

THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Mining Software Modeling Practices In Open Source Software Projects

TRUONG HO-QUANG



Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Göteborg University
Göteborg, Sweden, 2017

Mining Software Modeling Practices In Open Source Software Projects

TRUONG HO-QUANG

Copyright ©2017 Truong Ho-Quang
except where otherwise stated.
All rights reserved.

Technical Report No 163L
ISSN 1652-876X
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and Göteborg University
Göteborg, Sweden

This thesis has been prepared using \LaTeX .
Printed by Chalmers Reproservice,
Göteborg, Sweden 2017.

“Extraordinary claims require extraordinary evidence.”
- *Carl Sagan*

Abstract

Context: In modern software development, software modeling is considered as an essential part of the software architecture and design activity. The Unified Modeling Language (UML) has become the de facto standard for software modeling in industry. Surprisingly, there are few empirical evidences on the use of UML and a lack of evidence-based guidelines for applying UML in software development.

Objective: As a first step toward synthesizing practical guidelines for the use of UML, this thesis focuses on collecting a large set of OSS projects that use UML. Subsequently this thesis offers observations on the use and impacts of using UML in OSS projects.

Method: We combine techniques from repository mining and image classification in order to successfully identify more than 24 000 open source projects on GitHub that together contain more than 93 000 UML models. A quantitative analysis and a large-scale survey have been carried out across this set of projects.

Result: The results show that UML is used in OSS projects and in those projects that use UML, UML helps new contributors and is generally perceived as supportive. The most important motivation for using UML seems to be to facilitate collaboration, as teams use UML during communication and planning of joint implementation efforts. We hope researchers in the field will find data and findings from this thesis a valuable source for their empirical studies.

Keywords: Software Engineering, Software Modeling, UML, FOSS, Empirical Study, Data Mining, Mining Repository, GitHub

Acknowledgment

To accomplish this Licentiate thesis, I have received lots of encouragement from colleagues, friends and my family. I would take this opportunity to thank:

My main supervisor Prof. Michel R.V. Chaudron, for the continuous support of my Ph.D study, for your patience, motivation, and immense knowledge.

Prof. Jordi Cabot (Open University of Catalonia - UOC), for being discusant to this Licentiate thesis.

My co-supervisor Regina Hebig, for voluntarily supporting me and providing me with tips and comments whenever needed.

My former co-supervisor Patrizio Pelliccione, for offering interesting discussions and constructive comments in the first two years of my PhD.

My examiner Prof. Ivica Crnkovic, for your insightful comments and encouragement, for hard questions which incent me to widen my research from various perspectives.

Gregorio Robles and Miguel Ángel Fernández for being part of the best team that I've ever had.

To all of my colleagues at the Software Engineering Division, including of course administrative staff: Thank you all for creating such a friendly and productive work environment. I particularly thank Rodi Jolak for sharing the office with me, for interesting discussions and pingpong sessions. Hugo, Federico, Grischa, Salome: for welcoming all of my questions when Google cannot help out.

To Dave, Hafeez and Bilal: I am thankful to be part of our team. Thanks to your help, I came to the PhD life less nervous.

To industrial partners Henrik Harmsen and Jesper Derehag, for bringing industrial perspectives to part of my research.

And to the most important people in my life: My lovely wife and the kid to-be-born - thank you for stepping into my life and making it full of joy and happiness everyday. I also owe much of my success to my parents, who supported me spiritually throughout writing this thesis and my life in general. Last but not least, thank to my brothers and their families for the great suggestions and motivation.

List of Publications

Included publications

This thesis is based on the following publications:

- [A] T. Ho-Quang, M.R.V. Chaudron, I. Samelsson, J. Hjaltason, B. Karasneh, H. Osman “Automatic Classification of UML Class Diagrams from Images”
21st Asia-Pacific Software Engineering Conference (APSEC 2014), Jeju, Korea, December 1 - December 4, 2014.
- [B] R. Hebig, T. Ho-Quang, M.R.V. Chaudron, G. Robles, F. Miguel Angel “The Quest for Open Source Projects that Use UML: Mining GitHub”
ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint-Malo, France, October 2 - October 7, 2016.
- [C] T. Ho-Quang, R. Hebig, G. Robles, M.R.V. Chaudron, F. Miguel Angel “Practices and Perceptions of UML Use in Open Source Projects”
Accepted at 39th International Conference on Software Engineering - Software Engineering in Practice Track (ICSE SEIP 2017), Buenos Aires, Argentina, May 20 - May 28, 2017.

Other publications

The following publications were published during my PhD studies, or are currently in submission. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

- [a] H. Osman, M.R.V. Chaudron, P. van der Putten, T. Ho-Quang “Condensing Reverse Engineered Class Diagrams Through Class Name Based Abstraction”
4th World Congress on Information and Communication Technologies (WICT’14), Malacca, Malaysia, December 8 - December 10, 2014.
- [b] D.R. Stikkolorum, T. Ho-Quang, M.R.V. Chaudron “Revealing Students’ UML Class Diagram Modelling Strategies with WebUML and LogViz”
41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Funchal, Madeira, Portugal, August 26 - August 28, 2015.
- [c] D.R. Stikkolorum, T. Ho-Quang, B. Karasneh, M.R.V. Chaudron “Uncovering Students’ Common Difficulties and Strategies During a Class Diagram Design Process: an Online Experiment”
Educators Symposium at ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (EduSymp@MoDELS 2015), Ottawa, Canada, September 29, 2015.
- [d] R. Hebig, T. Ho-Quang, M.R.V. Chaudron, G. Robles, F. Miguel Angel “The Quest for UML in Open Source Projects Initial findings from GitHub”
in Proceedings of the Doctoral Consortium at the 12th International Conference on Open Source Systems (OSS 2016), Göteborg, Sweden, May 30, 2016
- [e] L. Sion, R.Scandariato, K.Yskout, T. Ho-Quang, W. Joosen “Where is my security code? A comparison of four program comprehension techniques”
In submission to 25th IEEE International Conference on Program Comprehension (ICPC 2017)
- [f] R. Jolak, E. Umuhuza, T. Ho-Quang, M.R.V. Chaudron, M.Brambilla “One Marshmallow Now And Two Later: Short- And Long-Term Benefits of UML Modeling”
In submission to the 21st conference on Evaluation and Assessment in Software Engineering Conference (EASE17)
- [g] G. Robles, T. Ho-Quang, R. Hebig, M.R.V. Chaudron, F. Miguel Angel “An extensive collection of UML files in GitHub”
In submission to the Data Showcase of the 14th International Conference on Mining Software Repositories (MSR 2017).

Research Contribution

My contributions to Paper A are study design, data analysis and the majority of paper writing. The tool that was used for extracting image-processing features in the paper was implemented by I. Samelsson and J. Hjaltason. The remaining authors contributed with reviews and improvement suggestions.

Studies reported in Paper B and C were conducted in collaboration with the Grupo de Sistemas y Comunicaciones (GSyC) ¹ at the Universidad Rey Juan Carlos (URJC)². In the two papers, the major effort in classifying UML images and validating the classification results was made by me.

In Paper B, I participated and contributed in designing the study, formulating research questions, analyzing data and discussing results. I wrote a majority of the publication regarding Introduction, Methodology and Threats to validity.

In Paper C, I took the leading role in study design, data collection and data analysis. My major effort in this work consisted of identifying UML images, executing the survey and analyzing the data. In term of paper writing, I wrote the majority of the sections Research Questions, Methodology, Results/Findings and Conclusion.

¹Home page of GSyC - <https://gsyc.urjc.es/>

²Home page of the Universidad Rey Juan Carlos (Madrid, Spain) - <http://www.urjc.es/>

Contents

Abstract	v
Acknowledgment	vii
List of Publications	ix
Personal Contribution	xi
1 Introduction	1
1.1 Research Focus	3
1.1.1 Goals of the PhD study	3
1.1.2 Goals and Outcomes of the Licentiate thesis	5
1.1.3 Research questions of the Licentiate thesis	5
1.2 Background	6
1.2.1 The Unified Modeling Language (UML): a short overview	6
1.2.2 Effort of collecting UML for empirical research	7
1.2.3 UML use and impacts of using UML in software engineering projects	9
1.2.3.1 UML use in industry	9
1.2.3.2 UML in OSS projects	10
1.3 Methodology	11
1.3.1 Constructive research method	11
1.3.2 Empirical methods	11
1.4 Contributions	12
1.4.1 Paper A: Automatic classification of UML class diagrams from images	13
1.4.2 Paper B: The quest for open source projects that use UML: mining GitHub	14
1.4.3 Paper C: Practices and perceptions of UML use in open source projects	15
1.4.4 Answers to research questions	16
1.5 Threats to validity	18
1.6 Future Work	19
1.6.1 Curating the dataset	19
1.6.2 Extending understanding about UML use: success and failure factors	20
1.6.3 Building guidelines for UML use	22
1.6.4 Other directions	23

2	Paper A	25
2.1	Introduction	26
2.2	Related Work	27
2.2.1	Image classification	27
2.2.2	Diagram feature extraction	28
2.3	Research Questions	28
2.4	Approach	28
2.4.1	Overall framework	28
2.4.2	Image processing	29
2.4.3	Feature extraction	30
2.4.3.1	Which features set UML CD apart from other diagrams?	31
2.4.3.2	Extraction features in details	31
2.4.4	UML CD classification	33
2.4.4.1	Choosing the most suitable classification algorithm	33
2.4.4.2	Training classifier	33
2.4.5	Analyse Result	34
2.5	Experiment Description	34
2.5.1	Dataset	34
2.5.2	Evaluation measures	34
2.5.2.1	Features Predictive Performance	34
2.5.2.2	Classification Algorithm Performance	34
2.5.3	Experiment settings	35
2.6	Analysis Of Results	35
2.6.1	RQ1: Influence of features	35
2.6.2	RQ2: Classification algorithms performance	36
2.6.3	RQ3: Set of features Performance	37
2.7	Discussion	38
2.7.1	Image Processing Time	38
2.7.2	Image Processing Features Performance	39
2.7.3	Classification Algorithms	40
2.7.4	Threats to validity	40
2.7.4.1	Threats to Internal Validity	40
2.7.4.2	Threats to External Validity	40
2.7.4.3	Threats to Construct Validity	40
2.8	Conclusions and Future Work	40
3	Paper B	43
3.1	Introduction	44
3.2	Research questions	45
3.3	Related research	46
3.3.1	Use of UML in FOSS	46
3.3.2	Mining	47
3.4	Methodology	47
3.4.1	Occurrence of UML	48
3.4.2	Data Collection	49
3.4.3	UML filters	50
3.4.3.1	Identify UML images	50

