th International Conference on Intelligent Data Engineering and Automated Learning, Daejeon, South Korea, 2008.
[154] H. Hamed and A. Salem, "UML-L: a UML based design description language," presented at the ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, 2001.
[155] T. Mens, R. Van Der Straeten, and M. D'Hondt, "Detecting and resolving model inconsistencies using transformation dependency analysis," presented at the 9th International conference on Model Driven Engineering Languages and Systems, Genova, Italy, 2006.
[156] S. Sengupta and S. Bhattacharya, "Formalization of UML diagrams and their consistency verification: A Z notation based approach," presented at the 1st India software engineering conference, Hyderabad, India, 2008.
[157] A. Egyed, "Consistent adaptation and evolution of class diagrams during refinement," presented at the 7th International Conference (FASE 2004), Held as Part of the Joint European Conferences on Theory and Practice of Software, Barcelona, Spain, 2004.
[158] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini, "A Formal Framework for Reasoning on UML Class Diagrams," presented at the 13th International Symposium on Foundations of Intelligent Systems, Lyon, France, 2002.
[159] W. Shen, K. Wang, and A. Egyed, "An efficient and scalable approach to correct class model refinement," *IEEE Transactions on Software Engineering,* vol. 35, pp. 515-533, 2009.
[160] R. Van Der Straeten, "Inconsistency detection between UML models using RACER and nRQL," presented at the the KI-2004 Workshop on Applications of Description Logics, Ulm, Germany, 2004.
[161] G. Engels, R. Heckel, and J. M. Küster, "Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model," presented at the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, Toronto, Ontario, Canada, 2001.
[162] R. Van Der Straeten, V. Jonckers, and T. Mens, "A formal approach to model refactoring and model refinement," *Software & Systems Modeling,* vol. 6, pp. 139-162, 2007.
[163] R. Wagner, H. Giese, and U. Nickel, "A plug-in for flexible and incremental consistency management," presented at the International Conference on the Unified Modeling Language 2003 (Workshop 7: Consistency Problems in UML-based Software Development), San Francisco, USA, 2003.

[164] L. Jing, L. Zhiming, H. Jifeng, and L. Xiaoshan, "Linking UML models of design and requirement," presented at the Australian Conference on Software Engineering, Melbourne, Australia, 2004.
[165] K. C. Lavanya, K. V. Bala, H. Mohanty, and R. K. Shyamasundar, "How Good is a UML Diagram? A Tool to Check It," presented at the IEEE Region 10, Melbourne, Australia, 2005.
[166] M. Snoeck, C. Michiels, and G. Dedene, "Consistency by construction: the case of MERODE," presented at the Conceptual Modeling for Novel Application Domains, Workshops ECOMO, IWCMQ, AOIS, and XSDM, Chicago, IL, USA, 2003.
[167] I. Ober and I. Dragomir, "Unambiguous UML composite structures: the OMEGA2 experience," presented at the 37th international conference on Current trends in theory and practice of computer science, Nový Smokovec, Slovakia, 2011.
[168] G. Spanoudakis and H. Kim, "Diagnosis of the significance of inconsistencies in object-oriented designs: a framework and its experimental evaluation," *Journal of Systems and Software,* vol. 64, pp. 3-22, October 2002 2002.
[169] A. Egyed, "UML/Analyzer: A Tool for the Instant Consistency Checking of UML Models " presented at the 29th international conference on Software Engineering, Minneapolis, MN, USA, 2007.
[170] A. Egyed, "Instant consistency checking for the UML," presented at the 28th international conference on Software engineering, Shanghai, China, 2006.
[171] H. Ledang, "B-based Consistency Checking of UML Diagrams," presented at the 2nd National Symposium on Research, Development and Application of Information and Communication Technology, Hanoi, Vietnam, 2004.
[172] R. Van Der Straeten, "ALLOY for Inconsistency Resolution in MDE," presented at the BENEVOL 2009 The 8 th BElgian-NEtherlands software eVOLution seminar, Université catholique de Louvain - Belgium, 2009.
[173] J. Chanda, T. Janowski, H. Mohanty, A. Kanjilal, and S. Sengupta, "UML-Compiler: A Framework for Syntactic and Semantic Verification of UML Diagrams," presented at the 6th international conference on Distributed Computing and Internet Technology, Bhubaneswar, India, 2010.
[174] Z. Khai, A. Nadeem, and G.-s. Lee, "A Prolog Based Approach to Consistency Checking of UML Class and Sequence Diagrams," presented at the Communication and Networking - International Conference (FGCN 2011), Held as Part of the Future Generation Information Technology Conference (FGIT 2011), in Conjunction with GDC 2011, Jeju Island, South Korea, 2011.
[175] R. Van Der Straeten, T. Mens, J. Simmonds, and V. Jonckers, "Using Description Logic to Maintain Consistency between UML Models," presented at the 6th International Conference on Unified Modeling Language, Modeling Languages and Applications, San Francisco, CA, USA, 2003.
[176] L. Xiaoshan, L. Zhiming, and J. He, "A formal semantics of UML sequence diagram," presented at the Australian Conference on Software Engineering, Melbourne, Australia, 2004.
[177] G. Spanoudakis, K. Kasis, and F. Dragazi, "Evidential diagnosis of inconsistencies in object-oriented designs," *International Journal of Software Engineering and Knowledge Engineering,* vol. 14, pp. 141-178, 2004.

[178] A. Reder and A. Egyed, "Computing repair trees for resolving inconsistencies in design models," presented at the 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, 2012.
[179] O. Vasilecas and R. Dubauskaite, "Ensuring consistency of information systems rules models," presented at the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, Ruse, Bulgaria, 2009.
[180] D. Allaki, M. Dahchou, and A. En-Nouaary, "Detecting and fixing UML model inconsistencies using constraints," presented at the 4th IEEE International Colloquium on Information Science and Technology (CiSt), 2016.
[181] C. F. J. Lange and M. R. V. Chaudron, "Effects of defects in UML models: an experimental investigation," presented at the 28th international conference on Software engineering, Shanghai, China, 2006.
[182] P. Selonen, K. Koskimies, and M. Sakkinen, "Transformations between UML diagrams," *Journal of Database Management,* vol. 14, pp. 37-55, 2003.
[183] P. Selonen, K. Koskimies, and M. Sakkinen, "How to Make Apples from Oranges in UML," presented at the 34th Annual Hawaii International Conference on System Sciences (HICSS '01), 2001.
[184] E. Mäkinen and T. Systä, "MAS — an interactive synthesizer to support behavioral modelling in UML," presented at the 23rd International Conference on Software Engineering (ICSE '01), Washington, DC, USA, 2001.
[185] H. Arksey and L. O'Malley, "Scoping studies: towards a methodological framework," *International Journal of Social Research Methodology,* vol. 8, 2005.
[186] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements Eng.,* vol. 11, pp. 102-107, 2005.
[187] J. Whittle and P. K. Jayaraman, "Synthesizing hierarchical state machines from expressive scenario descriptions," *ACM Trans. Softw. Eng. Methodol.,* vol. 19, 2010.
[188] S. Kang, H. Kim, J. Baik, H. Choi, and C. Keum, "Transformation Rules for Synthesis of UML Activity Diagram from Scenario-Based Specification " presented at the IEEE 34th Annual Computer Software and Applications Conference (COMPSAC '10), 2010.
[189] T. Ziadi, L. Helouet, and J.-M. Jezequel, "Revisiting Statechart Synthesis with an Algebraic Approach " presented at the 26th International Conference on Software Engineering (ICSE '04), 2004.
[190] J. Schumann, "Automatic debugging support for uml designs," presented at the 4th International Workshop on Automated Debugging, 2000.
[191] I. Khriss, M. Elkoutbi, and R. K. Keller, "Automating the Synthesis of UML StateChart Diagrams from Multiple Collaboration Diagrams," presented at the 1st International Workshop on The Unified Modeling Language: Beyond the Notation (UML '98), 1998.
[192] T. Maier and A. Zndorf, "The Fujaba Statechart Synthesis Approach," presented at the Workshop on Scenarios and State Machines (ICSE '03), Portland, Oregon, USA, 2003.
[193] P. Selonen and T. Systä, "Scenario-based Synthesis of Annotated Class Diagrams in UML," presented at the Workshop: Scenario-based round-trip engineering (OOPSLA '00) Tampere University of Technology, Software Systems Laboratory, 2000.
[194] R. Eshuis and P. V. Gorp, "Synthesizing object life cycles from business process models," *Software & Systems Modeling,* vol. 15, pp. 281–302, 2106.

[195] J. Whittle and J. Schumann, "Generating statechart designs from scenarios," presented at the 22nd international conference on Software engineering (ICSE '00), 2000

[196] J. Schumann and J. Whittle, "Automatic Synthesis of Agent Designs in UML," presented at the 1st International Workshop on Formal Approaches to Agent-Based Systems (FAABS '00), 2000.

[197] I. Krka and N. Medvidovic, "Component-Aware Triggered Scenarios," presented at the 2014 IEEE/IFIP Conference on Software Architecture (WICSA '14), 2014.

[198] T. Systä, "Incremental construction of dynamic models for object-oriented software systems," *Journal of Object-Oriented Programming,* vol. 13, pp. 18-27, 2000.

[199] M. T. Baldassarre, J. Carver, O. Dieste, and N. Juristo, "Replication types: Towards a shared taxonomy," *ACM International Conference Proceeding Series,* 2014.

[200] F. J. Lucas, F. Molina, and A. Toval, "A systematic review of UML model consistency management," *Information and Software Technology,* vol. 51, pp. 1631–1645, December 2009.

[201] M. Siikarla, J. Peltonen, and P. Selonen. "Combining OCL and Programming Languages for UML Model Processing. Electron". *Notes Theor. Comput. Sci.* vol. 102, pp. 175-194. 2004.

[202] K. Petersen, S. Vakkalanka, and L Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update" *Information and Software Technology*, vol. 64, pp. 1-18, 2014.

Appendix A

This appendix contains Table 18, where the final set of 116 UML consistency rules is presented. The following Table contains:

- type(s) of UML diagram(s) involved in the UML consistency rules (grey row) with the appropriate ID# presented in Table 18);
- the univocal number of each UML consistency rule;
- the description of the UML consistency rules;
- UML consistency dimensions (for definitions of these terms, see Section 3.1.1.4):
    - H: which corresponds to Horizontal Consistency;
    - V: which corresponds to Vertical Consistency;
    - I: which corresponds to Invocation Consistency;
    - O: which corresponds to Observation Consistency;
    - E: which corresponds to Evolution Consistency;
    - Sy: Syntactic Consistency;
    - Se: Semantic Consistency.

Table 18. Set of UML Consistency rules

1 - State Machine Diagram				
1	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. The initial states of the state machine diagrams U' and U must be identical.	**I**	**Se**	[107, 108]
2	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. Every transition of U' which is already in U has at least the same source states and sink states as it has in U.	**I**	**Se**	[107]
3	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. For each transition t in U' that is already present in U, the guard condition g'(t) in U' must be at least as strong as the guard condition g(t) for t in U: $g'(t) \rightarrow g(t)$.	**I**	**Se**	[107]
4	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. A transition of U in U'	**I**	**Se**	[107]

	cannot therefore receive an additional source state or sink state that is already present in U.			
5	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. A transition added to U' does not receive a source state or a sink state that was already present in U.	**I**	**Se**	[107]
6	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. The set of transitions of U' is a superset of the set of transitions of U.	**I**	**Se**	[107]
7	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. A transition in U' which is already present in U has in U' at most the source states that the transition has in U.	**I**	**Se**	[107]
8	Consider two State Machines U' and U of a class O' and its superclass O, where U' extends the state machine of U by adding states and transitions. For each transition t in U' that is already present in U, U: g(t) → g'(t).	**I**	**Se**	[107]
9	Consider two State Machine U'' and U of a class O'' and its superclass O, where U'' refines the state diagram of U by means of a refinement function h that maps transitions onto transitions and states of U'' and that maps simple states of U'' onto simple states of U. Intuitively, the concept of the refinement function means that if a simple state s of U has been refined to a composite state in U'', then h maps the substates of s in U'' and transitions between these states to s, and h maps the transitions in U'' that are incident to the substates of s into transitions of U that are incident to s. s is a state in U, so s Є U (from the rule: "a simple state s of U","s of h(t')") s' is a refined state in U', so s' Є U' (from the rule: "a state s' in U' ","sink state s' of t' in U' ") Source+(t): t → set of states. Source+(t) = { source of t} U sub-states*{source of t} s' Є Source+(t) For every transition t' in S': for every source state s of h(t'), there exists a state s' in S' such that h(s') = s, and for every sink state s of h(t') there exists a sink state s' of t' in S' such that h(s') = s. Informal rule definition: A transition t'' (of U'') that is refined from t (of U) must have s'' (state sender and receiver of U'') that is refined from s (of U) Є Source+(t).	**E**	**Se**	[107, 108]
10	Consider two State Machine U'' and U of a class O'' and its superclass O, where U'' refines the state diagram of U by means of refinement function h that maps transitions onto transitions and states of U'' and that maps simple states of U'' onto simple states of U. Intuitively, the concept of the refinement function means that if a simple state s of U has been refined to a composite state in U'', then h maps the substates of s in U'' and transitions between these states to s, and h maps the transitions in U'' that are incident to the substates of s into transitions of U that are incident to s. s is a state in U, so s Є U (from the rule: "a simple state s of U","s of h(t')") s' is a refined state in U', so s' Є U' (from the rule: "a state s' in U' ","sink state s' of t' in U' ") Source+(t): t → set of states. Source+(t) = { source of t} U sub-states*{source of t} s' Є Source+(t) For every source state s' of a transition t' in S'', where s'' and t'' do not belong to the same refined state (i.e., h(s'') /= h(t'')), h(s'') is therefore a source state of h(t''), and for every sink state s'' of a transition t'' in S'', where s' and t' do not belong to the same refined state, h(s'') is therefore a sink state of h(t'').	**E**	**Se**	[107]