3.4.3.2	Identify UML files among .xmi and .uml files	51
3.4.4	Metadata Extraction and Querying	52
3.5	Results	52
3.5.1	RQ1: UML in GitHub projects	53
3.5.2	RQ2: Versions of UML models	53
3.5.3	RQ3: Time of UML model introduction	54
3.5.4	RQ4: Time span of active UML	56
3.5.5	RQ5: Duplicates	58
3.6	Discussion	59
3.7	Threats to validity	62
3.7.1	Threats to construct validity	63
3.7.2	Threats to external validity	63
3.7.3	Threats to conclusion validity	64
3.8	Conclusions	64
4	Paper C	67
4.1	Introduction	68
4.2	Research Question	69
4.3	Related work	70
4.3.1	Modeling in Industry	70
4.3.2	Modeling in Open Source Software	70
4.4	Research Methodology	71
4.4.1	Data Collection	71
4.4.1.1	Obtaining the full list of GitHub projects	71
4.4.1.2	Identifying UML files	71
4.4.1.3	Extracting meta-data	71
4.4.2	Filtering the obtained projects and contributors	72
4.4.2.1	Filtering short-time projects	72
4.4.2.2	Merging duplicate contributors	72
4.4.3	Conducting the survey	72
4.4.3.1	Participant	72
4.4.3.2	Questionnaire	73
4.4.3.3	Sending out the survey	74
4.4.4	Data Analysis	74
4.5	Results/Findings	75
4.5.1	Respondent Demographics	75
4.5.2	Why is UML used?	76
4.5.2.1	What are the motivations to use UML modeling?	76
4.5.2.2	What are the reasons not to use UML in projects?	77
4.5.3	Is UML part of the interaction of contributors?	77
4.5.3.1	Developer's awareness about the existence of UML in their projects	77
4.5.3.2	Are UML models used during communication and team decision making?	78
4.5.3.3	Are modeled designs adopted afterwards, during the implementation phase by teams of OSS contributors?	79
4.5.4	What is the impact/benefit of UML?	80
4.5.4.1	Can UML models support new contributors?	80

4.5.4.2	What are the impacts of using UML in OSS projects?	81
4.5.4.3	Can UML models help to attract new contributors?	82
4.6	Discussions	83
4.6.1	Comparison to Insights to Related Works	83
4.6.2	Implications	84
4.6.2.1	OSS practitioners	84
4.6.2.2	OSS seniors	84
4.6.2.3	Industrial companies	84
4.6.2.4	University teachers	84
4.6.3	Threats to Validity	84
4.7	Conclusion and Future work	85
4.8	Appendix 1. Distribution of survey respondents by countries .	87
4.9	Appendix 2. Distribution of survey respondents by continents .	87

Bibliography	89
---------------------	-----------

Chapter 1

Introduction

In modern software development, software modeling is considered as an essential part of the software architecture and design activity. The Unified Modeling Language (UML) is a graphical language for modeling software systems. The UML was first introduced in 1997 [1]. Since then, it has become the de facto standard for software modeling in industry. The UML is a language but not a method. Its use and impact are different across companies and software systems. To shed some light on the practical use and impacts of using UML, a number of research studies have been conducted such as [2–11]. However, conclusions are diverse and partially contradictory. For example, in a large-scale interview, Petre [11] found that the majority of the interviewees refused to use the UML because of its complexity, lack of formal semantics, inconsistency, and issues of synchronization between different diagrams. However, there are case-studies where UML is actively used and positively impact software system. In a case study at ABB, Anda et al. [2] found out that UML is actively adopted and the UML adoption constitutes in the improvement of softwares traceability, communication within development teams, code-design.

Looking at these results, we are triggered by the questions: Why UML is successfully adopted in some cases but not on the others?, Are there any common success formula for UML adoption?, Are there any guidelines for applying UML in practice? We find out that it is difficult to properly answer the questions because: i) there is a surprising lack of empirical evidence on the use and impacts of using UML [8, 10] and ii) empirical research in the field is generally lacking of replication [12]. As a consequence, existing results are often not generalizable or comparable.

Yet, practical guidelines and underlying success/failure factors of UML use remain invisible. This could lead to the application of UML that does not lead to expected improvement in software quality or in communication between development teams. This, eventually, results in a waste of organization's effort and money for UML adoption.

This Licentiate thesis is a first step toward a PhD thesis that aims at *synthesizing practical guidelines for UML use by systematically collecting and studying UML practices in real-life software systems*.

We are inspired by the concept of business intelligence (BI) [13] to approach this problem. In particular, BI defines 5 stages to gain applicable knowledge

from raw data. They are *raw-data*, *information*, *knowledge*, *expertise* and ultimately, *wisdom*. Similarly, actionable guidelines of UML use can be built on top of empirical data of UML use, through a “mining” process. In the context of this PhD study, “mining software modeling practices” refers to the process of building actionable guidelines for UML use on top of a rich empirical dataset of UML design and its use.

The mining process comprises of three stages on which specific goals of the PhD thesis and Licentiate thesis build: i) Collecting empirical data, ii) Understanding factors that affect to UML use and iii) Building practical guidelines for UML use.

Collecting empirical data. First, evidence-based knowledge is built on top of empirical dataset/corpus. As a prerequisite to build up practical guidelines for UML use, a rich empirical data of UML use needs to be collected. When it comes to data collection, an obvious question is *where to collect the data from*. One option is to collect UML practices from industrial companies. This has pros and cons. The main advantage of industrial UML practices is industry-relevant contexts (including rationale, business decisions) behind UML use. The main drawback lays in data availability. Particularly, it is generally difficult to collect industrial cases where UML is used, because companies consider their design as commercially sensitive information or as a reflection of their state of IT-affairs. This could further limit the study’s replication. An alternative option is to collect UML practices from OSS projects. Data availability and transparency are clearly the main advantage of OSS projects compared with industrial cases. Public access to not only UML file but also other resources such as source code, documentation, issues, commit message, etc. would allow researchers to put the UML file in its usage context. The main challenge is to identify OSS projects that use UML. This is due to the large variety of UML file formats and the lack of support from most open source platforms, such as GitHub, for model versioning. This Licentiate thesis focuses on finding UML practices in OSS projects and takes the challenge of identifying OSS projects that use UML.

Understanding factors that affect UML use. In order to reach the final goal of the PhD thesis, we see the need of building sufficient understanding about UML and its use in software development projects as the intermediate step. Examples of “sufficient understanding” are circumstances where UML is used, aspects that affect the way UML is used and further, factors that make UML use successful or unsuccessful within software projects. To gain such the understanding, UML design should not just be considered as a standalone documentation but rather in its relation with other software resources such as source code, requirements, etc. This is a precondition to study impacts of use of UML on other software elements (e.g. UML models might have impact on source code modularity [8]), and vice versa (e.g. communication habits within teams might affect the way developers construct and use UML). In order to gain such understanding, in-depth analysis needs to be carried out on top of a rich and evidence-based data. Accordingly, having a well-structured data could result in better understanding. Likewise, understanding about UML and how it is used in different projects would allow us to better organize and classify

projects (e.g. by projects domain, habits of using UML, etc.). In this Licentiate thesis, we collect a large set of OSS projects that use UML, then build first observations on the use and impacts of using UML in OSS projects. Questions regarding success or failure of UML use require a certain level of maturity of the data, e.g. UML models are classified by their quality. Therefore, these questions are left for later part of the PhD study.

Building practical guidelines for UML use. Understanding the use and impacts of using UML does not necessarily mean knowing “how to efficiently apply” or “how to efficiently use” UML in real-life situations. A large scale survey by Dobing et al. [3] suggests that the lack of usage guidelines is one of the biggest concerns with UML. We aim to set the final goal of this PhD as building practical and actionable guidelines for UML use, targeting organizations and software practitioners. To this end, the guidelines needs to be built on top of firm understanding about the use of UML, in a large number of real-life software projects. This Licentiate thesis, as a first step toward building the guidelines, aims at providing a number of *implications* for software engineering practitioners, software development teams and teachers. The implications should be drawn from first pieces of understanding on the use and the rationale behind the use of UML in OSS projects. It is worth noting that learning from OSS practices does not mean to ignore or overlook UML use in industrial context. Given the ever-increasing penetration of OSS culture to industrial companies, findings from this Licentiate thesis can be valuable and generalizable to industrial context, as well.

The remainder of this chapter is constructed as follow. Section 1.1 presents the research focus and research questions of the thesis. Section 1.2 provides readers with background of the study. Section 1.3 discusses the methodology. Section 1.4 presents a short summary of each papers and their contribution to the goal of the thesis. Section 1.5 discusses the threats to validity and section 1.6 wraps up this chapter with future works.

1.1 Research Focus

This section frames the research goals of the PhD study into a big picture, and explains the focus of the thesis in this picture.

1.1.1 Goals of the PhD study

This PhD study aims at synthesizing practical guidelines for UML use by systematically collecting and studying UML practices in real-life software systems. The goals of this PhD study are:

- Goal G1. To collect and provide software practitioners with empirical evidence of UML use in real software systems, and
- Goal G2. To understand success/failure factors of UML use in practice, and
- Goal G3. To provide software practitioners and organizations with a practical guideline of using UML.

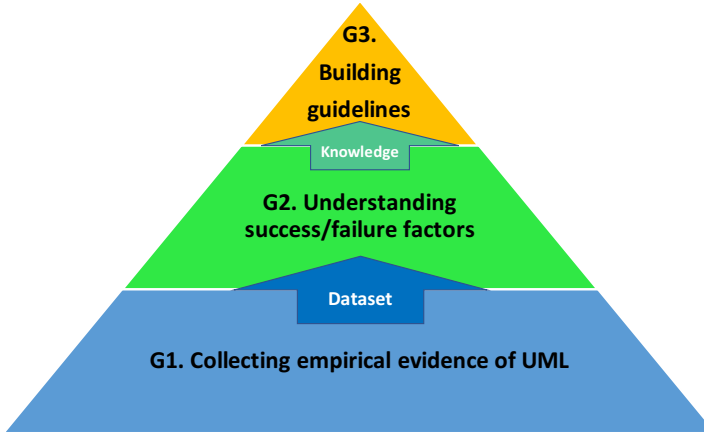


Figure 1.1: Pyramid of goals of the PhD Study

Figure 1.1 shows the relation of the three goals in the context of this PhD study. Goal G1 aims at establishing empirical data of UML use in real software projects. Goal G2 aims at gaining knowledge about UML use and factors that affect the use by quantitatively and qualitatively questioning and analyzing the built data (from G1). Goal G3, in its turn, aims at synthesizing knowledge gained from G2 and G1 in an actionable and practical guideline of UML use, targeting software practitioners and organizations.

The PhD study is designed in a replicable manner. Accordingly, data obtained from G1 is open for public access. Methods are described step by step, limitations and threats to validity are carefully discussed in the chapters.

The PhD study is designed and conducted incrementally. In particular, the main goals are divided into sub goals/steps, based on the maturity of the study (Figure 1.2).

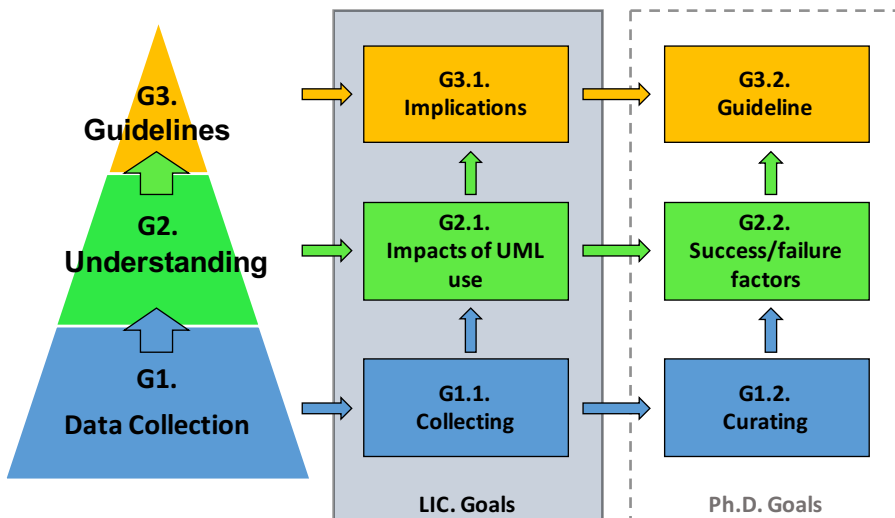


Figure 1.2: Licentiate goals in the relation with of PhD goals

We split **G1** into two sub goals:

- **G1.1** (Data Collection) aims to systematically collect a large data set of OSS projects that use UML.
- **G1.2** (Data Curation) focuses on classifying and organizing the projects in a systematic manner.

We split **G2** into two sub goals:

- **G2.1** (Exploratory understanding) aims at exploratory understanding of the use and impact of using UML in the projects.
- **G2.2** (Eliciting success/failure factors) makes a step further by eliciting the success and failure factors of UML use, taking into account the differences of the projects.

G3 is split into two subgoals:

- **G3.1** (Implications) Providing implications on the use of UML based on first pieces of understanding gained from **G2.1**.
- **G3.2** (Guideline) provides complete guideline for UML use.

1.1.2 Goals and Outcomes of the Licentiate thesis

The main goals of this Licentiate thesis are **G1.1**, **G2.1** and **G3.1**. Expected outcomes of the Licentiate are as follow:

- Goal **G1.1**. Researchers in the area of modeling in software engineering have performed some efforts to collect examples of models and of projects that use modelling. However, the results are often limited [14] due to the so far limited success in identifying open source projects with UML. **G1.1** aims at combining various techniques to solve this issue. The outcome of **G1.1** is a list of OSS projects that use UML and a systematic process to identify them.
- Goal **G2.1**. This goal aims at answering simple research questions regarding UML use by using the list obtained from **G1.1**. The expected outcome of **G2.1** is quantitative and qualitative analyses of UML use in OSS projects. Examples of such basic analyses are time period when UML is introduced, whether UML is updated and rationale behind the use of UML and impacts of using UML.
- Goal **G3.1**. The outcome of this should be a list of implications about UML use, targeting software practitioners in OSS and might possibly in industrial settings.

1.1.3 Research questions of the Licentiate thesis

To reach the goals of the Licentiate thesis, we formulate the following research questions:

- **RQ1**. How can we identify UML models in OSS projects?

- **RQ2.** How is UML used in OSS projects?
- **RQ3.** Why is UML used in OSS projects? What are impacts of using UML in OSS projects?
- **RQ4.** Can we draw implication for UML use?

Among the research questions, **RQ1** aims for **G1.1**, **RQ2** and **RQ3** target **G2.1**, and **RQ4** captures **G3.1**.

1.2 Background

This thesis takes the UML as the main research subject. Section 1.2.1 provides readers with a short overview about the emergence of the UML.

An important part of this Licentiate thesis is collecting empirical data of UML and its use. Section 1.2.2 summarizes some efforts performed by other researchers to collect UML for empirical research.

Section 1.2.3 aims at providing an overall picture of UML use in industry and OSS by summarizing previous empirical studies in the field.

1.2.1 The Unified Modeling Language (UML): a short overview

The evolution of UML is described in detail by Kobryn [15]. The UML emerged from the competition in creating notations for object-oriented design by Booch [16], Rumbaugh et al. [17], Jacobson et al. [18] and other researchers in the early nineties of the 20th century. The UML was developed by Rational Software in 1996. The 1.0 version was proposed in January 1997 and officially adopted by the Object Management Group (OMG) later that year. Since then, the UML has undergone many revisions, with UML 2.0 released in 2005 and most recently UML 2.5 in June 2015.

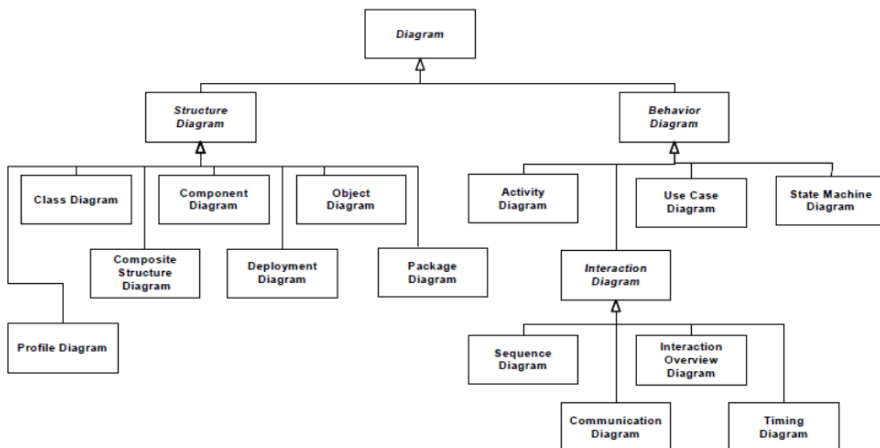


Figure 1.3: Taxonomy of UML diagrams

UML 2.5 has 14 diagram types. They are divided into two categories, namely structure- and behavior- diagrams. Structure diagrams show the static structure of systems, while behavior diagrams show the dynamic behavior of systems, including their methods, collaborations, activities, and state histories. Figure 1.3 shows a taxonomy of UML diagrams.

In the OMG's view, "modeling is the designing of software applications before coding." The OMG promotes model-driven architecture as the approach in which UML models of the software architecture are developed prior to implementation. However, as the UML is a language and not a method, there are much more ways of using it, e.g. for reverse engineering, refactoring, documenting an existing system, etc. In addition, the UML is a methodology-independent language, one could use it in different software development processes (e.g. when planning, analyzing requirements) and in different software development methods (e.g. water fall or agile approach). In this thesis, the UML is considered in all contexts where it is used, not limited to model-driven per se. Figure 1.4 shows examples of three different types of UML diagrams.

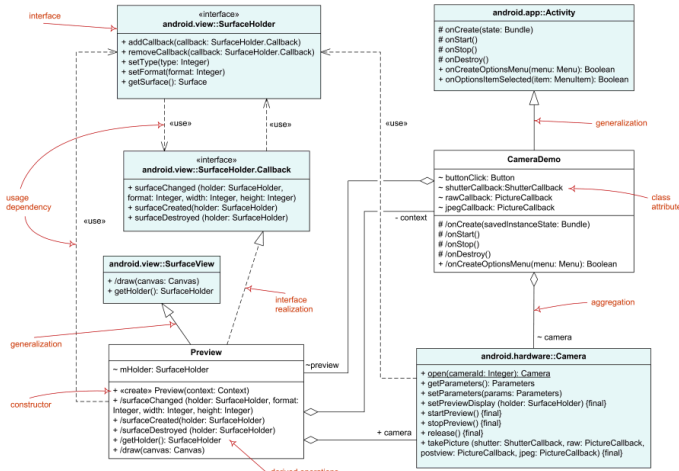
1.2.2 Effort of collecting UML for empirical research

Researchers in the area of modeling in software engineering have performed some efforts to collect examples of models and of projects that use modeling. However the results are often limited [14]. For example, the Repository for Model Driven Development (ReMoDD) [19] is an initiative driven by an international consortium of leading researchers in the field of modeling. Nevertheless its content is growing at a low rate: after 9 years (summer 2016) it contains around 81 models. Industrial projects are very reluctant to share models because they believe these reflect key intellectual property and/or insight into their state of IT-affairs.

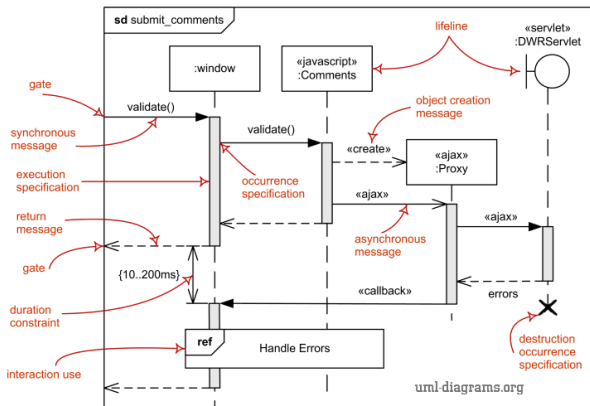
Karasneh et al. [20] use a crawling approach to automatically collect and fill an online repository with so far more than 700 model images. This work focus on the models only and do not take their project context into account. Further, Karasneh et al. do not distinguish between models that stem from actual software development projects and models that are created for other reasons, e.g. teaching. An index of existing model repositories can be found online [14]. However, in addition to their small size, these repositories seldom include other artifacts than the models, making it impossible to study the models in the environment of actual projects.

Further, there are some works addressing small numbers of case studies of modeling in open source projects. Yatani et al. [21] studied the models usage in Ubuntu development by interviewing 9 developers. They found that models are forward designs that are rarely updated. Osman et al. [22] investigated 10 case studies of open source projects from Google-code and SourceForge that use UML. They focused on the relation between classes in the diagrams compared to classes in the code. They find only seldom cases where models are updated.

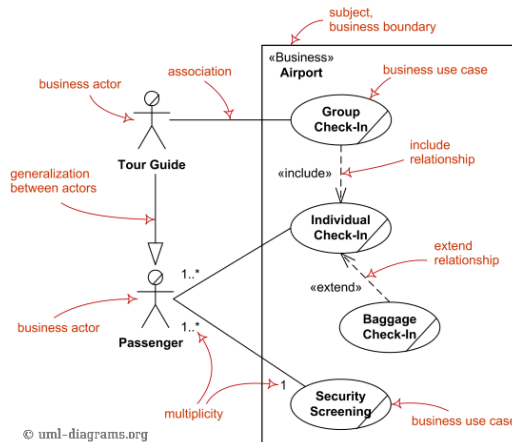
Finally, there are three works that actually approach a quantitative investigation of models in open source projects. Chung et al. [23] questioned 230 contributors from 40 open source projects for their use of sketches and found that participants tend to not update these sketches. A study that focuses on software architecture documentation in open source projects was performed



(a) Class diagram



(b) Sequence diagram



(c) Use-Case diagram

Figure 1.4: Examples of UML diagram - Source: <http://www.uml-diagrams.org/>

by Ding et al. [24]. They manually studied 2 000 projects from SourceForge, Google code, GitHub, and Tigris. Amongst those projects that used such documentations they identified 19 projects that actually use UML.

1.2.3 UML use and impacts of using UML in software engineering projects

In this section, we provide readers with an overall picture of UML use in industry and in OSS projects by summarizing related works.

1.2.3.1 UML use in industry

UML is widely studied in industry, however, there are real needs for more experiments and case studies Budgen et al. [10], in their systematic review, identified 49 empirical studies of UML published up to the end of 2008. Among them, 12 papers were about UML metrics, 14.5 about model comprehension, and 7.5 about model quality (half points indicate papers with more than one focus). Only 2 papers addressed UML adoption, i.e. by Anda et al. [2] and Grossman et al. [5]. These two studies identify a range of benefits and drawbacks associated with the adoption of UML. They particularly highlight the need for further research relating to the UML and its adoption as well as the need for, and importance of, an adequate level of training.

Another systematic review conducted by Fernández-Sáez et al. [25] identified 38 papers (published up to the end of 2010) that report 63 empirical studies of UML use in software maintenance processes. Only 3 of them are case-studies. Most research (60 empirical studies) concerns the maintainability and comprehensibility of the UML diagrams themselves. The authors conclude that there is a need for more experiments and case studies to be performed in industrial contexts.

Nugroho and Chaudron [8] also argue that Despite the fact that UML is widely used in practice, little is known about how UML is actually used.