	Informal rule definition: This rule is about transitions that go between states that are not substates of a complex state that is a refinement of a simple state. The rule says that such transitions have their sources and sinks in U' as refinements of those ones in U.			
11	Using a signal/message on a transition in a state diagram that no object sends, creates structural and syntactic inconsistencies. The information about the sending object can be found in another state machine, in another partition of the same state machine, in a sequence diagram, or any other diagram where one can specify an object can send a signal/message.	**H**	**Sy**	[109]
12	A state machine should be deadlock-free.	**H**	**Se**	[42, 110, 111]
13	A state machine must be deterministic, that is, in every state, only one transition (accounting for the different levels of nested states) should fire on a reception of an event.	**H**	**Sy**	[49, 110, 112]
14	An abstract operation cannot be invoked in a state machine.	**H**	**Sy**	[51]
	2 - Class Diagram, State Machine Diagram and Activity Diagram			
15	There is an inconsistency if a precondition on an operation is in contradiction with a state machine or an activity diagram including a call of that operation.	**H**	**Sy**	[113]
	3 - Sequence Diagram and Use Case Diagram			
16	If a use case is further specified by one or more sequence diagram, then every scenario described in one of those sequence diagrams should match a sequence of steps in that use case's use case description.	**H**	**Se**	[85, 114]
17	If a sequence diagram depicts all the behavior required for successful completion of a use case, it follows that each post-condition specified in the use case description must be achieved by a message (or a set of messages) in the sequence diagram (including referred diagrams) for that use case.	**H**	**Sy**	[115]
18	If a sequence diagram depicts all the behavior required for successful completion of a use case, it follows that results achieved by alternative message flows in the sequence diagram (including referred diagrams) must correspond to post-conditions specified in the use case description.	**H**	**Sy**	[115]
19	Each action specified or implied in a use case description should be detailed in a corresponding message or set of messages in the sequence diagram corresponding to that use case. Depending on the clarity and completeness of the use case description text, the author of the sequence diagram may need to infer some of the operations.	**H**	**Se**	[115]
20	A use case is complemented by a set of sequence diagrams, each sequence diagram representing an alternative scenario of the use case.	**H**	**Se**	[115]
	4 - Activity Diagram (*no rule*)			
	5 - Sequence Diagram			
21	Arguments to messages must represent information that is known to the sender. This includes attribute values of the sender, navigation expressions starting with the sender, constants; This should account for inheritance.	**H**	**Sy**	[115]
22	Variables used in the guard of a message must represent information that is known to the sender. This includes attribute values of the sender, navigation expressions starting with the sender, constants; This should account for inheritance.	**H**	**Sy**	[48, 115-117]
23	If SD2 is a sequence diagram referred to by an Interaction Use (a nested sequence diagram) embedded in sequence diagram SD1, then for every matched pair of message to and from SD2 in SD1, there is a corresponding matched pair of sourceless message and targetless message in SD2.	**H**	**Sy**	[116, 117]
24	In a sequence diagram, if an attribute is assigned the return value of a message, then the types have to be compatible	**H**	**Sy**	[51]
25	Return messages from ExecutionSpecification instances should always be shown.	**H**	**Sy**	[115]

26	Arguments of messages should always be shown.	**H**	**Sy**	[115]
6 – Object Diagram and Collaboration Diagram (*no rule*)				
7 - Object Diagram and Class Diagram				
27	The number of occurrences of a link in an object diagram, an instance of an association in a class diagram, must satisfy the multiplicity constraints specified for the association.	**H**	**Sy**	[118]
8 - Activity Diagram and Class Diagram				
28	A class name that appears in an activity diagram also appears in the class diagram.	**H**	**Sy**	[85, 119-121]
29	An action that appears in an activity diagram must also appear in the class diagram as an operation of a class.	**H**	**Sy**	[114, 119, 121, 122]
30	When an activity diagram specifies an exachange of information, through control or data flow, between two different instances, the class diagram should specify a mechanism (e.g., direct association) so that those instances can indeed communicate.	**H**	**Sy**	[119]
31	Swimlanes (Activity pattern in UML 2.0) in an Activity diagram (represented as className in activity state) must be present as a unique class in a class diagram.	**H**	**Sy**	[120, 122]
9 - State Machine Diagram and Activity Diagram (*no rule*)				
10 – Use Case Diagram and State Machine Diagram (*no rule*)				
11 - Class Diagram and State Machine Diagram				
32	When a state machine specifies the behavior of a class, the actions and activities in the state machine should be operations of the class (in the class diagram) which behavior the state machine specifies. An action or activity can be part of a navigation expression, in which case the navigation expression must legal according to the class diagram and the context (class) of the specified behavior in the state machine.	**H**	**Sy**	[28, 51, 109, 112, 119, 122-125]
33	When a state machine specifies the behavior of a class, any property (i.e., attribute, navigation) in the state machine should be one of the class (in the class diagram) which behavior the state machine specifies. A property can be part of a navigation expression, in which case the navigation expression must legal according to the class diagram and the context (class) of the specified behavior in the state machine.	**H**	**Sy**	[51, 122, 126]
34	When the state machine diagram specifies the behaviour of a class in the class diagram, the class is an active class with a classifierBehavior, then the triggers of the transitions in the state machine diagram are operations of the active class.	**H**	**Sy**	[53, 81, 109, 113, 127, 128]
35	No operation can be called on a state machine or from a state machine if this breaks the visibility rules of the class diagram (public, protected, private).	**H**	**Sy**	[51]
36	When behaviour is triggered from a state machine diagram (e.g., calling an operation, sending a signal) that describes the behaviour of a class, and the triggered behaviour belongs to another class, then the former must have a handle to the latter as specified in the class diagram. Another way of saying this is that the former must have visibility to the latter. A specific case of this situation is when the former class has an association (possibly inherited) to the latter class.	**H**	**Sy**	[51, 124, 125]
37	For a send action, there should be a reception within the classifier of the receiver instance that corresponds to the signal of the send action describing the expected behavior response to the signal.	**H**	**Sy**	[124]

38	For all call or send events specified in the class diagram for a classifier (context) which behavior is specified with a state machine, there should be transitions in that state machine describing the detailed behavior of the events.	**H**	**Sy**	[124]
12 – Protocol State Machine Diagram and Class Diagram (*no rule*)				
13 - Sequence Diagram and Activity Diagram				
39	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, the different diagrams should specify the same scenarios: e.g., same sequence of messages/operations/actions, same branching or repetition conditions.	**H**	**Sy**	[114, 129]
40	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, a flow of interaction between objects in an activity diagram should be a flow of interactions between the same objects in a sequence diagram.	**H**	**Sy**	[119]
41	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, a sequence of messages in the sequence diagram, uninterrupted by control flow structures, must correspond to one activity node in the activity diagram which name is the series of message names of the sequence diagram.	**H**	**Sy**	[129]
42	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, a synchronous message between objects running in different threads of control in the sequence diagram must match a join node for the receiving side (thread) in the corresponding activity diagram, and a fork node for the asynchronous reply.	**H**	**Sy**	[129]
43	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, an asynchronous creation of an active object in the sequence diagram must match a fork node in the corresponding activity diagram: the input action node is matching the calling thread message; two outgoing edges should flow out of the fork node, one for the node matching continuation of execution in the calling thread and one for the node matching the newly created active object.	**H**	**Sy**	[129]
44	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, an asynchronous message sent between two active objects in the sequence diagram should match, in the corresponding activity diagram, a join node on the receiver side and a fork node on the sender side.	**H**	**Sy**	[129]
45	If an activity diagram shows scenarios of an operation and that operation appears in a sequence diagram, a InteractionUse in the sequence diagram must match an activity node in the activity diagram that refers to the activity diagram matching the sequence diagram referred to in the InteractionUse.	**H**	**Sy**	[129]
14 – Use Case Diagram and Object Diagram (*no rule*)				
15 - State Machine Diagram and Communication Diagram				
46	When one specifies an active class, i.e., one that has a state-based behaviour described in a state machine diagram, and an instance of this active class is used in a communication diagram, the messages sent to this object and emitted by this object as specified in the communication diagram must comply to the protocol specified in the state machine diagram.	**H**	**Sy**	[119, 130]
16 - Communication Diagram and Class Diagram				
47	The methods used in a communication diagram must be declared in the class diagram and their declaration, with regard to parameters and types, must be correct.	**H**	**Sy**	[131]
17 - State Machine Diagram and Sequence Diagram				
48	When one specifies an active class, i.e., one that has a state-based behaviour described in a state machine diagram, and an instance of this active class is used in a sequence diagram, the messages sent to this objects and emitted by this object as specified in the sequence diagram must comply (e.g., sequence and types of signals, receivers and emitters of signals) to the protocol specified in the state machine diagram.	**H**	**Sy**	[28, 107, 119, 122, 126, 132-148]

18 – Protocol State Machine Diagram (*no rule*)				
19 - Communication Diagram				
49	If communication diagram A is a specialization of communication diagram B, all the messages present in B have to be included in A.	**E**	**Sy**	[149]
20 - Use Case Diagram and Class Diagram				
50	The noun of the use case's name should equal the name of one class in the class diagram.	**H**	**Sy**	[85, 114, 119, 121, 150]
51	The verb of the use case's name should equal the name of an operation of a class in the class diagram.	**H**	**Sy**	[85, 119, 121, 122, 150]
52	Entity classes correspond to data manipulated by the system as described in a use case description.	**H**	**Sy**	[28]
21 – Object Diagram and Activity Diagram (*no rule*)				
22 - Communication Diagram and Activity Diagram				
53	If an activity diagram specifies a flow of interaction between objects, and those objects (or a subset of those objects) appears in a communication diagram, then the communication diagram should specify the same flow of interaction.	**H**	**Sy**	[119]
23 - Communication Diagram and Class Diagram				
54	Objects involved in a communication diagram should be instances of classes of the class diagram.	**H**	**Sy**	[119, 151-153]
55	In order for objects to exchange messages in a communication diagram, the sending object must have a handle to the receiving object as specified in the class diagram. Another way of saying this is that the sender must have visibility to the receiver. A specific case of this situation is when the sending object's class has an association (possibly inherited) to the receiving object's class.	**H**	**Sy**	[119, 153, 154] [131]
56	Each class in the class diagram appears with at least one instance in at least one communication diagram.	**H**	**Sy**	[48, 119, 151-153]
24 – Communication Diagram and Use Case Diagram (*no rule*)				
25 - Use Case Diagram and Activity Diagram				
57	If a use case U (the including use case) includes use case V (the included use case) in the use case diagram, and both use case U's flows and use case V's flows are specified as activity diagrams, then the activity diagram specifying use case U should contain an action node (likely a CallBehaviorAction node) that refers to the activity diagram specifying use case V.	**H**	**Sy**	[12, 85, 155]
58	Each use case is described by at least one activity diagram.	**H**	**Sy**	[12, 155]
59	Each event (flow of steps) specified or implied in the use case description should be detailed in a corresponding event of the activity diagram. This rule is valid only if the activity diagram further specifies the use case.	**H**	**Sy**	[121]
60	An actor that is associated with a use case will be an activity partition in the activity diagram describing that use case.	**H**	**Sy**	[163]
26 - Use Case Diagram				
61	The use case description of a use case in the use case diagram should contain at least one event (flow of steps). Even if a use case description may be shown/represented in a	**H**	**Sy**	[156]

	separate view respect the use case diagram, we consider that a use case comes with a description.			
62	An event (flow of steps) in a use case description of a use case of the use case diagram has to be of either a basic or an alternate type. Even if a use case description may be shown/represented in a separate view respect the use case diagram, we consider that a use case comes with a description.	**H**	**Sy**	[156]
63	Each use case in the use case diagram must have a use case description specifying event flows. Even if a use case description may be shown/represented in a separate view respect the use case diagram, we consider that a use case comes with a description.	**H**	**Sy**	[28]
64	The name of a use case must include a verb and a noun (for instance "Validate User").	**H**	**Sy**	[150]
	28 - Class Diagram			
65	A class invariant must be satisfied by any non-trivial instantiation of the class diagram (i.e., an instantiation that is not reduced to having no instance of any class).	**H**	**Sy**	[113]
66	The type of a relationship between two classes at a (high) level of abstraction (e.g., plain association, aggregation, composition, generalization) must be the same as the type of a refinement of that relationship at a more concrete (low) level of abstraction. For instance, a plain association at a low level of abstraction being abstracted as an aggregation (a high level of abstraction) denotes an inconsistency.	**V**	**Se**	[157]
67	A class diagram is consistent if it can be instantiated without violating any of the constraints in the diagram (e.g., association end multiplicities).	**H**	**Sy**	[158]
68	Two classes are equivalent if they denote the same set of instances whenever the constraints imposed by the class diagram are satisfied. Determining equivalence of two classes allows their merging, thus reducing the complexity of the diagram.	**H**	**Sy**	[158]
69	A class relationship at a (low) level of abstraction must have an abstraction at a higher level of abstraction, either a class or a relation.	**V**	**Se**	[157, 159]
70	If a class relationship A refines relation B, A must have the same destinations classes as the destination classes of the abstraction of B.	**V**	**Se**	[157]
71	If a class relationship A refines relation B, A must have the same type as B.	**V**	**Se**	[157]
72	The group of relationships between any two high level classes must be identical with the group of relationships between their corresponding low-level classes: ensures the same interactions among low level classes and high-level classes.	**V**	**Se**	[157]
73	If a navigation expression is used in an operation contract, then the expression must be a legal one (according to the syntax of the language and the class diagram).	**H**	**Sy**	[51]
74	This Liskov's substitution principle holds.	**O**	**Se**	[107, 160-162]
75	A class that realizes an interface must declare all the operations in the interface with the same signatures (including parameter direction, default values, concurrency, polymorphic property, query characteristic).	**H**	**Sy**	[51, 163]
76	An abstract operation can only belong to an abstract class.	**H**	**Sy**	[127]
77	If an operation appears in a pre or post condition, then it must have the property isQuery equal to true.	**H**	**Sy**	[51]
78	No (public) method of a class violates, as indicated by its pre and post-conditions, the class invariant of that class.	**H**	**Sy**	[164]
79	In a class, the names of the association ends (on the opposite side of associations from this class) and the names of the attributes (of the class) are different.	**H**	**Sy**	[165]
80	No precondition should violate the class invariant.	**H**	**Sy**	[51]