UML adoption and its impacts are still under discussion Modeling has been widely studied in industry, in particular in several survey studies. Torchiano et al. [26] found that models help to improve design and documentation. However, they also found that model usage is connected to extra effort, especially due to a lack of supporting tooling. Forward et al. [27] found that models are primarily used for design and documentation, while code generation is rather seldom. Gorschek et al. [28] focused on a different population, which are programmers, partially working in industry and open source. Within their sample design models are not used very extensively. However, models and UML are found to be used mainly for communication purposes. Further, they report on a higher use of models for less experienced programmers.

Dobing and Parsons [3] report on a survey of 171 UML users (plus 11 who use UML components within another OO methodology). The authors found that "only class diagrams are being used regularly...". The majority of respondents found that all aspects of UML are useful for most projects. The authors also suggest that complexity and lack of usage guidelines are the biggest concerns with UML.

Lange et al. [6] conducted a web-based survey in 14 industrial companies. On the basis of the responses, they identified four main classes of problems encountered: scattered information (e.g., design choices dependencies); incompleteness, disproportion (more detail for some parts than others), inconsistency.

Nugroho et al. [29] conducted a survey among 48 professional developers (from 10 different countries) about their perception toward correspondence between source code and design. Respondents identify incompleteness of the UML design as the most prominent factor that often forces them to deviate from a UML design.

Scanniello [9] conducted a survey at 22 companies regarding the use of UML in software development and maintenance. Survey responses show that the majority of the companies (20) use UML in the analysis and design phases.

Besides these big surveys, *case studies* were performed in order to investigate the impact of the modeling/UML usage. For example, Baker et al. [4] found an increase of productivity when using UML in Motorola. Also Nugroho et al. [30] investigated an industrial case study and found that UML usage has the potential to reduce the defect density and, thus, increase the quality of software. Just as in the case described by Kuhn et al. [31], most of the case studies draw a picture of model use, where models are actually artifacts that are produced and consumed by different people. Anda et al. [2] reports a case study in ABB. This paper found anecdotal advantages of modeling such as improved traceability. This paper also pointed to potential trade-offs, such as time spending to integrate legacy code with models and organizational changes needed to accommodate modeling.

Petre et al. [11] reported a series of interviews conducted over 2 years with more than 50 practicing professional software developers. The author found out that the majority of interviewees do not use UML, and those who do use it tend to do so selectively and often informally.

Last but not least, Dzidek et al. [7] performed a controlled experiment to investigate the influences of the use of UML to maintenance task (20 professional developers). The result of the experiment shows a positive influence of the presence of UML for maintainers. UML also helps novice developers to produce code of better quality.

1.2.3.2 UML in OSS projects

Much less study has been done on UML use in OSS. One reason for this is the challenge to actually find cases that can be studied. For example, Badreddin et al. studied 20 projects, without finding UML and concluded that it is barely used in open source [32]. Similarly, Ding et al. [24] found only 19 projects with UML when manually studying 2000 open source projects.

There are several investigations of single or very small numbers of cases of open source projects that use UML, e.g. by Yatani et al. [21], who found that models are used to describe system designs, but are rarely updated. Osman et al. [22] studied to what extent classes in the diagrams are implemented in the code. Finally, Kazman et al. [33] investigate the Hadoop Distributed File System to learn how documentation impacts communication and commit behavior in the open source system. There are some studies that approach model use in open source with a quantitative perspective, studying large numbers of projects.

For example, to study the use of sketches, Chung et al. [23] collected insights from 230 persons contributing to 40 open source projects. Finally, Langer et al. [34] studied the lifespan of 121 enterprise architect models in open source projects.

1.3 Methodology

In this thesis, we employ constructive research and empirical research methods to achieve the three goals **G1.1**, **G2.1** and **G3.1**. Table 1.1 shows the presence of the two methods in papers **A**, **B** and **C**. Sections 1.3.1 and 1.3.2 discuss the use of the methods across the papers. A more-detailed discussion of the research strategies is found in the included papers.

Table 1.1: Summary of research method used in papers A, B and C

	Paper A	Paper B	PaperC
Constructive method	X	X	
Empirical method		X	X
Survey study			

1.3.1 Constructive research method

According to Crnkovic [35], the constructive research method implies building of an artifact that solves a domain specific problem in order to create knowledge about how the problem can be solved (or understood, explained or modeled) in principle. Artifacts such as models, diagrams, plans, organization charts, system designs, algorithms and artificial languages and software development methods are typical constructs used in constructive research.

In the context of the Licentiate thesis, we follow the constructive research approach to construct a process to systematically identify UML files in OSS projects, and further, to identify OSS projects that use UML. Outcome of the research is a list of OSS projects that use UML and a systematic process for identifying them.

The constructive method is applied in the two papers A and B. In particular, paper A propose an automated method for automatically identifying UML class diagram images among ordinary images. Paper B combines the method described in Paper A with other techniques (e.g. GHTorrent [36], CVSanaly [37]) in a complete process to identify UML diagram files in 10% of GitHub non-forked repositories (about 1.2 million projects). Paper C applies the process described in paper B to obtain UML files from all GitHub non-forked repositories (around 12 million projects).

1.3.2 Empirical methods

Empirical study aims at gaining knowledge by means of direct and indirect observation or experience. We see the need for empirical studies in order to increase our understanding of the use of UML and its impacts within OSS projects.

Among common empirical methods are survey studies. These are used to identify the characteristics of a broad population of individuals. They are used to obtain a representative picture of a larger population [38]. This fits goals **G2.1** and **G3.1** of the study, as we want to create a broad picture of the use and impacts of using UML in OSS projects.

Survey research is most closely associated with the use of questionnaires for data collection. However, survey research can also be conducted by using structure interviews, or data logging techniques [39]. We use different data collection methods for the survey studies presented in paper B and C.

In paper B, we attempt to address RQ2. In paper B, 3 295 OSS projects that use UML were identified. Subsequently, meta-data of the projects was collected. Quantitative analysis of the data allows us to understand the use of UML in OSS projects at the descriptive level, e.g. whether UML models are used/updated, and if so, active time of UML models, etc.

In paper C, qualitative and quantitative data was collected using a survey questionnaire. While quantitative data gives us an overview of OSS developers opinion regarding purposes and impacts of using UML, qualitative data provides us with the rationale behind their choices. We combine quantitative and qualitative analysis in this paper is necessary for us to draw implications concerning UML use.

1.4 Contributions

In this section, we firstly summarize and state the main contributions of the three papers on which this Licentiate thesis builds in sections 1.4.1, 1.4.2 and 1.4.3. Research questions are subsequently answered in section 1.4.4.

Figure 1.2 offers an overview of the contributions of the three papers to the main goals and research questions of the Licentiate thesis.

Table 1.2: Contributions of papers to specific goals and research questions

Stage	Data collection	Understanding		Guidelines
Lic. goal	G1.1	G2.1		G3.1
RQ	RQ1	RQ2	RQ3	RQ4
Paper A	Data collection method	—	—	—
Paper B	Identified 3 295 projects that use UML (from 10% GitHub repo)	Is UML used? updated? UML introduction time? What is active time?	—	—
Paper C	Identified 24 797 projects that use UML (from 100% GitHub repo)		Purposes/Impacts of using UML? Is UML helpful? Why?	Implications

1.4.1 Paper A: Automatic classification of UML class diagrams from images

The UML is a graphical modeling language. Therefore, it is common to store UML models in graphical file formats such as .png, .jpg, etc. While there is a need of collecting UML models for empirical research, current research methods often lack the systematical identification of images that represent UML diagrams.

Paper A presents an automated classification method for images that represent UML class diagram (UML CD). Building the classification would on one hand help improving the data collection process of the existing UML repositories, such as the one by Bilal et al. [20]. On the other hand, and more importantly, this could further open up the possibility to automate the UML crawling process and to build a larger collection of UML class diagrams.

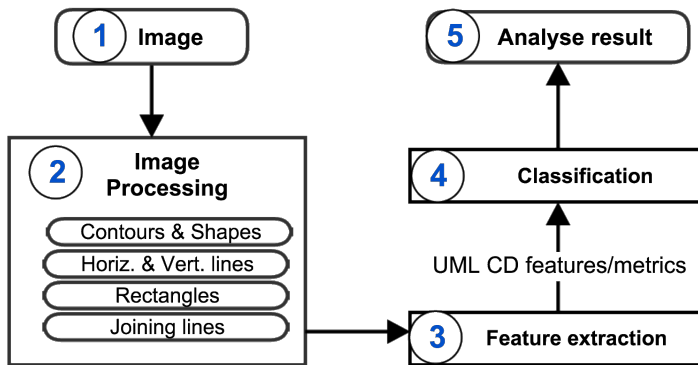


Figure 1.5: Overall processes of paper A

To build the classifier, we use a combination of image processing and machine learning. Figure 1.5 shows the processes of the paper A (this is cloned from Figure 2.1 in the Chapter 2). We first introduced 23 features that capture statistical and geometric characteristics of UML class diagrams. A tool that extracts these features from images was built accordingly. Finally we employed 6 well-known classification algorithms, including Decision Table, Random Forest, Support Vector Machine, Logistic Regression, REP-Tree and J48 Decision Tree [40], to build the classifier. The classification performance of the defined image processing features and algorithms was assessed in terms of classifying accuracy and robustness via several metrics, i.e. information gain, specificity and sensitivity. A dataset of 1 300 different images was collected from the Internet through Google Image Search (750 UML CD images and 750 non-UML CD images) for training and testing the classifier¹. This dataset was shared as a benchmark.

The main contributions of the paper are:

- [a] A machine learning method to build an automated classifier of UML CD images. We find out that our method could reach 95.9% and 91.4%

¹Image set - <http://bitly.com/dtsUMLClassifier>

(respectively) of correct classification of input images for UML CD and non-UML CD.

- [b] Evaluation of classification performance of six algorithms on different feature-sets in terms of classifying accuracy and robustness.
- [c] A dataset of 1 300 images was shared as a benchmark for other classification methods.

Overall, paper A encourages us that it is possible to automate the process of identifying images that contain UML notations with high accuracy. This is a prerequisite to build up a process to identify UML models from OSS projects (Goal 1.1). This work is extending in different ways: Using the same approach to create classifiers for sequence diagram, use-case diagram images.; Involving text-recognition to enhance classification performance.

1.4.2 Paper B: The quest for open source projects that use UML: mining GitHub

Little is known about UML use in Free/Open Source Software (FOSS) projects. This is due to the so far limited success in identifying open source projects that use UML. The lack of available data is the reason why so far no answers could be given to different basic questions on the amount of UML files in open source projects, the time span during which models are created or updated during the open source project, or the question which of the projects contributors do create models.

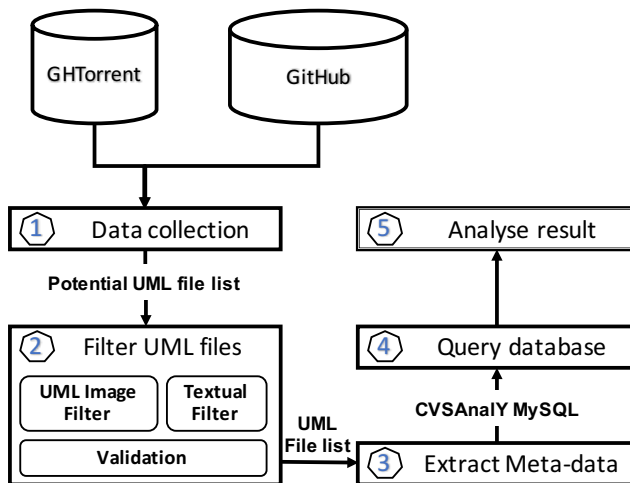


Figure 1.6: Overall processes of paper B

Paper B contributes to this body of knowledge by a five steps approach. Figure 1.6 shows these steps in a sequential order (this is cloned from Figure 3.1 in the Chapter 3). First, we combined different technologies, including the classifier in Paper A, to form a semi-automated process for collecting UML models stored in images, .xmi, and .uml files from over 1.2 millions GitHub

projects (10% of the whole) (from Step 1 - 2). Secondly, from Step 3 - 5, we collected meta-data of the projects that used UML and quantitatively analyzed the data to address the following research questions:

- Are there GitHub projects that use UML? Which are these projects?
- Are there GitHub projects in which the UML models are also updated?
- When in the project are new UML models introduced?
- What is the time span of active UML creation and modification?
- Are UML models copied across projects?