81	No post-condition should violate the class invariant.	**H**	**Sy**	[51]
82	A class that contains an abstract operation must be abstract.	**H**	**Sy**	[51]
83	There must be no cycle in the directed path of aggregation associations: A class cannot be a part of an aggregation in which it is the whole; A class cannot be a part of an aggregation in which its superclass (or ancestor) is the whole.	**H**	**Sy**	[51]
84	A class cannot be a part of more than one composition - no composite part may be shared by two composite classes.	**H**	**Sy**	[51]
85	Each concrete class, i.e., it is not abstract, must implement all the abstract operations of its superclass(es).	**H**	**Sy**	[51]
86	If an attribute's type is a class, then that class has to be visible to the class containing the attribute. Example: same package or there exists a path in the class diagram that allows the class containing the attribute to have a hold on that type.	**H**	**Sy**	[51]
87	If the return type of an operation is a class, then that class has to be visible to the class containing the operation. Example: same package or there exists a path in the class diagram that allows the class containing the operation to have a hold on that type.	**H**	**Sy**	[51]
88	An operation may not be overridden by a descendant class only if its isLeaf attribute (from metaclass RedefinableElement) is defined accordingly.	**H**	**Sy**	[51]
89	A static operation cannot access an instance attribute (as indicated by its pre and post conditions, for instance).	**H**	**Sy**	[51]
90	A static operation cannot invoke an instance operation (as indicated by its pre and post conditions, for instance).	**H**	**Sy**	[51]
91	If an association end has a private visibility, then the class at this end can only be accessed via the association by the class at the other association end (e.g., as indicated by pre and post conditions of operations of the class at this other end of the association).	**H**	**Sy**	[51]
92	If an association end has a protected visibility, then the class at this end can only be accessed via the association, by the class at the other association end and its descendants (e.g., as indicated by pre and post conditions of operations of the class at this other end of the association).	**H**	**Sy**	[51]
93	The multiplicity range for an attribute must be adhered to by all elements (operation contracts, guard conditions) that access it.	**H**	**Se**	[51]
94	For class A's operations to use another class B, as indicated by contracts in A, there must be a means (e.g., in the form of a path involving associations, generalization and/or dependencies) in the class diagram for A to get a hold on B.	**H**	**Sy**	[51]
95	A class at a low-level of abstraction refines at most one class from a higher-level of abstraction.	**V**	**Sy**	76]
96	A class at a high-level of abstraction is refined by at least one class from a lower-level of abstraction.	**V**	**Sy**	[159]
97	There should not be semantically redundant paths between any two classes in the class diagram graph, unless precisely specified by a constraint (e.g., specified in OCL). For instance, from class A, it may be possible to navigate as self.theA.theB, along with self.theC.theB. The question is then whether the two collections self.theA.theB and self.theC.theB are identical.	**H**	**Sy**	[166]
29 – Interaction Diagram and Class Diagram (*no rule*)				
30 - Use Case Diagram and Interaction Diagram				
98	Each interaction diagram corresponds to a use case in the use case diagram, and each use case in the use case diagram is specified by an interaction diagram.	**H**	**Sy**	[28]
99	If a use case is further specified by one or more interaction diagrams, then every scenario described in one of those interaction diagrams should match a sequence of steps in that use case's use case description.	**H**	**Sy**	[28]

31 – Protocol State Machine Diagram and State Machine Diagram (*no rule*)				
32 - Composite Structure Diagram				
100	A delegation connector connects a port on the boundary of a classifier to a port on the internal structure (or parts) of the classifier. It basically delegates signals or operation calls arriving on the boundary port to the internal port, or those coming from the internal port to the boundary port. Therefore, these ports at the end of the delegation connector must have the same (or compatible) interfaces. If both ports require the interface, then the direction of delegation is from the boundary port to the internal port. If both ports provide the interface, then the direction of delegation is from the internal port to the boundary port.	**H**	**Sy**	[167]
101	If an assembly connector exists between two ports, one of the ports (the source) must be a required port and the other (the destination) must be a provided port. This rule describes the opposite case of delegation, where both ports at the end of an assembly connector have conjugate interfaces (one port requires an interface; the other provides the same interface). What matters is that the two ports must either have the same interface but one of them is marked as isConjugate, while the other is not, or they should have conjugate interfaces (i.e., ones that have the same operations or signal receptions that are annotated as provided on one interface and required on the other).	**H**	**Sy**	[167]
102	If a connector is typed with an association, the direction of the association must conform to the direction of the connector as derived from the direction of the ports at its ends (association navigable from class A to class B if the connector between A and B indicates that A requires services that B provides). Given that the direction of associations and connectors (however, it is calculated) could be encoded using the order of their 'memberEnd' and 'end' collection, respectively, the rule is basically saying that such direction should be the same in both cases, if a connector is typed by an association.	**H**	**Sy**	[167]
103	If a link outgoing from a port is statically typed with an association, then the association must be navigable to an interface which is part of the set of interfaces and the type indicated by the association must belong to the set of transported interfaces for that link.	**H**	**Sy**	[167]
104	If the link originates from a component, then the link must be statically typed with an association, and the type of the entity at the other end of the link must be compatible with (i.e. be equal or a subtype of) the type at the corresponding end of the association.	**H**	**Sy**	[167]
105	The set of transported interfaces by a link should not be empty.	**H**	**Sy**	[167]
106	If several non-typed connectors start from one port, then the sets of interfaces transported by each of these connectors have to be pair wise disjoint.	**H**	**Sy**	[167]
107	The union of the sets of interfaces transported by each of the connectors originating from a port P must be equal to the set of interfaces provided/required by P.	**H**	**Sy**	[167]
33 - Sequence Diagram and Class Diagram				
108	The type of a lifeline (type of the connectable element of the lifeline) in a sequence diagram must not be an interface nor an abstract class.	**H**	**Sy**	[48, 51]
109	In case a message in a sequence diagram is referring to an operation, that operation must not be abstract.	**H**	**Sy**	[48, 51]
110	If a message in a sequence diagram refers to an operation, through the signature of the message, then that operation must belong, as per the class diagram, to the class that types the target lifeline of the message.	**H**	**Sy**	[14, 28, 48, 51, 52, 81, 85, 115-117, 119, 122, 123, 127, 128, 135,

				141, 153, 156, 164, 168-180]
111	Interactions between objects in a sequence diagram, specifically the numbers of types of interacting objects, must comply with the multiplicity restrictions specified by the class diagram (e.g., association end multiplicities).	**H**	**Sy**	[50, 116, 117, 160, 174]
112	In order for objects to exchange messages in a sequence diagram, the sending object must have a handle to the receiving object as specified in the class diagram. Another way of saying this is that the sender must have visibility to the receiver. A specific case of this situation is when the sending object's class has an association (possibly inherited) to the receiving object's class.	**H**	**Sy**	[14, 48, 50, 51, 81, 115-117, 119, 122-124, 128, 141, 145, 153, 160, 164, 168-170, 172, 174-178, 180, 181]
113	The behavioral semantics of a composition or aggregation association in the class diagram must be inferred in sequence diagrams. For instance, in a whole-part (composition) relation, the part should not outlive the whole.	**H**	**Sy**	[50, 115]
114	Each public method in a class diagram triggers a message in at least one sequence diagram.	**H**	**Sy**	[48, 126, 128]
115	Each class in the class diagram must be instantiated in a sequence diagram.	**H**	**Sy**	[115, 116, 119, 122, 123, 152, 153, 173, 181]
116	No operation can be used in a message of a sequence diagram if this breaks the visibility rules of the class diagram (public, protected, private).	**H**	**Sy**	[51]
34 - Sequence Diagram and Communication Diagram (*no rule*)				
35 - Class Diagram, State Machine Diagram and Communication Diagram (*no rule*)				
36 - Class Diagram, Sequence Diagram and Use Case Diagram (*no rule*)				

Appendix B

In order to develop a survey that would adequately gather the information needed to answer the four research questions presented in Section 4.2, we developed a questionnaire consisting of four Sections with a total of 21 questions (Q1 to Q21). Figure 36 describes the flow of questions in the questionnaire.

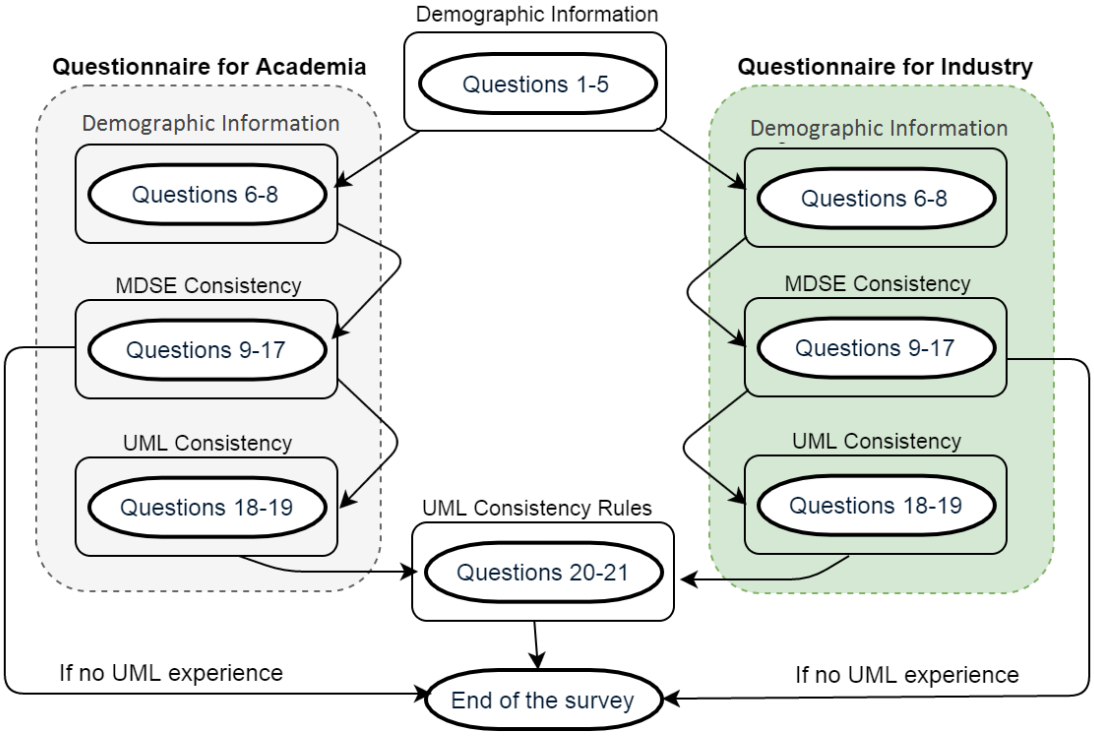

Figure 36. Series of questions for respondents with industry/academic experience

Table 19. Survey questionnaire

Section	Research questions	Type of Answers			
		Single from a list	Multiple from a list	Likert scale	Other text
1. Demographic Information (answering RQ1)	**Q1.** In what country do you work?	x			
	Q2. What is your highest level of education?	x			x
	Q3. Where are you currently working?	x			
	Q4. How long have you been working in the context of Software Engineering? (Academia/Industry)	x			
	Q5. How long have you been working in the context of Model Driven Software Engineering? (Academia/Industry)	x			

Section	Research questions	Type of Answers			
		Single from a list	Multiple from a list	Likert scale	Other text
	Q6. What is your current position? (Academia/Industry)		x		x
	Q7. What is the number of researchers in your research group? (Academia) - What is the number of employees in your company? (Industry)	x			
	Q8. What is the number of researchers in your research group who are working in the context of Model Driven Software Engineering? (Academia) - What is the number of employees in your company who are working in the context of Model Driven Software Engineering? (Industry)?	x			
2. MDSE Consistency (answering RQ2)	**Q9.** How often have you detected/studied/fixed/handled model consistency problems while developing software or system?			x	
	Q10. How important is the topic of model consistency in your institution? (Academia) How important is the topic of model consistency in your company? (Industry)			x	
	Q11. In what of the following Model Driven Software Engineering activities are models consistency constraints used? (see the activities below in this appendix)		x		x
	Q12. Are you familiar with the following dimensions and types of Consistency? (see Section 3.1.1.4 for dimensions definitions)			x	
	Q13. How often have you experienced consistency issues that involve the following dimensions and types of consistency? (see Section 3.1.1.4 for dimensions definitions)			x	
	Q14. How are models consistency constraints usually specified in your institution? (Academia) - How are models consistency constraints usually specified in your company? (Industry)		x		x
	Q15. What tools are used to check models consistency in your institution? (Academia) - What tools are used to check models consistency in your company? (Industry)		x		x
	Q16. What tool(s) is/are used in your institution to represent models? (Academia) - What tool(s) is/are used in your company to represent models? (Industry)		x		x
	Q17. What modeling language(s) is/are used in your institution? (Academia) – What modeling language(s) is/are used in your company? (Industry)		x		x
3. UML Consistency (answering RQ3)	**Q18.** Which UML diagrams are mostly involved in consistency issues in your institution? (Academia) - Which UML diagrams are mostly involved in consistency issues in your company? (Industry)		x		x
	Q19. How do you study, detect, and handle UML models inconsistencies?		x		x
4. UML Consistency Rules (answering RQ4)	**Q20.** Do you understand the rule?	x			
	Q21. Do you think the rule is relevant to be enforced?			x	

The 21 questions can be found in Table 19. We used four different types of answers, as indicated in the Table:

- **Single from a list**: in this type of question, the questionnaire displays a list of predefined answers which can be selected only one time.
- **Multiple from a list**: in this type of question, the questionnaire allows the respondents to select one or more options from a list of predefined answers.
- **Likert scale**: in this type of question, the questionnaire allows the respondents to give a rating on a n-pointing agreement scale (likert scale). The object of the scale is to measure a respondent's opinion or attitude towards any given subject.
- **Other text**: in this type of question, the questionnaire allows respondents to insert free-form text as answer of a specific question.

We used the following Likert scales:

- 4-point Likert scale for Q9: often, sometimes, rarely, never.
- 3-point Likert scale for Q10: very important, partially important, not important.
- 4-point Likert scale for Q12: very familiar, familiar, partially familiar, not familiar.
- 4-point Likert-scale (plus N/A) for Q13: often, sometimes, rarely, never.
- 3-point Likert scale for Q21: always, sometimes, never.

In order to answer Q11 in Section 4.2.2.2.3, we present the nine definitions of MDSE activities we identified in an earlier [17]:

1. The verification of consistency is the process of detecting inconsistencies between (or in a singular) view/diagram(s) during software development. This definition is in line with the IEEE definition of verification [47].
2. Consistency management includes the detection of inconsistencies in a software model, the diagnosis of the significance of inconsistencies, and the handling of detected inconsistencies [16].
3. Model Refinement [48] is a semantics-preserving transformation applied to a model, and which produces the same kind of model, e.g., refining a Platform Independent Model (PIM) into a new PIM;
4. Model Transformation [48] (also called mapping) generates a new kind of model, e.g., transforming a PIM into a Platform Specific Model (PSM), or a PSM into executable code. These transformations generally add detail to the model.

5. Safety consistency, from the point of view of automated analysis, implies that there are no conflicting requirements and unintentional (internal) non-determinism in a view/diagram of the model [49].
6. Security consistency, from the point of view of automated analysis, is defined as the consistency of bounded values in non-abstract elements of a view/diagram of the model [50].
7. Impact Analysis is defined as the process of identifying the potential consequences (side-effects) of a change to the model, and estimating what needs to be modified to accomplish a change [51].
8. Model formulation describes the formal process used to specify/develop a model [52, 53].
9. Model comprehension is the process employed to understand that a model is much more than a set of annotated boxes and lines as it reflects the meaning of a software, that is, the model has semantics. The semantics of the model is expressed through its meta-model and context/domain dependent consistency rules and in the case of UML, through both the so-called well-formedness rules. Those consistency rules may be described in plain language and/or using the Object Constraint Language (OCL) [28].

The previous list of activities was obtained from the primary studies of a systematic study carried out in Section 3.2, but it is not meant to exhaustively represent all the MDSE activities where model consistency constraints can be used.

Appendix C

Dependencies between diagrams can become so intricate that it is sometimes even possible to synthesize one diagram from other ones [182, 183]. The term "synthesis" has traditionally been used to describe the automatic construction of a program or a behavioral model from formal requirements [46]. In this Appendix, the accepted basic definition of synthesis is that of generating a UML diagram of one type from one or more of the remaining 13 types [182, 183]. Support for synthesizing one UML diagram from other diagrams can provide the designer with significant help, thus speeding up the design process, decreasing the risk of errors, and guaranteeing consistency among the various diagrams in the model [184].

In this appendix, we report the results of a Systematic Mapping Study (SMS) to carry out the current state-of-the-art in UML diagram synthesis. The ultimate goal of this work focuses on the general notion of consistency between UML diagrams that describe one software system [11, 18, 23, 24, 26]. Indeed, while synthesizing one diagram from others, a synthesis technique enforces some consistency between the source diagram(s) and the generated diagram. However, synthesis techniques may or may not explicitly specify a set of consistency rules. A further objective of this appendix is therefore to discover the consistency rules that result from UML diagram synthesis techniques. To achieve these goals, we have performed a SMS [185], following the guidelines of Kitchenham and Charters [31] and Petersen et al. [32].

The rest of this appendix is structured as follows: the SMS protocol adopted in this work [31] is illustrated in the next Section (Section I). We then present the results in Section II, and we discuss and summarize them in Section III. This appendix ends with a discussion of our conclusions (Section IV).

I. SMS – Planning and execution

In this Section, we present the main components of the planning of our SMS [31], which are: the specification of the research questions that the study aims to answer (see below); the strategy followed to find existing research papers on the topic (Section I.a); the procedures followed to select (or reject) papers for further investigation (Section I.b); and the procedures that are used to

extract data from the selected papers in order to answer the research questions (Section I.c). We then discuss the execution of the SMS (Section I.d). The underlying motivation for the research questions was to determine the current state-of-the-art as regards to UML synthesis techniques and collect the corresponding set of UML diagram consistency rules. We therefore considered posing the following seven research questions (RQ#) along with their main motivations (MM#): **RQ1)** Which versions of UML are used by researchers in their synthesis techniques? **MM1**: To understand which versions of UML are used in UML synthesis techniques and whether there are any differences between UML versions (possibly owing to differences between metamodel versions). **RQ2**) Which types of UML diagrams are used as the source and target of synthesis techniques? **MM2**: To discover the UML diagrams that research has focused on in order to reveal the UML diagrams that are most frequently considered in this topic, and perhaps those that lack attention. **RQ3)** Which technologies are used to implement synthesis techniques? **MM3**: To find the technologies that are used to implement UML diagram synthesis. From the outset, it is possible to think of, for instance, model transformation technologies and ad hoc transformation algorithms. **RQ4)** Are the UML synthesis techniques automatic, semi-automatic, or manual? **MM4**: To discover whether the UML diagram synthesis techniques are automated or require human input. **RQ5)** What types of research papers report on UML synthesis techniques? **MM5**: To determine whether the field is generally more applied or consists of more basic research, as evidenced when classifying papers according to a well-known taxonomy [186]. **RQ6)** What are the UML consistency rules involved in the UML synthesis techniques? **MM6**: To find the UML consistency rules involved in the context of UML synthesis and to assess the state of the field. **RQ7**) Which types of UML consistency problems are tackled in the existing UML synthesis techniques? **MM7**: To find the types of consistency problems [40] tackled in the consistency rules involved in UML synthesis techniques: 1) horizontal, vertical and evolution consistency; 2) syntactic and semantic consistency; 3) observation and invocation consistency; and assess the state of the field as regards that dimension of UML diagram consistency.

a. Search strategy

Conducting a search for research papers using search engines requires the identification of a search string, and the specification of the parts of research papers in which the search string will be sought

(the search fields). We identified our search string by following the procedure of Brereton et al. [38], which is composed of five steps: 1) Define the major terms; 2) Identify alternative spellings, synonyms or related terms for major terms; 3) Check the keywords in any relevant paper that may already be available to consolidate the list of terms; 4) Use the Boolean OR to incorporate alternative spellings, synonyms or related terms; 5) Use the Boolean AND to link the major terms. In our case, the major search terms were "UML" and "Synthesis" and the alternative spellings, synonyms or terms related to the major terms were: 1) UML (uml OR unified modeling language OR unified modelling language); 2) Synthesis (synthesis diagram). When selecting the search string, we considered various alternatives with the following objective in mind: we were interested in collecting synthesis techniques, or in other words, in identifying synthesis algorithms in order to precisely understand them and identify the consistency rules they (implicitly) enforce. Other search strings were experimented with, but are not discussed in this manuscript due to space limitations. Of all the alternative search strings in the set, we selected the following one as it allowed us to retrieve, the largest number of papers focusing on UML diagram synthesis:

((uml OR unified modeling language OR unified modelling language) AND (Synthesis Diagram)).

The search was limited to electronic papers and in particular those that were published in peer-reviewed journals, international conferences and workshops only in English language. We did not establish any restriction with regard to the publication year, except that we stopped with papers published in January 2018 at the latest. We used the abovementioned search string with the following seven search engines: Scopus, Google Scholar, CiteSeer, IEEE Digital Library, Science Direct, ACM Digital Library, and Inspec database. The searches were limited to title, keywords and abstract.

b. Selection procedure

Since a systematic, automated search in databases based on a search string may return papers that cannot be used to answer a set of research questions of interest in an SMS (e.g., in our case, a paper that mentions the synthesis of test cases from a UML sequence diagram), the set of papers collected from the search needs to be trimmed in a systematic manner. This is typically done by relying on so-called inclusion and exclusion criteria. Inclusion and exclusion criteria are based on the research

questions and are piloted to ensure that they can be reliably interpreted and that they classify studies correctly [31]. In this Section, we discuss the inclusion and exclusion criteria we used. According to standard terminology in empirical software engineering, the set of research papers finally retained is referred to as primary studies [31]. The inclusion criteria were: 1) Electronic papers focusing on UML diagram synthesis; 2) Electronic papers written in English; 3) Electronic papers published in peer-reviewed journals, international conferences and workshops; 4) Electronic papers published until January 2018; 5) Electronic papers which proposed UML synthesis techniques. The exclusion criteria were: 1) Electronic papers not focusing on UML diagram synthesis; 2) Electronic papers which were extended abstracts; 3) Duplicated electronic papers (e.g., returned by different search engines); 4) Electronic papers which discussed synthesis techniques between UML diagrams and other non-UML sources of data, such as requirements or source code.

c. Data extraction strategy

We extracted the data from the primary studies according to a number of criteria, which were directly derived from the seven research questions detailed in Section I. Using each data extraction criterion required we read the full-text of each primary study. Once extracted, we recorded data on a spreadsheet that represented our data form. The following information was collected from each primary study:

1) Search engines: where the paper was found (see Section I.a);

2) Inclusion and exclusion criteria used (see Section I.b);

3) Data related to research questions (see Section I):

a) Which UML version was used;

b) What the UML synthesis techniques are;

c) The UML diagrams that were involved as input and output in synthesis techniques: Class (CD), Collaboration (COD), Use Case (UCD), Communication (COMD), State Machine (SMD), Sequence (SD), Protocol State Machine (PSMD), Object (OD), Interaction (ID), Activity (AD), Composite Structure (CSD), Timing (TD), Interaction Overview (IOD), and Deployment (DD) Diagram;

d) Tool support: ‘Automatic’ signifies that the UML synthesis techniques were fully-automated, i.e., supported by an implemented and working tool; 'Semi-automatic' means that the synthesis techniques were partially automated (for instance when the synthesis of a UML model needs a user’s support for the process to be completed); ‘Manual’ means that the UML synthesis techniques were not supported by any implemented and automatic tool;

e) Type of research used in the primary study, for which we used the following classification [186]: *i)* An *Evaluation Research* (ER) paper investigates techniques that are implemented in practice, and reports on their evaluation. Such a paper shows how the technique is implemented in practice (solution implementation) and what the consequences of the implementation are in terms of benefits and drawbacks (implementation evaluation). *ii)* A *Proposal of Solution* (PS) paper proposes a solution to a problem and argues for its relevance, without a full-blown validation. *iii)* A *Validation Research* (VR) paper investigates the properties of a solution that has not yet been implemented in practice. *iv)* A *Philosophical Paper* (PP) sketches a new way of looking at things, a new conceptual framework, etc. *v)* An *Opinion Paper* (OP) contains the author’s opinion about what is wrong or good about something, how something should be done, etc. *vi)* A *Personal Experience Paper* (PEP) places more emphasis on what than on why. The experience may concern one project or more, but it must be the author’s personal experience;

f) The consistency rules implicitly enforced by the UML synthesis technique(s) (see Table 22 of Section II.f).

g) UML consistency dimensions, as discussed in the [40]: *i) Horizontal, Vertical and Evolution Consistency*: Horizontal consistency, also called intra-model consistency, refers to consistency between different diagrams at the same level of abstraction (e.g., class and sequence diagrams during analysis) in a given version of a model [15]; Vertical Inconsistency, also called inter-model consistency, refers to consistency between diagrams at different levels of abstraction (e.g., analysis vs. design) in a given version of a model [41]; Evolution consistency refers to consistency between diagrams of different versions of a model in the process of evolution [15]. *ii) Syntactic versus Semantic consistency*: Syntactic consistency ensures that the elements involved in the synthesis of two different diagrams conforms in the same way to the abstract syntax specified by the meta-model. Semantic consistency requires diagrams to be semantically compatible [41], and is not restricted to behavioral diagrams. Semantic consistency applies at

one level of abstraction (with horizontal consistency), at different levels of abstraction (vertical consistency), and during model evolution (evolution consistency) [30]. *iii) Observation versus Invocation consistency*: Observation consistency requires an instance of a subclass to behave like an instance of its superclass, when viewed according to the superclass description [161]." Invocation consistency requires an instance of a subclass of a parent class to be used wherever an instance of the parent is required [42].

d. Execution

The execution for this SMS with the seven search engines began in September 2017 and was completed in January 2018. In this Section, we present the use of the search string with these engines and the selection of primary studies according to the inclusion/exclusion criteria previously described. In order to document the review process with sufficient details [31], we describe the multi-phase process of the four sub-phases employed: SP1) First sub-phase: the search string was used to search in the seven search engines, as mentioned earlier. SP2) Second sub-phase: we deleted duplicates. SP3) Third sub-phase: we obtained an initial set of studies by reading the title, abstract and keywords of all the papers obtained after SP2 while enforcing the inclusion and exclusion criteria. When this was not sufficient to decide whether or not to include or exclude a paper, we checked the full-text of the paper. SP4) Fourth sub-phase: the papers from SP3 were read in their entirety while reapplying the exclusion criteria. This resulted in the final set of 14 primary studies.

Table 20 breaks down the number of papers we have found into sub-phases. We eventually collected 14 primary studies for further analysis [182-184, 187-197]. These were then analyzed according to the criteria detailed in Section I.b and data was recorded in a spreadsheet.

Table 20. Summary of Primary Study Selection

Sub phase	IEEE	Inspec	ACM	Google Scholar	Cite Seer	Science Direct	Scopus	Tot.
SP1	33	98	18	14	25	6	81	275
SP2	20	20	7	4	13	3	41	108
SP3	8	7	6	0	4	0	3	28
SP4	4	3	3	0	3	0	1	14

II. Systematic mapping study – results

A quantitative summary of the results for research questions RQ1-RQ5 and RQ7 is presented in Table 21. The results of RQ6 are presented in Table 22 of Section II.f. More details are provided in Sections II.a to II.0. We then discuss some of the main findings in Section III.

Table 21. SMS results at a glance

Research question	Possible Answer		Result		Reference
			# Papers	Percentage	
RQ1: UML versions	UML 1.3		7	50%	[183, 184, 190, 191, 193, 195, 196]
	UML 1.4		2	14%	[182, 192]
	UML 2.0		2	14%	[189, 194]
	UML 2.1.		1	7%	[187]
	UML 2.1.2		1	7%	[188]
	UML 2.2		1	7%	[197]
RQ2: UML diagrams	Diagram to start the synthesis	IOD	1	6%	[187]
		SMD	2	13%	[182]
		AD	1	6%	[194]
		SD	11	69%	[182-184, 188-190, 192, 193, 195-197]
		COD	1	6%	[191]
	Diagram to synthesize	CD	3	19%	[182, 183, 193]
		SMD	11	69%	[182, 184, 187, 189-192, 194-197]
		AD	1	6%	[188]
		SD	1	6%	[182]
RQ3: UML Synthesis techniques	IOD → SMD		1	6%	[187]
	SD → SMD		8	50%	[192, 195, 196] [182, 184, 189, 190, 197]
	SD → CD		2	13%	[183, 193]
	AD → SMD		1	6%	[194]
	SD → AD		1	6%	[188]
	COD → SMD		1	6%	[191]
	SMD → SD		1	6%	[182]
	SMD → CD		1	6%	[182]
RQ4: Research methods	Evaluation Research		1	7%	[183]
	Validation Research		1	7%	[182]
	Proposal of Solution		12	86%	[184, 187-197]
RQ5: Support	Automatic		12	86%	[182-184, 187, 190-197]
	Manual		2	14%	[188, 189]
RQ7: Type of rules	Semantic Consistency		41	87%	see Table 22 for details
	Semantic Consistency		6	13%	
	Horizontal Consistency		47	100%	

a. UML version (RQ1)

Although UML 2.x has an improved semantics in comparison to UML 1.x, which could help devise diagram synthesis techniques, we did not find more synthesis techniques for UML 2.x (five papers, i.e., 35%) than for UML 1.x (9 papers, i.e., 64%) during the period ranging between 1999—2016 (period of publication of the 14 primary studies): Table 2. It is interesting to note that we found more papers on the synthesis of UML 1.x diagrams than on UML 2.x in a shorter period of time (four years in the case of UML 1.x versus 13 years in the case of UML 2.x). No UML version was reported in a paper [192], but as it was published in 2003 which is the same year of publication as the UML version 2.0 [9], we classified it as a UML version 1.4 paper assuming the work was conducted prior to publication of the paper, i.e., prior to the publication of the UML 2.0.

b. UML diagrams (RQ2)

Figure 37 summarizes the number of times each diagram of some synthesis has been found to be the source or target.