The main contributions of the paper are:

- [a] A semi-automated process for collecting UML models stored in over 1.2 millions GitHub projects.
- [b] A list of 21 316 UML diagrams from 3 295 GitHub projects and their meta-data. This is the first time the modeling community can establish a corpus comparable to collections already exist for source code only, such as QualitasCorpus ².
- [c] Quantitatively answering a number of research questions regarding the use of UML in OSS projects.

1.4.3 Paper C: Practices and perceptions of UML use in open source projects

Paper B reveals some facts about the use of UML based on quantitative analysis on meta-data of OSS projects. These facts trigger us to discover the rationales behind the use of UML and impacts of using UML in OSS projects.

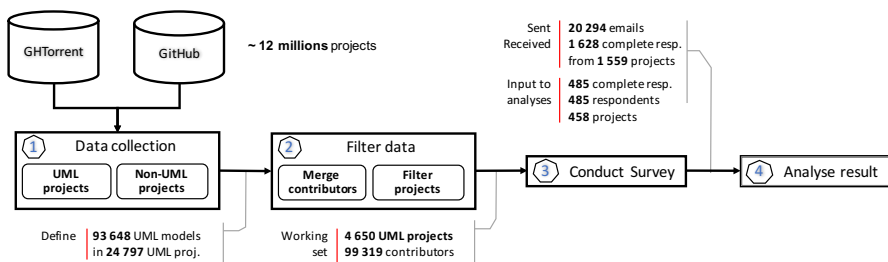


Figure 1.7: Overall processes of paper C

Paper C represents our large-scale survey on software developers of GitHub projects that use UML. The overall process is shown in Figure 1.7 this is cloned from Figure 4.1 in Chapter 4). We first extended the data collection method presented in paper B to 100% of all GitHub repositories (over 12.8 millions repos). This resulted in a dataset of over 93 000 UML models from over 24 000 GitHub projects. A number of filters was then applied in order to filter out those projects that were suspected to be “toy”, e.g. active less than 6 months.

²Qualitas Corpus home page - <http://qualitascorpus.com/>

For the survey, we collected up to three persons per project, targeting persons who had introduced UML models (1UC), persons who had updated UML models (UC) and persons who had not committed UML models (NUC), to send our survey to. As a result, we received 485 survey responses from 458 GitHub projects. Analyzing the responses allows us to answer the following main research questions:

- Why is UML used in OSS projects?
- Is UML part of the interaction of (a team of) contributors?
- What is impact/benefit of UML?

Findings from survey responses allow us to not only answer the research questions, but also to compare UML use and impacts of using UML in OSS projects and related empirical research. Furthermore, a number of recommendations/implications on the use of UML were given to different UML practitioners.

The main contributions of this paper are:

- [a] A large set of OSS projects that use UML, and
- [b] Insights from a large scale survey of OSS developers that use UML.

1.4.4 Answers to research questions

Answer to RQ1. “*How can we identify UML models in OSS projects?*”

Answer for **RQ1**: Paper A presents a method for automatically identifying UML class diagram images in any image set. Paper B demonstrates one way to systematically identify UML models in OSS projects, by joining different technologies, i.e. mining software repositories, text-based search and the technique proposed in paper A. In paper C, the data collection method described in paper B is applied to obtain UML models from all GitHub repositories. We publish the replication package of our approach online so that other researchers can replicate these steps ^a. There might be alternative ways to identify UML models, however to our best knowledge, this is the first time the modeling community can establish a corpus comparable to collections already exist for source code only, such as Qualitas Corpus.

^aReplication package: <http://oss.models-db.com/>

Answer to RQ2. “*How UML is used in OSS projects?*”

Findings from paper B help us to answer the research question **RQ2** as follow:

Answer for **RQ2**: The majority of models are never updated. Those projects that do update their models, do this regularly. Models can be introduced during all possible phases in the lifespan of an OSS project. Nevertheless a peak of model introduction is during the first 10% of the duration of projects. A few projects are active with UML during their whole lifetime. Most projects work very shortly on UML, usually at the beginning. We found that 12% of the distinct models occurred several times. Duplicates are in average spread across 1.88 projects.

Answer to RQ3. “*Why is UML used in OSS projects? What are impacts of using UML in OSS projects?*”

Paper C helps us to answer research question **RQ3** as follow:

Answer for **RQ3** (1/2) - Why is UML used in OSS projects?
The majority of UML models are intended for creating software designs and documenting software systems. NUCs use UML models to comprehend a system and to communicate with team members. Within most software teams/projects, UML models are used for communication, making architecture decisions and for mentoring. UML designs are adopted in most cases during the implementation phase. Most often, these designs are implemented by a group of 2 - 5 persons, including the one who created the designs.

Answer for **RQ3** (2/2) - What are the impacts of using UML in OSS projects?
The survey responses suggest that UML is helpful for new contributors to get up to speed. However, UML does not seem to have the potential to attract new contributors. One third of the respondents reported changes of the working routine due to UML, mainly in the planning phase, the development process and in communication. Most of the reported changes can be considered positive.

Answer to RQ4. “*Can we draw implications for UML use?*”

Findings from the survey responses in paper C allow us to answer research question **RQ4** as follow:

Answer for **RQ4**: By comparing UML use and impacts of using UML in OSS (from paper C) and other settings (such as industry, from related work), we were able to see the components where one side could learn from others and vice versa. In particular:

- To OSS practitioners: Use UML to coordinate team-work!
- To OSS seniors: Provide UML to support junior peers!
- To Industrial companies: Let’s adopt team-modeling!
- To University teachers: Promote consumption of UML models as first experiences when learning UML!

1.5 Threats to validity

In this section, we give an overview of the threats to the validity of the results of this thesis, i.e. the answers to research question **RQ1** to **RQ4**. Detailed threats to validity of included publications are discussed in their corresponding chapters, from Chapter 2 to Chapter 4.

Research question **RQ1** aims to establish a process for identifying UML models from OSS repositories. We use a constructive method to build the process. There were a number of threats to construct validity that might cause the loss of UML files when performing the data collection process. First, our collection method, which made use of a number of heuristic filters, might overlook potential UML files which are not complying with searching terms and file-type list. Second, limitations of the materials that were used to collect data could probably cause the loss of potential UML models (false-negatives) or the inclusion of files that do not actually contain UML (false-positives) Examples of such the limitations are: out-dated GHTorrent SQL dump, incorrect detection of UML images and the limit of 5000 hits/hour of GitHub API. We partly mitigate these threats by adding a manual check (validation) to the process.

Research question **RQ2** is intended to achieve understanding of the way UML is used in OSS projects. Answer to **RQ2** is drawn from a quantitative analysis on a set of 21 316 UML models from 3 295 (from 10% of all GitHub repositories). There are threats that could affect to the construct validity of this answer. In particular, the loss of potential UML files might affect to this analysis in the sense that it could make us underestimate the number of projects with UML models and the number of UML models. Being aware about this, we focus on getting a descriptive overview of various aspects of the use of UML in GitHub projects and avoid giving statistic conclusion. We expect no systematic bias concerning the aspects that we investigated.

Research question **RQ3** aims at revealing the rationales behind the use of UML and impacts of using UML in OSS projects. Answer to this question is drawn from survey responses of 485 developers in 458 different GitHub projects that use UML. There is a threat to internal validity. We focus on projects that do use UML only, to ensure that questioned developers have the experience of working in a project with UML. To ensure those persons that prefer to not use UML are not underrepresented, we sent the questionnaire not just to persons who manipulated UML, but also to contributors who did not change or introduce UML files (NUCs). Therefore, we believe that our findings still provide valuable insights.

Research question **RQ4** aims to provide implications for UML use. Answer to this research question is built on understanding about UML use (**RQ2**) and impacts of using UML (**RQ3**). Therefore, we could expect threats to validity from both answers to **RQ2** and **RQ3**. However, given these threats are mitigated when answering **RQ2** and **RQ3**, we believe that the posed implications are still valid and significant.

Finally, answers to all research questions have threats to external validity. In particular, data in this research was only taken from GitHub, but not other OSS hosts/platforms such as SourceForge, Google Code, etc. As they differ to each other in terms of size, functionality, users and users behaviors, there is a threat that answers for questions **RQ1**, **RQ2**, **RQ3** and **RQ4** might not be

generalized to other platforms. It is possible that UML is used in a different manner within projects at other platforms. However, as GitHub is one of the biggest player in the field, we strongly believe that our investigation gives valuable insights to a majority of the OSS community.

1.6 Future Work

In section 1.6.1, 1.6.2 and 1.6.3, we discuss possible extensions of current findings towards the pyramid of goals of the PhD study. We suggest directions that we might not explore ourselves because of scope of this PhD study, but now become possible with our dataset in section 1.6.4.

1.6.1 Curating the dataset

In this Licentiate thesis, we achieved Goal 1.1 by establishing a dataset of over 24 000 GitHub projects that use UML and a semi-automated process for identifying them. The dataset allows us to perform different studies regarding UML use in OSS projects (as in Paper B and C).

One lesson learned from performing paper B and C is that the projects in the dataset differ from each other in terms of size, active time, number of contributors, quality of UML models as well as source code. This is due to the fact that GitHub is an open space where anyone could create their own projects for very different purposes. Therefore, it will be necessary to curate our dataset in the future. This can be done by several ways.

Using image processing techniques. A large amount of UML models in our dataset are stored in image formats (57 822 models/93 596 models - 61.7%). These models will probably consume the majority of manual effort when curating the dataset. We believe that image recognition techniques will improve in future and will help automate this task. For example, text recognition could be used to detect names and contents of different elements/components of UML diagrams. Image processing features extracted from UML diagram images can be used for classifying: *reverse engineered diagrams* vs *forward design diagrams*; *layered* layout vs *non-layered* layout (of diagrams), etc.

Extending the dataset. We plan to extend the dataset in the future by collecting all file types that might contain UML models, i.e. tool-specific file types such as .plantuml, .argo, .dia and documentation files such as .pdf, .docx, .doc. Similarly, software models that are not conforming the UML standard, such as SysML, Capella models can be added to the dataset.

The dataset can also be extended by involving more industrial cases. Currently, we can see in our dataset a number of industrial projects such as the ones by HP Helion³ or Azure⁴. These cases will allow us to see how UML is used in a mixed settings of OSS and industrial culture.

³<https://github.com/hphelion>

⁴<https://github.com/ashimaabrol/azure-content>

Enriching the dataset by annotations. It is not just future extensions that will make the dataset more valuable, but also annotations that can bring significant value to the dataset. Annotations can be made at projects and models levels. For example, at the project level, annotations about “project license”, “business domain”, the “goals of project when using models” (for design or documentation), “general impacts of using UML” can be employed.

At the models level, annotations on the type, layout style of the UML model, tools that was used to generate the model, general role of the model, quality of the model, etc. could be very beneficial.

Manual annotating is known as a time-consuming and prone-to-human-error process. In the future, we aim at building automated taggers for this task, starting from basic annotations such as type and layout-style of UML models.

1.6.2 Extending understanding about UML use: success and failure factors

Within this Licentiate thesis, we achieved goal G2.1 by performing quantitative and qualitative analyses on the use of UML and impact of using UML in GitHub projects. As a step further, goal G2.2 aims at eliciting success and failure factors of UML use. Toward this goal, we would think of several research possibilities as follow.

Impacts of UML. Paper C reveals impacts of UML to new OSS contributors and changes in working styles. However, we were unable to see the how UML models affect the actual implementation of the system. Therefore, a possible next step is to study in more general how the use of UML modeling impacts the code structure and whether improvements in software quality and productivity can be observed when UML is introduced.

Research has been conducted in this direction, such as the one by Karasneh et al. [41]. They investigated if anti-patterns are propagated from models to the code, by comparing anti-patterns found in UML models and source code of 10 OSS projects. Our dataset will probably bring more quantitative aspects to research in the field.

UML use. Paper B and C discover some patterns of using UML as well as the developer’s rationale behind the UML use. These findings can be extended in several ways. One way is to conduct follow-up interviews to gain indeed understanding about specific survey response(s). For example, it is interesting to know the communication methods in which UML models were discussed, the CASE tools that were used.

The other extension is to “learn” characteristics of UML model that were successfully used in OSS projects. For example, what model layouts developers use and what average size models have. “Learning” can be done via both manual observations and machine learning.

In addition, studying UML that occurs in images can also deliver hints on needs that OSS developers have for visual highlighting strategies. In particular, during manual check of the image set, we have seen diagrams that were colored and highlighted. Those diagrams might possibly be “important ones” as well.

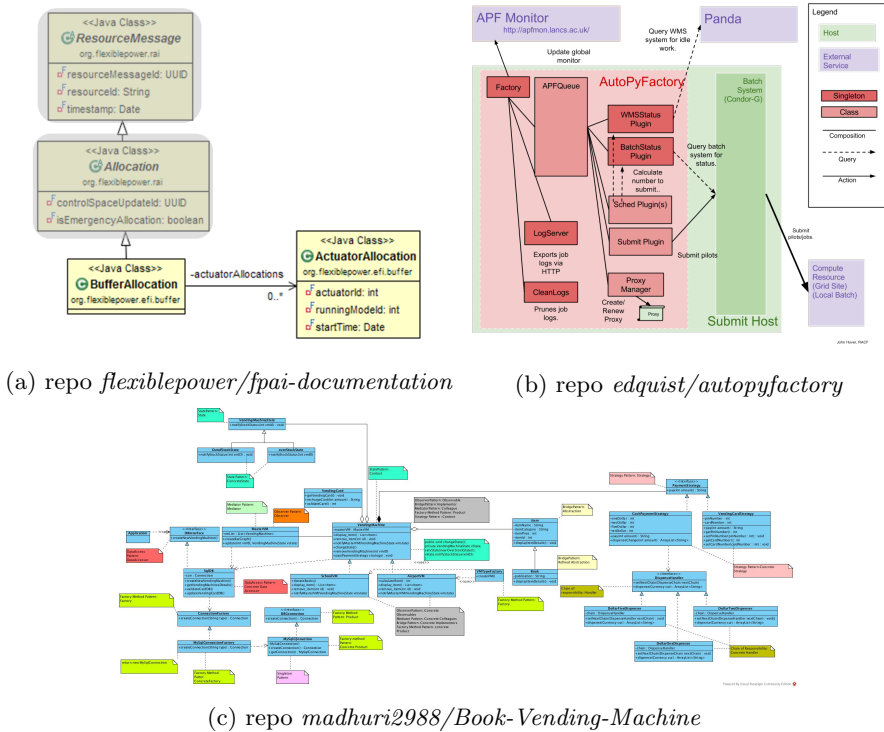


Figure 1.8: Examples of highlighted UML diagrams - Source: GitHub repos

Figure 1.8 shows examples of three highlighted diagrams from three GitHub projects.

Moreover, due to the availability of UML models (and projects that they belong to), we can be able to study how UML models and other artifacts (such as source code, bug-reports, issues, etc.) related to each other. For example, we know from the survey responses in paper C that UML models are (partly) adopted during implementation phase but we do not know how strictly models are implemented, or to what extent models abstract the code.

Assessment of quality of UML models. One aspect that can affect UML use/adoption is the quality of UML models. This is the case in one of the surveyed projects in Paper C. The founder of the GitHub project answered “*I feel that it depends on two things: ... and how elegantly and interesting the models was structured*” as for reason(s) why UML models attracted new contributors to his/her project. We see a need for evaluating the quality of UML model and studying its impacts to UML use in software projects.

Evaluating quality of UML models could allow us to classify UML models by different quality aspects. Furthermore, knowing the quality of a UML model, we will be able to provide recommendations on how to improve the model. At the project level, understanding the quality of UML models of a project could enable to identify the need for actions for quality improvement.

One way to evaluate quality of UML models is to use quality models such as [42–46]. Figure 1.9 shows the relationship between characteristics of quality

of UML and various software metrics and rules proposed by Lange et al. [42]. Accordingly, communicativeness of a UML model can be measure by calculating the depth of inheritance tree (DIT) of the model.

Metrics and Rules	Modularity	Complexity	Completeness	Consistency	Communicativeness	Self-Descriptiveness	Detailedness	Balance	Conciseness	Esthetics	Correspondence
Dynamicity		✓	✓					✓			
Ratios	✓		✓				✓	✓	✓		
DIT	✓	✓			✓		✓	✓	✓		
Coupling	✓						✓				
Cohesion	✓	✓					✓				
Class Complexity							✓				
NCU	✓	✓					✓	✓	✓		
NUC	✓	✓					✓	✓	✓		
Fan-In	✓						✓				
Fan-Out	✓						✓				
Naming Conventions						✓	✓				
Design Patterns	✓		✓			✓	✓	✓			
Layout-Guidelines							✓			✓	
Multi defs.				✓			✓				
ID Coverage			✓				✓				
Message needs Method			✓	✓			✓				
Code Matching			✓				✓				✓
Comment						✓	✓				

Figure 1.9: Mapping of model quality characteristics and software metrics. Source: Table 5, paper [42]

1.6.3 Building guidelines for UML use

In paper C, we provide implications for UML use, targeted software practitioners and educators. In the future, we aim at building guidelines which have the following properties:

Content. Guidelines will be provided to specific subjects based on two sources of understanding: i) understanding about common practices of UML use/adoption observed from a large number of software projects; and ii) understanding about the subjects (individuals or teams/companies) in terms of current use of UML, expected level of UML adoption. The guidelines are expected to cover:

- Methods to assess current status of UML use within a software development team/project. This will be provided in form of a check list or an analytical tool;
- Common mistakes on applying UML;
- Success stories of applying UML in a software projects;

Characteristics. First, the guidelines should be *actionable*. Each guideline should contain the following parts: i) Targeted subjects (individuals or teams/companies); ii) Targeted context and problem; iii) Expected outcomes; iv) Guideline in action; v) Common mistakes; vi) Examples from real-life projects.

Second, guidelines should be *evidence-based*. All guidelines can be traced back to empirical studies and real-life software projects that support them.

1.6.4 Other directions

Our dataset comprises a large number of UML models and meta-data of the projects that they belong to. This dataset is expected to be a valuable source for empirical research in the field, such as designcode traceability, software quality assurance, etc. Below, we present two general cases where our dataset could be useful:

Evaluation of scientific approaches and modeling tools. Constructive research on software modeling often has the problem that there are not enough real cases of models to evaluate newly developed approaches and techniques. Currently, this limitation is worked around on the basis of toy examples or artificially generated models. In exceptional cases, researchers are allowed to use obfuscated industrial models or models created with the help of practitioners for the purpose of the evaluation [47]. Our dataset provides real cases of UML models in machine readable form. Professional tool vendors, who provide case tools for modeling, might be able to use the dataset to test new features on real data e.g. layout generation.

UML for education. Software modeling and UML have been taught at Universities for long time. Novice software designers or students struggle with different problems during their training tasks or first software development assignments [48–50]. A recent research by Karasneh et al. shows that using examples could help students who are novices with UML modeling to create better designs [51]. This is aligned with our implication for university teachers in paper C, that teachers should promote students to consume UML models before creating models. Our dataset provides UML models in their real-life settings. Teachers and students can therefore use this dataset as a basis to discuss and explore not only UML syntax but also its use.

Chapter 2

Paper A

Automatic Classification of UML Class Diagrams from Images

T. Ho-Quang, M.R.V. Chaudron, I. Samelsson, J. Hjaltason, B. Karasneh, H. Osman

21st Asia-Pacific Software Engineering Conference (APSEC 2014), Jeju, Korea, December 1 - December 4, 2014.

Abstract

Graphical modelling of various aspects of software and systems is a common part of software development. UML is the de-facto standard for various types of software models. To be able to research UML, academia needs to have a corpus of UML models. For building such a database, an automated system that has the ability to classify UML class diagram images would be very beneficial, since a large portion of UML class diagrams (UML CDs) is available as images on the Internet. In this study, we propose 23 image-features and investigate the use of these features for the purpose of classifying UML CD images. We analyse the performance of the features and assess their contribution based on their Information Gain Attribute Evaluation scores. We study specificity and sensitivity scores of six classification algorithms on a set of 1300 images. We found that 19 out of 23 introduced features can be considered as influential predictors for classifying UML CD images. Through the six algorithms, the prediction rate achieves nearly 96% correctness for UML-CD and 91% of correctness for non-UML CD.

Keywords: Software Engineering; UML; UML class diagram; classification; machine learning; feature extraction.

2.1 Introduction

In software development, UML class diagrams (CD) are used to design and illustrate the structure of software. They are a very important tool when engineers need to understand the basic structure of a system, e.g. when a new engineer, that is unfamiliar with a system, needs to maintain it. UML CDs are becoming ever more prevalent within industry and academia, where model-driven development is becoming a common practice, and it is widely agreed that they have become an integral part [52, 53]. Accordingly, studying UML models and sharing of modeling artefacts [54] is an emerging need in recent years. In order to facilitate this need, a set of UML models should be collected in some forms of repository. Recently, both commercial UML repositories [55, 56] as well as general model repositories [19] have been built. B. Karasneh et.al [20, 57, 58] proposed an automated system (named *Img2UML*), which has the ability to extract UML CDs from images and share these via an online repository.

Among these repositories, enriching the one in [57] is easier, because a large portion of UML CDs is available as images on the Internet. However, the problem is that the automated collection of images needs a classifier to identify which image is related or not. We consider two scenarios where the classifier could be very useful:

- Users want to share their UML diagrams in image formats to the repository; and
- Automatic collection of images from various online sources into the repository. We think about several types of sources: image crawler (e.g. Google Image Search); shared image sets (from academia), etc.

Creation of such a classifier will bring a significant opportunity to automate the repository's collection phase. That is our main motivation to conduct this research.

Research Problem. This paper specifically aims at providing suitable features and classification algorithms to decide which images should be considered as UML CDs and which images should be left out. The classifier operates by extracting relevant information about the image and processing that information with a machine learner. The classifier is expected to have ability of inclusion of UML CD images and exclusion of non-UML CD images. Among these tasks, eliminating non-UML CD images has greater value than including UML CD images.