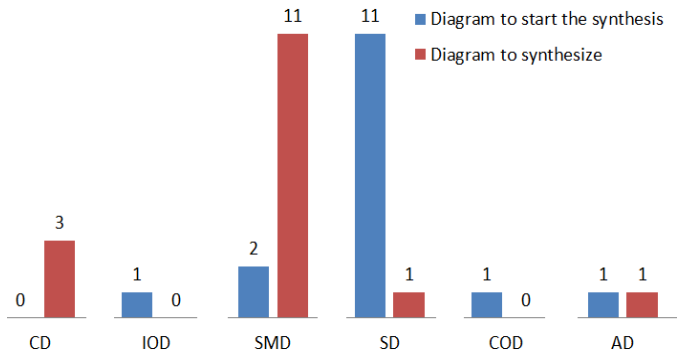

Figure 37. UML diagrams in Synthesis Techniques (see accompanying text for a detailed description of the semantics of the figure).

The sequence diagram (SD) is by far the diagram type that is most frequently used for the synthesis of a diagram of another type (68%, 11 synthesis techniques). Conversely, the state machine diagram (SMD) is the most synthesized diagram type (68%, 11 techniques), followed by the class diagram (CD) (19%, three techniques). Overall, only half of the UML diagram types are involved in some sort of synthesis technique, and it is not entirely surprising that the diagram types involved in synthesis techniques are also identified as the most frequently used diagrams [44].

c. Technologies for synthesis techniques (RQ3)

In the 14 primary studies, we identified 16 different synthesis techniques, each of which involved one input diagram and one output diagram, which can be classified into eight different types of synthesis techniques: Figure 38 and Table 23.

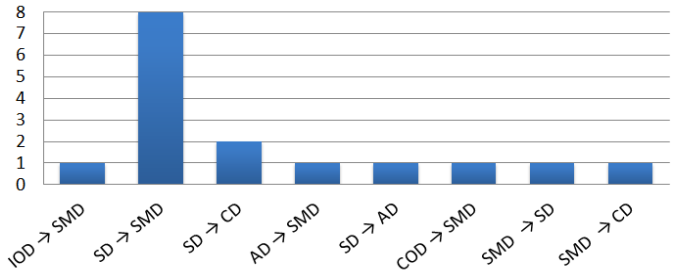

Figure 38. From → To Synthesis Techniques (see accompanying text for a detailed description of the semantics of the figure).

The most frequently described synthesis technique type is sequence diagram to state machine diagram (SD→SMD) for a total of eight techniques (50%, out of 16 synthesis techniques). Each of the 16 synthesis techniques is summarized below, and we also discuss the technologies they rely on in order to perform the synthesis. Due to space limit, in this Section we only describe 16 UML synthesis techniques considered in this Appendix. Whittle and Jayaraman [187] presented a technique to automatically synthesize hierarchical state machine for state dependent objects from Extended Interaction Overview Diagrams (EIODs), that is a three-layer structure, with a precise semantics, where the top layer defines use cases, the second layer defines scenarios, and the bottom layer defines sequence diagrams. We classify this synthesis technique as a mapping since, even though the authors precisely describe the mapping between elements of the EIOD to elements of hierarchical state machines, they did not provide a detailed algorithm or did not rely on some other technology (e.g., model transformation) to perform the synthesis. So far, research mostly focused on technical issues such as how to best derive SMD from SD and how to best ensure the correctness of this synthesis: half (five) of the UML synthesis techniques address this issue. The Minimally Adequate Synthesizer (MAS) [184] infers a SMD from a set of SDs, using an engineer-guided grammatical inference technique: the state machine is seen as describing an unknown grammar to be inferred, where the sequence diagrams are sentences. We classify this synthesis technique as grammatical inference. Ziadi and colleagues [189], very similarly to Whittle and Jayaraman [187],

introduce an algebraic framework for such a synthesis: the authors define an algebraic representation of sequence diagrams, an algebraic representation of state machine diagrams, and then a precise mapping between the two. We classify this technique as algorithmic since the authors provide a detailed algorithm for the mapping. Schumann [190] synthesized state machines from sequence diagrams, where messages are annotated with OCL pre and post conditions, based on an earlier work of the author [195], which is itself based on an earlier work of Whittle [187] that does not account for interaction fragments. We classify this technique as algorithmic since the earlier paper provides a detailed algorithm for the synthesis. The Fujaba project [192] also includes this type of synthesis. It is based on grammatical inference as in MAS [184], and that we therefore also classify as grammatical inference. Selonen and colleagues also rely on a grammatical inference algorithm to synthesize a series of state machines from sequence diagrams [182].

Khriss and colleagues [191] proposed an incremental UML synthesis technique to generate SMDs of all the objects involved from a set of CODs. The technique employed to perform the synthesis is informally described and we therefore classify this work as informal description. Whittle and Schumann in both [195,196] present an algorithm which supports the design process by generating state machine diagrams (SMD) automatically from scenarios (SD). Krka and Medevidovic [197] propose component-aware triggered scenarios (caTS) operationalized through a synthesis algorithm which represents a step forward to triggered scenario languages. On the other hand, Eshuis and Gorp [194] present an automated synthesis approach for generating state machine diagrams from activity diagrams.

These different synthesis approaches [182, 184, 187, 189-192] are very similar in the sense that they recognize object interactions, provided in various, though similar forms (Extended Interaction Overview Diagrams, Sequence Diagrams, Collaboration Diagrams) as specifying legal sequences of messages objects can receive and from which the complete set of legal sequences of messages that objects can receive can be constructed (i.e., the state machine diagrams). Selonen and colleagues presented the synthesis of UML CDs from SDs in 2000 [193], which they extended in 2001 [183] and summarized, along with other synthesis techniques (see below) in 2003 [182]. The authors informally discussed how elements of sequence diagrams can be mapped to elements of a class diagram. We therefore classify this technique as informal description. Kang and colleagues [188] synthesized ADs from SDs thereby representing the flow of messages in sequence diagrams as activity diagrams. The mapping rules from SD to AD are precisely discussed and the overall

synthesis is described with an algorithm. We therefore classify the technique as algorithmic. We already discussed two of the synthesis techniques that Selonon and colleagues summarized [182]. They also discuss the synthesis of CD from SMD, as well as the synthesis of SMD from SD following the work of Systä [198]. These are only summaries, which we classify as informal descriptions.

d. Research papers (RQ4)

As reported in Table 3, the primary studies can be classified as ER (Evaluation Research), VR (Validation Research), and PS (Proposal of Solution), contributing 7%, 7% and 86% of the total, respectively. The vast majority of the primary studies therefore propose a complete technical solution for the synthesis of UML diagrams. One paper was categorized as an evaluation research paper [183] and one other was categorized as a validation research paper [188].

e. Type of support (RQ5)

As described in Table 3, the synthesis techniques we collected are usually supported by a tool (86%, 12 papers). Only one of them [187] is integrated with its UCSIM UML CASE tool, a vendor-independent implementation that the authors intended to integrate with IBM Rational Software Modeler. The synthesis technique of Schumann [190] was implemented in Java and the author mentioned the intention to integrate the technique into the commercial UML tool Magic Draw. We were, however, unable to find more recent publications by this author since the original publication of the work in 2008 confirming that this actually happened. The synthesis technique of Selonen and colleagues [182, 183, 193] was implemented in an industrial software development environment by the same authors: the Nokia TED. Whittle and Schumann in both [195] and [196] present a prototype of their synthesis algorithm which has been implemented in Java. Maier and Zndorf [192] implemented their synthesis technique in the Fujaba environment, which is a free open source UML CASE tool environment, to provide roundtrip engineering support. Ziadi and colleagues [189] developed a prototype of their synthesis technique in Java. Mäkinen, and Systä [184] implemented their synthesis technique in their own prototype tool, whereas Krka and Medvidovic [197] illustrate a synthesis algorithm as a specification of their component-aware

triggered scenario (caTS). Kang and colleagues [188] mentioned that they were planning to improve their synthesis technique by automating its manual steps. Khriss and colleagues similarly deferred complete automated support (in their case adding editors for the UML diagram types involved in their synthesis technique) to future work [191]. We were unable to find more recent publications by these authors discussing these aspects since the initial publications (2010 and 1998, respectively).

f. UML consistency rules (RQ6)

Since the different UML diagrams that are constructed during a specific software development specify different aspects of one, and only one system, these diagrams must be consistent with one another. This is true regardless of whether the diagrams are generated by hand (using a CASE tool for instance) or synthesized from other diagrams. The UML diagram synthesis techniques we have collected, therefore, enforce some kind of consistency between the input diagrams of the synthesis and the diagrams being synthesized. We finally collected a set of 47 UML consistency rules (from the different synthesis techniques–the primary studies) that are presented in Table 22. No consistency rule was identified in some of the primary studies [192, 194-197]. Paper [193] presented the same nine rules as in paper [183] but with a new characteristic that can be used to synthesize class diagrams with operation description annotations.

Table 22 lists the 47 horizontal UML consistency rules. The first column "S.T." (Synthesis Technique) presents the diagrams involved in the technique (for example SD to SMD, from a Sequence Diagram, a State Machine Diagram is synthesized). The third column indicates whether this is a Syntactic (Sy) or Semantic (Se) rule. The fourth column indicates the reference (Ref.) of the primary study paper from where a given rule was collected.

Table 22. Summary of UML Consistency Rules

S.T.	UML Consistency rules		Ref.
SD to	A reception in the SD becomes an event in the SMD and emissions become actions.	Sy	[189]
	For a transition associated with a reception, the action part will be empty, and for transitions associated with actions, the event part will be empty.	Sy	

S.T.	UML Consistency rules		Ref.
SMD	SMDs generated from SDs will be sequences of states, and will contain a single junction state that corresponds to the state reached when all events situated on an object lifeline have been executed.	Sy	
	When an object does not participate in an interaction, the projection of an SD on this object's lifeline is the empty word	Sy	
	For a single life line within a SD, incoming messages are interpreted as events attached to certain transitions and outgoing messages as do-actions of certain states.	Se	[184, 196]
	Following the lifeline of a given object, reuse already existing transitions and states in the corresponding SMD for as long as possible. Otherwise new transitions and states are created.	Se	
	For each such possible start element, compute the number of subsequent messages that match subsequent SMD elements without the need to create new elements.	Sy	
	Map items in the message trace in SD to transitions and states in an SMD.	Sy	[182]
	Messages sent in SD are regarded as primitive actions associated with states in SMD.	Sy	
	Each message received in SD is mapped in a SMD transition.	Sy	
	A synthesized SMD is deterministic, i.e., there cannot be two similarly labeled leaving transitions in any particular state, unless their guards are mutually exclusive. Two applicable transitions cannot be satisfied simultaneously.	Sy	
	A completion transition and a labeled transition cannot leave the same state unless their guards differ.	Sy	
SD to CD	Generate a class for each classifier role with a distinctive class name. Transfer stereotypes, such as «actor» and «active» into the respective class.	Sy	[183]
	For each message, if an association between the base classes of the sending and receiving classifier roles does not exist, generate the corresponding association ends and an association representing the communication connection for the message in question, and link them together.	Se	
	For each class receiving a message, if an operation with the same name does not already exist, generate a new operation for the class. Mark the navigability of the association.	Sy	
	If there are arguments attached to the message, generate new parameters for the operation with corresponding basic types and type expressions. Add this operation to the corresponding class if it does not already have an operation with the same name and signature.	Sy	
	If the message is marked with a stereotype «signal» and a signal with the same name does not exist, generate a new signal. Associate the corresponding class with this signal.	Sy	
	If the message is marked with a stereotype «become», the sending and receiving classifier roles are equated (this rule assumes that an object cannot dynamically change its type).	Sy	
	If the message has an object that appears in the SD as an argument, we add an association between the corresponding classes of the sending object and argument object, if one does not already exist and the classes are not the same.	Sy	

S.T.	UML Consistency rules		Ref.
	If the message is a return message, containing only a return value of some type, we conclude the return type of the operation of the preceding message.	Sy	
	When SDs contain contradictory information, for example an active and a passive object of the same class. In this case, the result of the transformation operation should be undefined.	Se	
IOD to SMD	Normal edges in an EIOD are interpreted as a weak sequential composition but taking into account flow final and final nodes.	Sy	[187]
	Parallel fork/join edges lead to orthogonal regions in the SMDs generated. Since fork/join edges are well nested, there is no ambiguity as regards deciding where the orthogonal regions should be placed.	Sy	
	For preempting and suspending edges, composite states are introduced into the SMDs generated.	Sy	
	Negative edges result in orthogonal regions in the SMDs generated. Negation is handled by monitoring for the negative events and transitioning to a special error state if they occur. The negative events being monitored are placed in an orthogonal region so that, if the sequence of negative events ever occurs (even with other events interleaved), then the error state will be entered.	Sy	
	If a node group is marked as having multiple concurrent nodes (i.e., it has an asterisk attached to it), then the EIODs semantics state that the node group can be replaced with a parallel fork/join edge in which the node group is repeated an undetermined number of times.	Sy	
SD to AD	A self call action of an object in a SD translates to an action of the corresponding participant of an AD.	Sy	[188]
	A message passing description in a SD is translated using the notations such as send signal, receive signal, object nodes, and transition between the signal node and the object node.	Sy	
	When a reference is used in a SD, we omit explicit reference notation and simply use a reference name. This should then be later expanded after the expansion of the reference of a SD is recursively translated into the AD and connected to the rest of the AD	Sy	
	When references are used, the reference type of a SD specification is translated into the interaction occurrence type of an AD.	Sy	
	The application of the alt operator of a SD can be transformed into an AD by using the synchronization bars, decision nodes and the merge nodes.	Sy	
	In order to transform a loop operator of a SD into an AD, setup, test, and body Sections should be identified first. After finishing the body Section, each process that participates in the body Section goes back to the join node to check the condition again.	Sy	
	The par operator of a SD represents the parallel composition of multiple scenarios. By using the fork and join nodes of an AD, a participant can be split into multiple threads and joined together.	Sy	
	Create an SMD for every distinct class implied by the objects in the COD.	Se	[191]
	Introduce as state variables all variables which are not attributes of the objects of COD.	Sy	
	Create transitions for the objects from which messages are sent.	Sy	