Since the input is images, information regarding UML CD that helps classifying the images needs to be discovered through image processing. This paper focusses on using basic image processing features as predictors (input variables used by the classification algorithm). The advantage of using the features is that these can be obtained very fast with little effort. This fits our objective of creating a fast method that will be of practical use to automated classification system.

We analyse the predictive power of the features to discover the influence of individual feature on the performance of the classifier. On the other hand,

to find the most suitable set of features, we prepare some sets of features and evaluating their classification performance.

In addition, with the aim at finding the most suitable classification algorithm, we make a comparison between candidate algorithms based on their classification performance. Costa et.al [59] investigated a range of measures that can be used for evaluating classification performance. In this study, the measures are specifically related to algorithms ability to eliminate non-UML CDs.

Contribution. The contributions of this study are as below:

- Proposal of a set of features for UML CD inclusion/exclusion prediction. It consists of 23 features formulated from the image processing properties. The performance of the features is considered as their importance to the classifier. The suitability of four subsets of features is discovered as well.
- Evaluation on classification performance of six algorithms in terms of classifying accuracy and robustness. Candidate set of algorithms includes: *J48 Decision Tree*, *Logistic Regression*, *Decision Tables*, *Random Forests* and *SVM*, and *REP Tree*.
- Our dataset including images together with the list of extracted features are freely provided in order for researchers to test and make comparisons.

The remainder of this paper is structured as follow: Section 2.2 discusses the related research and section 2.3 indicates research questions. Section 2.4 explains the approach while section 2.5 describes the experiment. We present the analysis of results in section 2.6. Section 2.7 discusses our findings and section 2.8 ends with conclusions and future work.

2.2 Related Work

Recognition of special types of graphics is an area of intensive research. A survey of diagrams recognition and treatment can be found in [60]. UML CDs are a type of diagram that graphically represents classes and their relationships to one another. The majority of existing approaches to UML diagram image classification and understanding were developed within the scenario of image feature extraction. This section is aimed at discussing prior research on the topic, with focus on the fields of image classification and UML diagram feature extraction.

2.2.1 Image classification

Image classification refers to the labelling of images into one of a number of predefined categories. D. Lu et.al. [61] introduced major steps for image classification process. The steps may include 1) Selection of training samples; 2) Image pre-processing; 3) feature extraction; 4) Selection of suitable classification approaches and 5) Classification performance assessment. In this study, we follow these steps to build our classifier.

Much research has been done in this field, especially for classifying remote-sensing images [62]. With regards to diagrams, chart image classification seems

to be one of the most concerned topic [63]. However, until now (to the best of our knowledge) there is no study about classifying UML class diagram images.

2.2.2 Diagram feature extraction

Recently, research has been conducted in this field of study, varying in method and approach. B. Messmer et.al. [64] proposed a system for recognizing and automatic learning of sketched graphic symbols in engineering drawings. The objective of this research is to combine pattern recognition techniques with machine learning concepts in order to be able to learn and recognize new symbols in engineering diagrams. In [65–67], a range of methods for online recognition of entirely hand drawn UML diagrams were introduced. However, since the methods were used information regarding the movement of drawing, which are not available in images, sketching tools cannot be carried over to recognizing UML models in final/static images.

L. Fu et.al. [68] presented a method for converting image based engineering diagrams (including UML models) into attributed graphs which can be used for content-based retrieval.

B. Karasneh et.al. [58] proposed a tool to extract class diagrams from computer-generated images. The tool recognizes UML class diagram properties and translates them into XMI format. Geometric-based feature as well as texture features were detected.

2.3 Research Questions

This section describes our main research question and three sub-questions. The main question of this research is as follows: How can classification of UML class diagram images be automated?

In order to answer this question, these sub-questions need to be figured out:

RQ1. What is the performance of image processing features in predicting the presence of UML CD?

RQ2. What is the performance of the classification algorithms in using the features as predictors?

RQ3. Which subset of the proposed features performs the best in classifying UML CDs?

2.4 Approach

In this section, we describe our approach in conducting this experiment.

2.4.1 Overall framework

The overall framework of this experiment is shown in Figure 2.1. To achieve the classifier, we use a machine learning approach.

Input for the process are images (Step 1). The images are then processed by applying a number of sub processes (Step 2) which can be listed as: Recognising contours and shapes; Recognising lines; and joining lines in form of UML connections. Additionally, to avoid prolonged processing time on complex

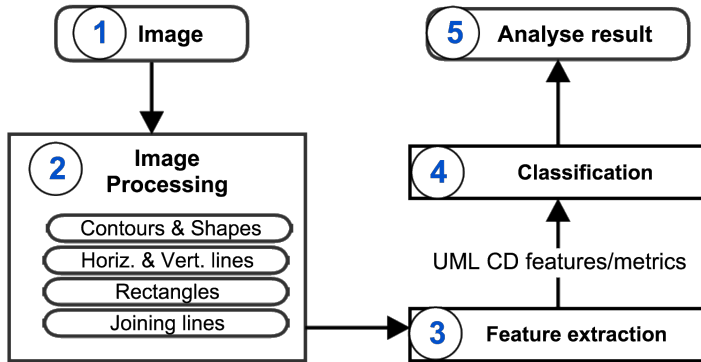


Figure 2.1: Overall framework

photographs, images have to pass a pre-check before being processed. Section 2.4.2 describes the process in details. The outputs of the process are basic characteristics of detected objects (such as size, area, etc.).

In the feature extraction phase (Step 3), information received from the previous step is calculated into 23 invented features/metrics. The output data of the process is represented as float numbers and is much more complicated when metrics comparing with its input data. Detailed information about the features is described in Section 2.4.3.

The features are then used as input data for a UML class diagram classifier (Step 4), which was trained by using our 1300-image-set in conjunction with classification algorithms. The processes of training and predicting is also discussed in the section 2.4.4.

Finally, at the end of the above steps (Step 5), we evaluate performance of the said features and algorithms.

2.4.2 Image processing

This sub-section shows how we process an input image for the purpose of extracting features. The process includes two main phases as follow:

Pre-check. In order to avoid prolonged processing time on complex photographs, images have to pass a pre-check before being processed as follows:

- The most used colour in the image has to cover at least 10
- The images colour-histogram median value must be above 100.

Image processing. Shape and line extraction is carried out using three external algorithms: *Hough transform (HT)* [69]; *Suzuki85 (S85)* [70]; and *RamerDouglasPeucker (RDP)* [71]. The contours that *S85* finds are used to find various shapes and are subsequently broken down into straight lines. Using the algorithm in conjunction with *HT* leads to better detection of lines. The lines are then processed, so that horizontal and vertical lines, that are on the same axes and represent the same line, are joined together into a single line. Rectangles that are not caught by using *S85* are then extracted by finding

horizontal lines that are parallel and in the same position on the x-axis, and have the same two vertical lines intersecting them on each end. *RDP* is used to find different types of polygon: rectangles, rhombuses, triangles and ellipses.

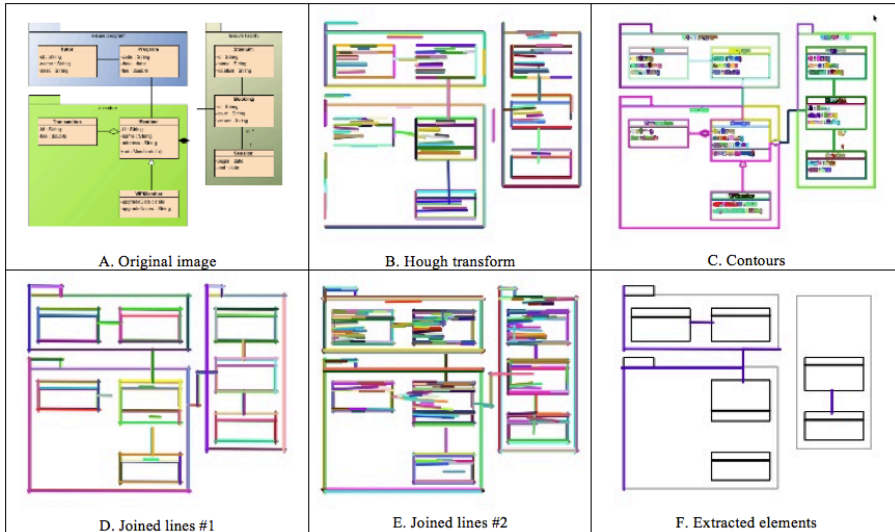


Figure 2.2: Image processing

Figure 2.2 shows the basic steps of the image processing. As can be seen in picture B in Figure 2.2, with *HT*, many of the rectangle lines are not extracted, or the extracted lines are segmented and/or incomplete. Such lines make it very difficult to find the rectangles in the image. *S85* returns an unlimited amount of points in each contour. The extracted contours from *S85* can be seen in picture C. By examining that picture, it is apparent that the algorithm catches more of the lines than *HT*. The lines are joined in three phases:

- (1) The contours that are found are split into lines, and horizontal and vertical lines are extracted;
- (2) Horizontal and vertical lines that *HT* finds are collected and joined with (1);
- (3) Lines, found by *HT*, that are not vertical or horizontal are collected and joined with (1).

After the phases (1) and (2) (picture D), rectangles are collected through the above-mentioned method. After the rectangles have been collected, phase (3) (joining lines; picture E) is conducted, and then all lines within shapes are removed, results in picture F.

2.4.3 Feature extraction

As a diagram, a UML CD image can be distinguished from other images by detecting diagrams main characteristics such as lines, rectangles, number of colours, etc. The task becomes more complicated when recognising UML CD from another type of (engineering) diagram. This section explains the problems and describes the features we extract for solving this classification problem in detail.

2.4.3.1 Which features set UML CD apart from other diagrams?

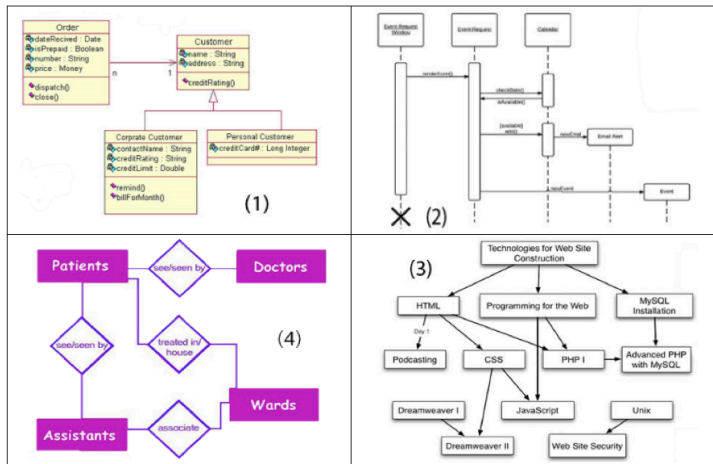


Figure 2.3: Diagram examples.

(1) UML Class diagram; (2) UML Sequence diagram; (3) Flow chart; (4) E/R model

Diagrams come in all shapes and forms (Figure 2.3), and for this reason, it was important to consider not only CDs, but also other different but similar diagrams such as Entityrelationship models (E/R), UML Sequence diagrams and Flowcharts amongst others, when finding the right features.