S.T.	UML Consistency rules		Ref.
COD to SMD	Create transitions for the objects to which messages are sent.	Sy	
	For all SMDs, put the set of transitions generated into correct sequences, connecting them by states, split bars and merge bars.	Sy	
SMD to SD	Actions are messages sent or actions performed by the object whose behavior is described by the SMD. Event triggers of transitions are messages received by the object.	Sy	[182]
	Actions that are attached to transitions or states are transformed into corresponding actions of an interaction. These actions imply messages and classifier roles.	Sy	
	Signal events and call events associated with transitions or states are also mapped onto send actions and call actions, respectively.	Sy	
	The external stimuli can also be given as an SMD, thus defining the response of an external actor to the output of the system.	Sy	
SMD to CD	Signal events of transitions and states are mapped onto signals and ultimately onto behavioral features that define their context.	Sy	[182]
	Call events are mapped onto operations of classifiers.	Sy	
	Send actions and call actions are mapped onto signals and behavioral features, and operations, respectively.	Sy	
	Actions are also used to infer stimuli and corresponding links which are in turn mapped onto associations and association ends of classifiers.	Sy	
	The context of the state machine is mapped onto the classifier for which the behavior is defined.	Se	

g. Types of consistency rules (RQ7)

The results obtained for RQ7 show that all the 47 rules were Horizontal (100%) and the vast majority of them Syntactic (87%, 41 out of 47 rules). Moreover, we found 6 (13%) Semantic rules. We conjecture that the main reason for this is that syntactic rules are easier to specify than semantic rules and that the UML standard more formally specifies syntax than semantics, which hinders the specification of semantic rules. It may also be the case that semantic ones are specific to the way in which the UML notation is actually used, which is organization, project, or team specific, and are therefore seldom described in published manuscripts.

III. Summary

Table 23 shows some results regarding the seven research questions according to each of the 14 primary studies [182-184, 187-197].

Table 23. Summary of Synthesis and Support

Ref.	Research papers	UML version	Year	Diagram synthesized	Technique used	Automatic support
[187]	PS	2.1.1	2010	IOD to SMD	Synthesis Algorithm	YES
[183]	ER	1.3	2001	SD to CD	Two tr. approaches	YES
[193]	PS	1.3	2000	SD to CD	Synthesis Algorithm	YES
[184]	PS	1.3	2001	SD to SMD	Synthesis Algorithm	YES
[192]	PS	1.4	2003	SD to SMD	Synthesis Algorithm	YES
[189]	PS	2.0	2004	SD to SMD	Algebraic framework Synthesis Algorithm	YES
[190]	PS	1.3	2000	SD to SMD	Synthesis Algorithm	YES
[188]	PS	2.1.2	2010	SD to AD	Mapping Rules Synthesis Algorithm	NO
[195]	PS	1.3	2000	SD to SMD	Synthesis Algorithm	YES
[196]	PS	1.3	2000	SD to SMD	Synthesis Algorithm	YES
[197]	PS	2.2	2014	SD to SMD	Synthesis Algorithm	YES
[194]	PS	2.0	2016	AD to SMD	Automated Approach	YES
[182]	VR	1.4	2003	SD to SMD SMD to SD/CD	Synthesis Algorithm	YES
[191]	PS	1.3	1998	COD to SMD	Synthesis Algorithm	NO

It is interesting to note the absence of synthesis techniques published between 2004 and 2010. Having identified nine primary studies with an old version of UML (64%) [182-184, 190-193, 195,

196] emphasizes that the synthesis of UML diagrams has started to become relevant since the initial launch of the UML (it has been evolving since the second half of the 1990s [9]). Furthermore, the other five primary studies that rely on the more recent UML versions 2.x [187-189, 194, 197] prove that the new version of the metamodel did have an impact on research in this domain. Another interesting aspect is that 11 of the 16 techniques synthesize UML sequence diagrams (68%).

Observations about the UML diagrams involved in synthesis techniques show that the synthesis of state machine diagrams from a collection of scenarios (i.e. sequence diagrams) has received a lot of attention. Finally, we have observed that the researchers who are most actively involved in UML synthesis techniques are Jon Whittle [187, 195, 196] and Petri Selonen [182, 183, 193], both with three publications.

IV. Conclusions

This appendix presents the results obtained after applying the Systematic Mapping Study (SMS) protocol with the aim of identifying and evaluating the current approaches for synthesizing UML diagrams from other UML diagrams. To the best of our knowledge, no such mapping studies or reviews existed prior to this work. The SMS was carried out systematically following well-known guidelines [31]. A significant amount of space in this appendix has been used to discuss how these guidelines were followed, since we felt that it was of the upmost importance to allow readers to precisely understand the results and conclusions, and to allow others to replicate our study or compare our results with others. From an initial set of 108 papers published in literature and extracted from seven scientific research databases, a total of 14 studies were eventually analyzed in depth, by following a precise selection protocol driven by seven research questions. We observe that (in no particular order of importance):

1) There is no commercially available UML CASE tool standard with which to run UML synthesis techniques between UML diagrams.

2) The most frequently used UML diagram type as a source for synthesis is the sequence diagram (68%), and the most synthesized diagram types are the state machine diagram (68%)

followed by class diagram (19%). This is not entirely surprising since these are the most frequently used UML diagrams [44].

3) We did not find any synthesis techniques for Communication, Protocol State Machine, Object, Interaction, Composite Structure, Timing and Deployment Diagrams. This may mean: a) that no such synthesis really makes sense; b) that additional research on synthesis techniques involving all 14 UML diagrams is warranted; c) or that those UML diagrams are least understood or least interesting.

4) We presented a set of 47 UML consistency rules systematically collected from the 14 primary studies. Having identified such a set of rules may provide several benefits for the MISE community: it provides a reference for practitioners, educators, and researchers alike; it provides guidance as to which rules to use in each context; and finally, it highlights which areas (i.e., what software development phase) are more developed and which need further work. Moreover, the set of rules could be a good input to the UML revision task force for inclusion in a forthcoming standard.

5) All the 47 identified rules were horizontal and most of them syntactic (a few semantic). We did not find any vertical, invocation, observation and evolution consistency rules. One reason for this could be the fact that the UML syntax is more formally described (by the UML metamodel) in the specification than semantics, and creating syntactic rules may therefore prove more feasible. These results show that consistency rules specified in the context of UML synthesis techniques focus only on UML horizontal consistency problems.

6) Synthesizing UML diagrams from other UML diagrams is not a very active line of research since the most recent papers were published in 2010 [187, 188], in 2014 [197] and only one in 2016 [194]. In addition, we only collected 14 primary studies. We wonder whether the fact that we did not find many primary studies is likely due to a change of terminology, i.e. the MDE community nowadays prefers the term “transformation” to the term “synthesis”. Nevertheless, considering that any UML synthesis technique needs to enforce consistency between the input UML diagram and the output synthesized UML diagram, this topic is a very mature and relevant topic [11].

Finally, thanks to the synthesis techniques presented in the papers collected, a UML model, and its UML diagrams attain a higher quality: increased consistency and correctness, because they are

either produced or updated automatically, or are checked against each other using the synthesis techniques. We feel that the time is ripe for these UML synthesis techniques to be put through their paces and that in-depth studies as well as replications [199] on how they can most effectively support UML consistency rules should now be undertaken.

Appendix D

This appendix contains the list of 95 primary studies discussed in 3.1.

Table 24. 95 Primary Studies

PS1	Zisman, A., and Kozlenkov, A. 2001. Knowledge Base Approach to Consistency Management of UML Specifications. In *Proceedings of the 16th IEEE international conference on Automated software engineering* (Coronado Island, San Diego, CA, USA, 2001). ASE '01. IEEE Computer Society, 359.
PS2	Zhao, X., Long, Q., and Qiu, Z. 2006. Model checking dynamic UML consistency. In *Proceedings of the 8th International Conference on Formal Engineering Methods*, Liu, Z., and He, J., Ed. (Macao, China, November 1-3, 2006). ICFEM '06. Springer-Verlag, 440-459. Doi=10.1007/11901433_24.
PS3	Zapata, C.M., González, G., and Gelbukh, A. 2007. A rule-based system for assessing consistency between UML models. In *Proceedings of the 6th Mexican International Conference on Advances in Artificial Intelligence*, Gelbukh, A., and Morales, F.K., Ed. (Aguascalientes, Mexico, November 4-10, 2007). MICAI'07. Springer-Verlag, 215-224. Doi=10.1007/978-3-540-76631-5_21.
PS4	Yang, J., Long, Q., Liu, Z., and Li, X. 2004. A predicative semantic model for integrating UML models. In *Proceedings of the 1st international Conference on Theoretical Aspects of Computing*, Liu, Z., and Araki, K., Ed. (Guiyang, China, 2004). ICTAC '04. Springer-Verlag, 170-186. Doi=10.1007/978-3-540-31862-0_14.
PS5	Yang, J. 2009. A framework for formalizing UML models with formal language rCOS. In *Proceedings of the 4th International Conference on Frontier of Computer Science and Technology* (Shanghai, China, December 17-19, 2009). FCST '09. IEEE, 408-416.
PS6	Xiaoshan, L., Zhiming, L., and He, J. 2004. A formal semantics of UML sequence diagram. In *Proceedings of the Australian Conference on Software Engineering* (Melbourne, Australia, April 13-16, 2004). ASWEC '04. IEEE Computer Society, 168-177. Doi=10.1109/aswec.2004.1290469.
PS7	Wilke, C., Bartho, A., Schroeter, J., Karol, S., and Aßmann, U. 2012. Elucidative development for model-based documentation. In *Proceedings of the 50th international conference on Objects, Models, Components, Patterns*, Furia, C.A., and Nanz, S., Ed. (Prague, Czech Republic, 2012). TOOLS'12. Springer-Verlag, 320-335. Doi=10.1007/978-3-642-30561-0_22.
PS8	Wang, Z., He, H., Chen, L., and Zhang, Y. 2012. Ontology based semantics checking for UML activity model. *Information Technology Journal*. 11, 3, 301-306.
PS9	Wang, H., Feng, T., Zhang, J., and Zhang, K. 2005. Consistency check between behaviour models. In *Proceedings of the International Symposium on Communications and Information Technology* (Beijing, China, 2005). ISCIT '05. IEEE, 486-489.

PS10	Wagner, R., Giese, H., and Nickel, U. 2003. A plug-in for flexible and incremental consistency management. In *Proceedings of the International Conference on the Unified Modeling Language 2003 (Workshop 7: Consistency Problems in UML-based Software Development)* (San Francisco, USA, 2003) Technical Report, Blekinge Institute of Technology.
PS11	Vasilecas, O., Dubauskaitė, R., and Rupnik, R. 2011. Consistency checking of UML business model. *Technological and Economic Development of Economy*. 17, 1, 133-150.
PS12	Vasilecas, O., and Dubauskaite, R. 2009. Ensuring consistency of information systems rules models. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, Rachev, B., and Smrikarov, A., Ed. (Ruse, Bulgaria, 2009). CompSysTech '09. ACM, 96.
PS13	Van Der Straeten, R., Mens, T., Simmonds, J., and Jonckers, V. 2003. Using Description Logic to Maintain Consistency between UML Models. In *Proceedings of the 6th International Conference on Unified Modeling Language, Modeling Languages and Applications*, Stevens, P., Whittle, J., and Booch, G., Ed. (San Francisco, CA, USA, October 20-24, 2003) Springer Berlin Heidelberg, 326-340. Doi=10.1007/978-3-540-45221-8_28.
PS14	Van Der Straeten, R., Jonckers, V., and Mens, T. 2007. A formal approach to model refactoring and model refinement. *Software & Systems Modeling*. 6, 2, 139-162. Doi=10.1007/s10270-006-0025-9.
PS15	Van Der Straeten, R., and D'Hondt, M. 2006. Model refactorings through rule-based inconsistency resolution. In *Proceedings of the ACM symposium on Applied computing* (Dijon, France, 2006). SAC '06. ACM, 1210-1217. Doi=10.1145/1141277.1141564.
PS16	Van Der Straeten, R. 2009. ALLOY for Inconsistency Resolution in MDE. In *Proceedings of the BENEVOL 2009 The 8 th BElgian-NEtherlands software eVOLution seminar* (Université catholique de Louvain - Belgium, December, 2009). BENEVOL '09.54.
PS17	Van Der Straeten, R. 2004. Inconsistency detection between UML models using RACER and nRQL. In *Proceedings of the KI-2004 Workshop on Applications of Description Logics* (Ulm, Germany, September, 2004). ADL'04. CEUR Workshop.
PS18	Szlenk, M. 2006. Formal Semantics and Reasoning about UML Class Diagram. In *Proceedings of the International Conference on Dependability of Computer Systems* (Szklarska Poręba, Poland, May 25-27, 2006). DEPCOS-RELCOMEX '06. IEEE Computer Society, 51-59. Doi=10.1109/depcos-relcomex.2006.27.
PS19	Stumptner, M., and Schrefl, M. 2000. Behavior consistent inheritance in UML. In *Proceedings of the 19th international conference on Conceptual modeling*, Laender, A.H.F., Liddle, S.W., and Storey, V.C., Ed. (Salt Lake City, Utah, USA, October 9–12, 2000). ER'00. Springer-Verlag, 527-542. Doi=10.1007/3-540-45393-8_38.
PS20	**[**Spanoudakis, G., and Kim, H. 2002. Diagnosis of the significance of inconsistencies in object-oriented designs: a framework and its experimental evaluation. *Journal of Systems and Software*. 64, 1 (October 2002), 3-22. Doi=10.1016/S0164-1212(02)00018-3.

PS21	Spanoudakis, G., Kasis, K., and Dragazi, F. 2004. Evidential diagnosis of inconsistencies in object-oriented designs. *International Journal of Software Engineering and Knowledge Engineering*. 14, 2, 141-178.
PS22	Song, I.-Y., Khare, R., An, Y., and Hilsbos, M. 2008. A Multi-level Methodology for Developing UML Sequence Diagrams. In *Proceedings of the 27th International Conference on Conceptual Modeling*, Li, Q., Spaccapietra, S., Yu, E., and Olivé, A., Ed. (Barcelona, Spain, October 20-24, 2008). ER '08. Springer-Verlag, 114-127. Doi=10.1007/978-3-540-87877-3_10.
PS23	Snoeck, M., Michiels, C., and Dedene, G. 2003. Consistency by construction: the case of MERODE. In *Proceedings of the Conceptual Modeling for Novel Application Domains, Workshops ECOMO, IWCMQ, AOIS, and XSDM*, Jeusfeld, M.A., and Pastor, O., Ed. (Chicago, IL, USA, October 13, 2003). ER '03. Springer-Verlag, 105-117. Doi=978-3-540-20257-8.
PS24	Shengjun, W., Longfei, J., and Chengzhi, J. 2006. Ontology Definition Metamodel based Consistency Checking of UML Models. In *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design* (Nanjing, China, May 3-5 2006). CSCWD '06. IEEE Computer Society, 1-5. Doi=10.1109/cscwd.2006.253005.
PS25	Shen, W., Wang, K., and Egyed, A. 2009. An efficient and scalable approach to correct class model refinement. *IEEE Transactions on Software Engineering*. 35, 4, 515-533.
PS26	Sengupta, S., and Bhattacharya, S. 2008. Formalization of UML diagrams and their consistency verification: A Z notation based approach. In *Proceedings of the 1st India software engineering conference* (Hyderabad, India, 2008). ISEC '08. ACM, 151-152. Doi=10.1145/1342211.1342248.
PS27	Schwarzl, C., and Peischl, B. 2010. Static- and dynamic consistency analysis of UML state chart models. In *Proceedings of the 13th international conference on Model driven engineering languages and systems: Part I*, Petriu, D.C., Rouquette, N., and Haugen, Ø., Ed. (Oslo, Norway, 2010). MODELS '10. Springer-Verlag.
PS28	Satish, S.S., Shashikant, S.R., Sambhe, V.K., Shelke, R.B., and Kocharekar, G. 2010. A minimum cardinality consistency-checking algorithm for UML class diagrams. In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology* (Mumbai, Maharashtra, India, 2010). ICWET '10. ACM, 222-223.
PS29	**[**Sapna, P.G., and Mohanty, H. 2007. Ensuring consistency in relational repository of UML models. In *Proceedings of the 10th International Conference on Information Technology* (Orissa, India, December 17-20, 2007). ICIT '07. IEEE Computer Society, 217-222.
PS30	Ruta, D., and Olegas, V. 2010. The approach of ensuring consistency of UML model based on rules. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, Rachev, B., and Smrikarov, A., Ed. (Sofia, Bulgaria, June 17-18, 2010). CompSysTech '10. ACM, 71-76. Doi=10.1145/1839379.1839393.

PS31	Reder, A., and Egyed, A. 2012. Computing repair trees for resolving inconsistencies in design models. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering* (Essen, Germany, September 3-7, 2012). ASE '12. ACM, 220-229. Doi=10.1145/2351676.2351707.
PS32	Rasch, H., and Wehrheim, H. 2003. Checking Consistency in UML Diagrams: Classes and State Machines. In *Proceedings of the 6th IFIP WG 6.1 International Conference*, Najm, E., Nestmann, U., and Stevens, P., Ed. (Paris, France, November 19-21, 2003). FMOODS '03. Springer Berlin Heidelberg, 229-243. Doi=10.1007/978-3-540-39958-2_16.
PS33	Quan, L., Zhiming, L., Xiaoshan, L., and He, J. 2005. Consistent code generation from UML models. In *Proceedings of the Australian Conference on Software Engineering* (Brisbane, Australia, 29 March-1 April 2005). ASWEC '05. IEEE Computer Society, 23-30. Doi=10.1109/aswec.2005.17.
PS34	Qiu, H. 2004. *Consistency Checking of UML-LIGHT with CSP*. WS 2004/2005: Seminar : Modellbasierte Softwareentwichklung University of Paderborn.
PS35	Pilskalns, O., Williams, D., Aracic, D., and Andrews, A. 2006. Security Consistency in UML Designs. In *Proceedings of the 30th Annual International Computer Software and Applications Conference* (Chicago, USA, September 17-21, 2006). COMPSAC '06. IEEE Computer Society, 351-358. Doi=10.1109/compsac.2006.76.
PS36	**[**Petriu, D.C., and Sun, Y. 2000. Consistent behaviour representation in activity and sequence diagrams. In *Proceedings of the 3rd international conference on The unified modeling language: advancing the standard*, Evans, A., Kent, S., and Selic, B., Ed. (York, UK, October 2–6, 2000). UML'00. Springer-Verlag, 369-382.
PS37	Pelliccione, P., Inverardi, P., and Muccini, H. 2009. CHARMY: A Framework for Designing and Verifying Architectural Specifications. *IEEE Transactions on Software Engineering*. 35, 3, 325-346. Doi=10.1109/tse.2008.104.
PS38	Pattanasri, N., Wuwongse, V., and Akama, K. 2004. XET as a Rule Language for Consistency Maintenance in UML. In *Proceedings of the 3rd International Workshop*, Antoniou, G., and Boley, H., Ed. (Hiroshima, Japan, November 8, 2004). RuleML '04. Springer-Verlag, 200-204. Doi=10.1007/978-3-540-30504-0_17.
PS39	Pap, Z., Majzik, I., Pataricza, A., and Szegi, A. 2005. Methods of checking general safety criteria in UML statechart specifications. *Reliability Engineering & System Safety*. 87, 1, 89-107. Doi=http://dx.doi.org/10.1016/j.ress.2004.04.011.
PS40	Paige, R.F., Kolovos, D.S., and Polack, F.A.C. 2005. Refinement via Consistency Checking in MDA. *Electronic Notes in Theoretical Computer Science*. 137, 2, 151-161. Doi=http://dx.doi.org/10.1016/j.entcs.2005.04.029.
PS41	Paige, R.F., Brooke, P.J., and Ostroff, J.S. 2007. Metamodel-based model conformance and multiview consistency checking. *ACM Transactions Software Engineering Methodology*. 16, 3 (July 2007), 11. Doi=10.1145/1243987.1243989.
PS42	Ohnishi, A. 2002. A supporting system for verification among models of the UML. *Systems and Computers in Japan*. 33, 4 (April 2002), 1-3. Doi=10.1002/scj.10016.

PS43	Ober, I., and Dragomir, I. 2011. Unambiguous UML composite structures: the OMEGA2 experience. In *Proceedings of the 37th international conference on Current trends in theory and practice of computer science*, Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., and Wolf, S., Ed. (Nový Smokovec, Slovakia, January 22-28, 2011). SOFSEM '11. Springer-Verlag, 418-430. Doi=10.1007/978-3-642-18381-2_35.
PS44	Nimiya, A., Yokogawa, T., Miyazaki, H., Amasaki, S., Sato, Y., and Hayase, M. 2010. Model checking consistency of UML diagrams using alloy. *World Academy of Science, Engineering and Technology*. 71, 47, 547-550.
PS45	Nakanishi, H., Miura, T., and Shioya, I. 2004. Formalizing UML collaborations by using description logics. In *Proceedings of the 2nd IEEE International Conference on Computational Cybernetics* (Vienna, Austria, 2004, 2004). ICCC '04. IEEE Computer Society, 243-248. Doi=10.1109/icccyb.2004.1437718.
PS46	Muskens, J., Bril, R.J., and Chaudron, M.R.V. 2005. Generalizing Consistency Checking between Software Views. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture* (Pittsburgh, Pennsylvania, USA, November 6-10, 2005). WICSA '05. IEEE Computer Society, 169-180. Doi=10.1109/wicsa.2005.37.
PS47	Mens, T., Van Der Straeten, R., and D'Hondt, M. 2006. Detecting and resolving model inconsistencies using transformation dependency analysis. In *Proceedings of the 9th international conference on Model Driven Engineering Languages and Systems*, Nierstrasz, O., Whittle, J., Harel, D., and Reggio, G., Ed. (Genova, Italy, 2006). MODELS '06. Springer-Verlag, 200-214. Doi=10.1007/11880240_15.
PS48	Martínez, F.J.L., and Álvarez, A.T. 2005. A precise approach for the analysis of the UML models consistency. In *Proceedings of the 24th international conference on Perspectives in Conceptual Modeling*, Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., and Comyn-Wattiau, I., Ed. (Klagenfurt, Austria, 2005). ER '05. Springer-Verlag, 74-84. Doi=10.1007/11568346_9.
PS49	Malgouyres, H., and Motet, G. 2006. A UML model consistency verification approach based on meta-modeling formalization. In *Proceedings of the ACM symposium on Applied computing* (Dijon, France, April 23-27, 2006). SAC '06. ACM, 1804-1809. Doi=10.1145/1141277.1141703
PS50	**[**Litvak, B., Tyszberowicz, S., and Yehudai, A. 2003. Behavioral consistency validation of UML diagrams. In *Proceedings of the 1st International Conference on Software Engineering and Formal Methods* (Brisbane, Australia, September 22-27, 2003). SEFM '03. IEEE Computer Society, 118-125. Doi=10.1109/sefm.2003.1236213.
PS51	Ledang, H. 2004. B-based Consistency Checking of UML Diagrams. In *Proceedings of the 2nd National Symposium on Research, Development and Application of Information and Communication Technology* (Hanoi, Vietnam, October 2006, 2004). ICT.rda '04.1-10.
PS52	Lavanya, K.C., Bala, K.V., Mohanty, H., and Shyamasundar, R.K. 2005. How Good is a UML Diagram? A Tool to Check It. In *Proceedings of the IEEE Region 10*

	(Melbourne, Australia, November 21-24, 2005). TENCON '05. IEEE Computer Society, 1-5. Doi=10.1109/tencon.2005.301101.
PS53	**[**Lano, K. 2007. Formal specification using interaction diagrams. In *Proceedings of the 5th IEEE International Conference on Software Engineering and Formal Methods* (London; United Kingdom, September 10-14, 2007). SEFM '07. IEEE Computer Society, 293-301. Doi=10.1109/SEFM.2007.20.
PS54	Lange, C.F.J., and Chaudron, M.R.V. 2006. Effects of defects in UML models: an experimental investigation. In *Proceedings of the 28th international conference on Software engineering* (Shanghai, China, 2006). ICSE '06. ACM, 401-411. Doi=10.1145/1134285.1134341.
PS55	Laleau, R.g., and Polack, F. 2008. Using formal metamodels to check consistency of functional views in information systems specification. *Information and Software Technology*. 50, 7-8, 797-814. Doi=http://dx.doi.org/10.1016/j.infsof.2007.10.007.
PS56	Lagarde, F., Espinoza, H., Terrier, F., and Gérard, S. 2007. Improving uml profile design practices by leveraging conceptual domain models. In *Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering* (Atlanta, Georgia, USA, 2007). ASE '07. ACM, 445-448. Doi=10.1145/1321631.1321705.
PS57	Labiche, Y. 2008. The UML is more than boxes and lines. In *Proceedings of the Workshops and Symposia at MODELS 2008*, Chaudron, M.R.V., Ed. (Toulouse, France, September 28 - October 3, 2008). MODELS '08. Springer-Verlag, 375-386. Doi=10.1007/978-3-642-01648-6_39.
PS58	Kuster, J., and Stroop, J. 2001. Consistent design of embedded real-time systems with UML-RT. In *Proceedings of the 4th International Symposium on Object-Oriented Real-Time Distributed Computing* (Magdeburg, Germany, 2001). ISORC '01. IEEE Computer Society, 31-40. Doi=10.1109/isorc.2001.922815.
PS59	Kim, S.K., and Carrington, D. 2004. A formal object-oriented approach to defining consistency constraints for UML models. In *Proceedings of the Australian Conference on Software Engineering* (Melbourne, Australia, April 13-16, 2004). ASWEC '04. IEEE Computer Society, 87-94.
PS60	Kienzle, J., Abed, W.A., and Klein, J. 2009. Aspect-oriented multi-view modeling. In *Proceedings of the 8th ACM international conference on Aspect-oriented software development* (Charlottesville, Virginia, USA, 2009). AOSD '09. ACM, 87-98. Doi=10.1145/1509239.1509252.
PS61	Khai, Z., Nadeem, A., and Lee, G.-s. 2011. A Prolog Based Approach to Consistency Checking of UML Class and Sequence Diagrams. In *Proceedings of the Communication and Networking - International Conference (FGCN 2011), Held as Part of the Future Generation Information Technology Conference (FGIT 2011), in Conjunction with GDC 2011*, Kim, T.-h., Adeli, H., Kim, H.-k., Kang, H.-j., Kim, K.J., Kiumi, A., and Kang, B.-H., Ed. (Jeju Island, South Korea, December 8-10, 2011). FGCN '11. Springer-Verlag, 85-96. Doi=10.1007/978-3-642-27207-3_10.

PS62	Kanjilal, A., Sengupta, S., and Bhattacharya, S. 2009. Analysis of object-oriented design: A metrics based approach. In *Proceedings of the IEEE Region 10 Annual International Conference* (Singapore; Singapore, November 23-26, 2009). TENCON '09. IEEE Computer Society, 1 - 6
PS63	Kaneiwa, K., and Satoh, K. 2010. On the complexities of consistency checking for restricted UML class diagrams. *Theoretical Computer Science*. 411, 2 (January 2010), 301-323. Doi=http://dx.doi.org/10.1016/j.tcs.2009.04.030.
PS64	Jing, L., Zhiming, L., Jifeng, H., and Xiaoshan, L. 2004. Linking UML models of design and requirement. In *Proceedings of the Australian Conference on Software Engineering* (Melbourne, Australia, April 13-16, 2004). ASWEC '04. IEEE Computer Society, 329-338. Doi=10.1109/aswec.2004.1290486.
PS65	Il-kyu, H., and Kang, B. 2008. Cross Checking Rules to Improve Consistency between UML Static Diagram and Dynamic Diagram. In *Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, Fyfe, C., Kim, D., Lee, S.-Y., and Yin, H., Ed. (Daejeon, South Korea, 2008). IDEAL '08. Springer-Verlag, 436-443. Doi=10.1007/978-3-540-88906-9_55.
PS66	**[**Il-Kyu, H., and Byung-Wook, K. 2003. Meta-validation of UML structural diagrams and behavioral diagrams with consistency rules. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and signal Processing* (Victoria, B.C., Canada, August 28-30, 2003). PACRIM '03. IEEE Computer Society, 679-683. Doi=10.1109/pacrim.2003.1235872.
PS67	Ibrahim, N., Ibrahim, R., Saringat, M.Z., Mansor, D., and Herawan, T. 2011. Consistency rules between UML use case and activity diagrams using logical approach. *International Journal of Software Engineering and its Applications*. 5, 3, 119-134.
PS68	Ibrahim, N., Ibrahim, R., and Saringat, M.Z. 2011. Definition of Consistency Rules between UML Use Case and Activity Diagram. In *Proceedings of the 2nd International Conference*, Kim, T.-h., Adeli, H., Robles, R.J., and Balitanas, M., Ed. (Daejeon, South Korea, April 13-15, 2011). UCMA '11. Springer-Verlag, 498-508. Doi=10.1007/978-3-642-20998-7_58.
PS69	Hilsbos, M., and Song, I.-Y. 2004. Use of Tabular Analysis Method to Construct UML Sequence Diagrams. In *Proceedings of the 23th International Conference on Conceptual Modeling* (Shanghai, China, Nov 8-12, 2004). ER '04. Springer-Verlag, 740-752. Doi=10.1007/978-3-540-30464-7_55.
PS70	Hausmann, J.H., Heckel, R., and Sauer, S. 2002. Extended model relations with graphical consistency conditions. In *Proceedings of the UML 2002 Workshop on Consistency Problems in UML-based Software Development* (Blekinge Institute of Technology, 2002) Citeseer, 61-74.
PS71	Hammal, Y. 2008. A modular state exploration and compatibility checking of UML dynamic diagrams. In *Proceedings of the 6th IEEE/ACS International Conference on Computer Systems and Applications* (Doha, Qatar, 2008). AICCSA '08. IEEE Computer Society, 793-800. Doi=10.1109/AICCSA.2008.4493617.

PS72	Hamed, H., and Salem, A. 2001. UML-L: a UML based design description language. In *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications* (Beirut, Lebanon, June 2001, 2001). AICCSA'01. IEEE Computer Society, 438-441. Doi=10.1109/aiccsa.2001.934037.
PS73	Engels, G., Küster, J.M., Heckel, R., and Groenewegen, L. 2001. A methodology for specifying and analyzing consistency of object-oriented behavioral models. *Sigsoft Software Engineering Notes*. 26, 5 (September 2001), 186-195. Doi=10.1145/503271.503235.
PS74	Engels, G., Heckel, R., and Küster, J.M. 2001. Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model. In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, Gogolla, M., and Kobryn, C., Ed. (Toronto, Ontario, Canada, October 1 - 5, 2001). UML'01. Springer-Verlag, 272-286.
PS75	Egyed, A., Letier, E., and Finkelstein, A. 2008. Generating and evaluating choices for fixing inconsistencies in UML design models. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering* (L'Aquila, Italy, 2008). ASE '08. IEEE Computer Society, 99-108. Doi=10.1109/ASE.2008.20.
PS76	Egyed, A. 2007. UML/Analyzer: A Tool for the Instant Consistency Checking of UML Models In *Proceedings of the 29th international conference on Software Engineering* (Minneapolis, MN, USA, 2007). ICSE '07. IEEE Computer Society, 793-796. Doi=10.1109/icse.2007.91.
PS77	Egyed, A. 2007. Fixing Inconsistencies in UML Design Models. In *Proceedings of the 29th international conference on Software Engineering* (Minneapolis, MN, USA, 2007). ICSE '07. IEEE Computer Society, 292-301. Doi=10.1109/icse.2007.38.
PS78	Egyed, A. 2006. Instant consistency checking for the UML. In *Proceedings of the 28th international conference on Software engineering* (Shanghai, China, 2006). ICSE '06. ACM, 381-390. Doi=10.1145/1134285.1134339.
PS79	Egyed, A. 2004. Consistent adaptation and evolution of class diagrams during refinement. In *Proceedings of the 7th International Conference (FASE 2004), Held as Part of the Joint European Conferences on Theory and Practice of Software* (Barcelona, Spain, 2004). ETAPS '04. Springer-Verlag, 37-53. Doi=10.1007/978-3-540-24721-0_3.
PS80	Egyed, A. 2000. Semantic abstraction rules for class diagrams. In *Proceedings of the 15th IEEE international conference on Automated software engineering* (Grenoble, France, September 11-15, 2000). ASE '00. IEEE Computer Society, 301-304. Doi=10.1109/ase.2000.873683.
PS81	Du, D., Liu, J., Cao, H., and Zhang, M. 2009. BAS: A Case Study for Modeling and Verification in Trustable Model Driven Development. *Electronic Notes in Theoretical Computer Science*. 243, 28 (July 2009), 69-87. Doi=http://dx.doi.org/10.1016/j.entcs.2009.07.006.

PS82	Costagliola, G., Deufemia, V., Ferrucci, F., and Gravino, C. 2003. Exploiting Visual Languages Generation and UML Meta Modeling to Construct Meta-CASE Workbenches. *Electronic Notes in Theoretical Computer Science*. 72, 3, 25-35. Doi=http://dx.doi.org/10.1016/S1571-0661(04)80609-1.
PS83	Chiorean, D., Pasca, M., Carcu, A., Botiza, C., and Moldovan, S. 2004. Ensuring UML Models Consistency Using the OCL Environment. *Electronic Notes in Theoretical Computer Science*. 102 (November 2004), 99-110. Doi=10.1016/j.entcs.2003.09.005.
PS84	Chen, Z., and Motet, G. 2009. A language-theoretic view on guidelines and consistency rules of UML. In *Proceedings of the 5th European Conference*, Paige, R.F., Hartman, A., and Rensink, A., Ed. (Enschede, The Netherlands, June 23-26, 2009). ECMDA-FA '09. Springer-Verlag, 66-81.
PS85	Chang, K.N. 2008. Model checking consistency between sequence and state diagrams. In *Proceedings of the International Conference on Software Engineering Research and Practice* (Las Vegas, NV, USA, July 14-17, 2008). SERP '08.457-461.
PS86	Chanda, J., Kanjilal, A., Sengupta, S., and Bhattacharya, S. 2009. Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach. In *Proceedings of the International Conference on Methods and Models in Computer Science* (New Delhi, India, December 14-15, 2009). ICM2CS '09. IEEE Computer Society, 1-4. Doi=10.1109/icm2cs.2009.5397941.
PS87	Chanda, J., Janowski, T., Mohanty, H., Kanjilal, A., and Sengupta, S. 2010. UML-Compiler: A Framework for Syntactic and Semantic Verification of UML Diagrams. In *Proceedings of the 6th international conference on Distributed Computing and Internet Technology*, Janowski, T., and Mohanty, H., Ed. (Bhubaneswar, India, February 15-17, 2010). ICDCIT'10. Springer-Verlag, 194-205. Doi=10.1007/978-3-642-11659-9_22.
PS88	Campbell, L.A., Cheng, B.H.C., McUmber, W.E., and Stirewalt, R.E.K. 2002. Automatically Detecting and Visualising Errors in UML Diagrams. *Requirements Engineering*. 7, 4, 264-287. Doi=10.1007/s007660200020.
PS89	Calì, A., Calvanese, D., De Giacomo, G., and Lenzerini, M. 2002. A Formal Framework for Reasoning on UML Class Diagrams. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, Hacid, M.-S., Ras, Z.W., Zighed, D.A., and Kodratoff, Y., Ed. (Lyon, France, 27–29 June, 2002). ISMIS '02. Springer-Verlag, 503-513.
PS90	Briand, L.C., Labiche, Y., and O'Sullivan, L. 2003. Impact analysis and change management of UML models. In *Proceedings of the International Conference on Software Maintenance* (Amsterdam, The Netherlands, September 2003, 2003). ICSM '03. IEEE Computer Society, 256-265.
PS91	**[**Borba, C.F., and Da SIlva, A.E.A. 2010. Knowledge-based system for the maintenance registration and consistency among UML diagrams. In *Proceedings of the 20th Brazilian Conference on Advances in Artificial Intelligence*, Costa, C.D.R.,

	Vicari, R.M., and Tonidandel, F., Ed. (São Bernardo do Campo, Brazil, October 2010, 2010). SBIA'10. Springer-Verlag, 51-61.
PS92	Bjørn, B. 2008. *Consistency checking UML interactions and state machines*. Master Thesis. University of Oslo.
PS93	Balaban, M., and Maraee, A. 2006. Consistency of UML class diagrams with hierarchy constraints. In *Proceedings of the 6th international Conference on Next Generation Information Technologies and Systems*, Etzion, O., Kuflik, T., and Motro, A., Ed. (Kibbutz Shefayim, Israel, July 4-6, 2006). NGITS'06. Springer-Verlag, 71-82.
PS94	Astesiano, E., and Reggio, G. 2003. An attempt at analysing the consistency problems in the UML from a classical algebraic viewpoint. In *Proceedings of the 16th International Workshop*, Wirsing, M., Pattinson, D., and Hennicker, R., Ed. (Frauenchiemsee, Germany, September 24-27, 2003). WADT '02. Springer-Verlag, 56-81.
PS95	Alanazi, M.N., and Gustafson, D.A. 2009. Super state analysis for UML state diagrams. In *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering* (Los Angeles, California, USA, 31 March - 2 April, 2009). CSIE '09. EEE Computer Society, 560-565.

Appendix E

A discussion in Section 3.1.1.2 the process of the search string (SS) selection is presented in detail in this appendix. All of the following SSs have been tried in the keywords, abstracts and titles of the papers by using the Scopus search engine. To carry out this process of trials, we chose Scopus as search engine because it is the largest abstract and citation database of peer-reviewed literature: scientific journals, books and conference proceedings[26].

The following SSs have been tried with the following results shown in Table 25.

The first SS we tried was the SS1 that is the SS of the closest piece of work that deals to our problem [200]. From SS1 to SS2, the search was extended by looking for "UML" instead of simply "model". Since by using SS2 the search was extended, more hints were obtained. From SS2 to SS3 the first part of SS3 was extended to include the following search strings (MANAGEMENT UML INCONSISTENCY and MANAGEMENT MODEL INCONSISTENCY). SS2 and SS3 are different for this reason, but we obtained the same hints with SS2 and SS3. SS4 is even more permissive than the other three, and therefore there are more hints, but it was felt that it should be dismissed because it will find some documents related only to the "management of the consistency or inconsistency" that might be not connected with the models and UML on which this research is focused.

Table 25. Search strings tried on 15/09/2012

Search strings	Res.
SS1 ((management AND model AND (inconsistency OR consistency)) OR (model AND (inconsistency OR consistency)))	335
SS2 ((management AND model AND (inconsistency OR consistency)) OR ((UML OR model) AND (inconsistency OR consistency)))	361
SS3 ((management AND (model OR UML) AND (inconsistency OR consistency)) OR ((UML OR model) AND (inconsistency OR consistency)))	361
SS4 (UML OR model OR management AND (inconsistency OR consistency))	438

[26] https://www.scopus.com

Consequently, we tried the following SSs (Table 26):

Table 26. Search strings tried on 05/10/2012.

Search Strings	Res.
SS5 (management AND (inconsistency OR consistency))	70
SS6 (management AND (model OR UML) AND (inconsistency OR consistency))	4
SS7 ((UML OR model) AND (inconsistency OR consistency))	361
SS8 ((management AND (model OR UML) AND (inconsistency OR consistency)) OR ((UML OR model) AND (inconsistency OR consistency)))	361

The 70 hints of SS5 would probably be only about the management of the (in)consistency, therefore there would not be any relation with models or UML. SS5 was inserted for completeness only, but no strong focus was given to it. The four hints of SS6 could be of interest to this research, but the number of hints was too low. The results of SS7 and SS8 were considered a good starting point to refine the final SS. The four hints of SS6 were also present in the results of SS7 and SS8. As the emphasis is not on the management of inconsistencies, the word "management" was dropped, so SS7 is the SS that is considered for the next step of SS refinement.

Starting with SS7 the following two SSs (Table 27) were created and tried with the following results:

Table 27. Search strings tried on 08/10/2012.

Search strings	Res.
SS9 ((UML or (unified modeling language) or model) and (consistency or inconsistency))	361
SS10 ((UML or (unified modeling language)) and (consistency or inconsistency))	43

The difference between the two SSs (Table 27) is that in SS9 (compared with SS10) the search was extended to look also for "model" with inconsistency or consistency, where some papers pertinent to UML might be found. In the case that an author writes about UML consistency, she/he cannot be expected to only mention "model", avoiding the word "UML" or the string "Unified

Modeling Language" in keywords, title and abstract. Therefore, it was felt that SS10 would provide a better screening of the results, because it is focused on UML consistency.

This sub-Section shows how the search fields were identified between abstract (ABS), keywords (KEY) and TITLE. By using SS10 with the Scopus search engine, it was possible to obtain the following hints:

Table 28. Search fields on 08/10/2012.

Search Field	Res.
SF1 KEY and ABS and TITLE	43
SF2 ABS and TITLE	70
SF3 TITLE and (ABS or KEY)	71
SF4 KEY and (ABS or TITLE)	125
SF5 ABS and (TITLE or KEY)	151
SF6 TITLE or ABS or KEY	587

Considering the high hints of papers reached, we chose the search field combination SF6 among the ones presented in Table 28.

There might exist papers that are related to this work and that might not mention consistency. For instance, there are papers that address the generation of state machines from a series of sequence diagrams. In doing this kind of exercise, the authors assume (and sometimes disclose) rules that the two sets of diagrams must follow, which in this context are consistency rules. It might be interesting to look for such works in addition to the current work. These works are more about model synthesis than anything else (see Section 3.2).

Table 29. SS and SF on 03/11/2012

Search strings and search field	Res.
SS10 ((UML or (unified modeling language) and (consistency or inconsistency))	587
SF6 TITLE or ABS or KEY	

Lastly, if additional uses of consistency rules were to be found, hopefully along with examples of consistency rules, we could consider other engineering activities that rely on UML diagrams. However, we believe that these rules might also be covered by our initial (SMS) search, so this one is kept for future studies. Consequently, we chose search string SS10 and search field combination SF6 (Table 29) to be used in our SMS.

The major search terms were “UML” and “Consistency” and the alternative spellings, synonyms or terms related to the major terms are presented in Table 29.

Table 30. Search string

Major Terms	Alternative terms
UML	(uml OR unified modeling language OR unified modelling language)
Consistency	(consistency OR inconsistency)