Three key factors that can be used to describe UML CDs are: (1) Classes, in the form of rectangles; (2) the classes are related to each other in the form of connecting lines; and (3) the classes are divided into sections with the name of the class, attributes and operations. The 3rd describing factor is, though, not universal. It does not apply to all classes within the diagram, but in almost all UML CDs there are classes divided in this manner. As can be seen in Figure 1, the 1st and 2nd of the defined characteristics of UML Class diagrams can apply to many types of diagrams or charts. Because of that it was also important to extract more information from the image, than only information that is descriptive of CD. As a result, other geometrical shapes (i.e. ellipses, rhombuses, and triangles) and statistical metrics (e.g. distribution of shapes) had to be extracted as well.

In order to obtain a general solution, we considered it important that the input images cover wide ranges of sizes, colours and number of objects. In order to make our features comparable, we use normalisation: all extracted features are represented in the form of ratios and percentages.

2.4.3.2 Extraction features in details

There are 23 features that are calculated from image processing images. Table 2.1 describes the features in details.

Table 2.1: Extracted Features

Feat.	Name	Description
F01	Rectangles portion of image, %	Dividing the sum of the area of all the rectangles with the area of the image
F02	Rectangle size variation, ratio	Dividing the rectangle size standard deviation with the rectangle average size
F03-06	Rectangle distribution, %	The image is divided into four equally sized sections and the area of the rects inside the sections is then divided by the total area of the rects. The 4 sections sum up to 100%
F07	Rectangle connections, %	Calculated by counting all rectangles that are connected to at least one rect., and dividing that number by the total amount of rectangles in the image
F08-10	Rectangle dividing lines, %	The rectangles are split into three groups, with rectangles that have: no dividing lines (F08); one or two dividing lines (F09); or three or more dividing lines (F10). This produces 03 numbers that represent the percentage of rects. within each group
F11 F12	Rectangles horizontally/ vertically aligned, ratio	Sides of rectangles, horizontal (F11) and vertical (F12), that are aligned with sides of other rectangles are counted. The numbers are divided with the number of detected rect. in the image, resulting in two ratios on rect. horizontal & vertical alignments
F13 F14	Average horizontal/ vertical line size, ratio	Average size of horizontal (F13) and vertical (F14) lines that are larger than 2/3 of the images width or height, divided by the images width or height, respectively
F15	Parent rectangles in parent rectangles, %	Rectangles that have rectangles within them can possibly be packages. This feature is the percentage of the area of those parent rectangles that is within other parent rects.
F16	Rectangles in rectangles, %	This is calculated in the same manner as F15, but with rects., instead of parent rects.
F17	Rectangles height-width ratio	The average ratio between the height of the rectangles and the width of the rects.
F18	Geometrical shapes portion	The same as F01, but with rhombuses, triangles and ellipses
F19	Lines connecting geometrical shapes, ratio	The number of connecting lines from shapes, other than rectangles, divided by the number of detected shapes in the image
F20	Noise, %	Detected lines that are outside of rectangles, divided by the number of all lines
F21-23	Colour frequency, %	Three most frequent colours in the image are found. Then a percentage out of all appearing colours is found for the three colours

2.4.4 UML CD classification

This subsection explains how we choose a classifier. The process includes two main tasks: 1) Choosing the most suitable classification algorithm; 2) Training for the classifier with the chosen algorithm. The influence of extracted features and correlation-based feature-sets are discovered as well.

2.4.4.1 Choosing the most suitable classification algorithm

We selected the algorithms that represent different approaches in classification. The six classification algorithms are listed as: (1) Decision Table (DT); (2) Random Forest (RF); (3) Support Vector Machine (SVM); (4) Logistic Regression (LR); (5) REP-Tree (RT) and (6) J48 Decision Tree (J48) [40].

At first, we use Information Gain Attribute Evaluator (InfoGain) to find out the influence of extracted features. Secondly, by applying the *Correlation-base feature selection (CFS)* algorithm described in [72] on the extracted features, we prepared several sets of predictors. The set of predictors used for this evaluation are top 3, top 6, top 9 and top-all of most suitable features. Then, we apply these sets of features to all classification algorithms to get their false-positive (FP) and true-positive (TP) rates on our dataset.

2.4.4.2 Training classifier

This sub-section shows the phase of training the UML CD classifier. To that end, we use our 1300-image-set as training data and a supervised learning approach. The collection of the image-set and configurations for training and testing set are described in 2.5.1 and 2.5.3, respectively.

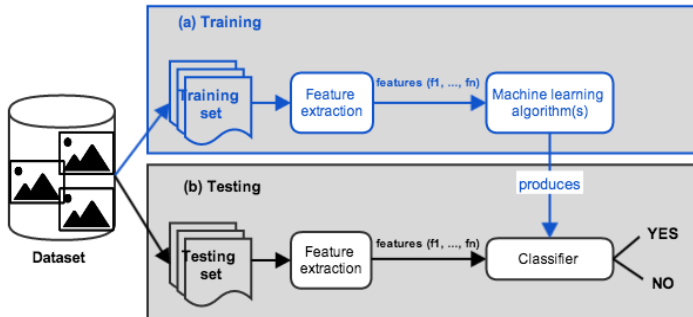


Figure 2.4: Supervised classification - (a) Training phase; (b) Predicting phase

As shown in the Figure 2.4, during the training phase, the feature extraction is to convert each input image to a feature set as mentioned in section 2.4.3. Feature sets are then inputted into the chosen machine learning algorithm to train a model. During the prediction phase, the same feature extraction is applied to the test data, and the extracted feature sets are input into the model to generate the predicted labels.

2.4.5 Analyse Result

The InfoGain measures and the FP and TP rates from the classification process are analysed. We compare the performance of the evaluated classification algorithms across all datasets. The detailed analysis is demonstrated in the section 2.6 of this paper.

2.5 Experiment Description

This section explains the dataset that we used in this study and the evaluation measure for analysing the results.

2.5.1 Dataset

The images that were used for training machine learner collected by using Google Image Search. The image collection consisted of two separate accumulation phases: collecting images that represented CDs; and collecting non-CD images and images that represented similar diagrams.

To search for CDs the phrase UML Class diagram was used. Various types of diagram such as blueprint, sequence diagram, chart, flow chart, E/R model, and architectural diagram were found by their corresponded phrases.

It was verified that no duplicates are in the set. The end-result was a collection of 650 UML CDs and 650 non-UML diagrams (1300 images in total). The non-UML images include 60 sequence diagrams, 34 use-cases, 61 ER diagrams, 80 architectural diagrams and 155 charts. Our dataset together with the results that are presented later in the paper can be found online via: <http://bitly.com/dtsUMLClassifier>.

2.5.2 Evaluation measures

This subsection describes the evaluation measures used in this experiment. The evaluation measures are the following:

2.5.2.1 Features Predictive Performance

In order to measure predictive performance of extracted features we uses the information gain with respect to the class.

The Information Gain Attribute Evaluation (InfoGain Attribute Evaluation) is a method that evaluates the worth of an attribute by measuring the information gain with respect to the class [73]. This method is able to evaluate the predictive power of an attribute (an extracted feature in our case). Accordingly, we use the method to identify the influence of a feature in UML CD prediction. The InfoGain Attribute Evaluation produce a value from 0 to 1 in which a higher value indicates a stronger influence.

2.5.2.2 Classification Algorithm Performance

We use a confusion matrix to evaluate the machine learning classification algorithms. Table 2.2 shows a confusion matrix. In this table, for the case of the actual data is positive (Y), TP represents the number of correct predictions

(true positive) and FN represents the number of incorrect predictions (false negative) by the classification algorithms. In the case of the actual data is negative (N), FP represents the incorrect predictions (false positive) while TN represents correct predictions (true negative).

Table 2.2: Confusion matrix

Actual Result	Prediction Result	
	Y	N
Y	TP	FN
N	FP	TN

We use Sensitivity and Specificity to evaluate the performance of classification algorithms. Sensitivity (or True Positive Rate) measures the proportion of images, which actually are UML CDs, are correctly identified as UML CDs. Specificity (or True Negative Rate) denotes the proportion of actual non-UML CD images that are correctly classified as non-UML CDs. In other words, while specificity represents the ability of excluding non-UML CD images, sensitivity represents the ability of including UML CD images. The two metrics are calculated from the confusion matrix as below:

$$specificity = TNR = \frac{TN}{TN+FP} ; sensitivity = TPR = \frac{TP}{TP+FN}$$

For our purpose, the exclusion of non-UML CDs is more important than the inclusion of UML CDs. As a result, specificity is considered more important than sensitivity. The two measures range from 0% to 100%. The higher the value of the measures, the better classification algorithm.

2.5.3 Experiment settings

This subsection describes the experiment setting in this study. We choose 10-fold cross-validation [74] for performance evaluation where all images are randomly split into ten exclusive folds. For each of the ten experiments, typically a single fold is retained as a validation data, and the remaining nine folds are used as training data. The default settings suggested from WEKA were used for classification algorithms.

2.6 Analysis Of Results

This section describes the analysis of results. Every subsection is presented to answer the questions specified in Section 2.3.

2.6.1 RQ1: Influence of features

The overall results for this evaluation are illustrated in Table 2.3 in which features are sorted by descending order of InfoGain values.

Overall, 19 out of 23 proposed features are considered as influential predictors (InfoGain value > 0) for classifying UML CD images. *F09* and *F08* (ranked first and fifth respectively) are features formulated by calculating splitting lines in the rectangle. Thus, this result shows that splitting lines the rectangle gives a high impact in predicting class diagrams from images.

Table 2.3: Information Gain

No.	Features	Value	No.	Features	Value
1	F09	0.473	13	F18	0.111
2	F20	0.433	14	F14	0.086
3	F01	0.374	15	F10	0.07
4	F13	0.352	16	F21	0.055
5	F08	0.306	17	F19	0.052
6	F02	0.302	18	F22	0.039
7	F07	0.255	19	F15	0.008
8	F04	0.241	20	F23	0
9	F05	0.227	21	F16	0
10	F03	0.208	22	F12	0
11	F06	0.206	23	F11	0
12	F17	0.201			

Note: Features that have InfoGain value greater than 0 are highlighted

Another important feature is *F20*, which is defined to eliminate those images that have too much information outside rectangles. Also, *F01* (ranked third) which denotes rectangle coverage, is one of the most vital features.

F23, *F16*, *F12* and *F11* have a trivial impact on the classification. Thus, these features are then omitted from the feature-set.

2.6.2 RQ2: Classification algorithms performance

The classification algorithms were evaluated by measuring specificity and sensitivity over 10 runs for the feature set.

Table 2.4: Specificity and Sensitivity Scores

	SVM	RF	J48	LR	RT	DT
Spec.	0.89	0.904	0.901	0.914	0.901	0.895
	0.04	0.04	0.04	0.03	0.04	0.04
Sens.	0.924	0.959	0.925	0.902	0.92	0.919
	0.04	0.03	0.03	0.04	0.04	0.04

Table 2.5: Confusion matrix Logistic Regression Classification

Actual Result	Prediction Result	
	Y	N
Y	596	54
N	63	587

Table 2.4 shows the evaluation result. The first row and the second row show specificity score and sensitivity score, respectively. Followers are their *standard deviation*.

As can be seen in the Table IV, in term of sensitivity, *Random Forest* shows an excellent result with almost 96% UML CDs images were correctly classified. This follows with *J48* and *SVM* with 92.5% and 92% respectively.

On the other hand, in term of specificity. *LR* performed the best with 91.4% of correctly classified non-UML CDs images. *SVM* performed the worst specificity with 89%.

The results also show that the standard deviation on the results are relatively small (0.01- 0.05) that indicate the results are considered reliable (small variation). In summary, *LR* performs the best in term of eliminating non-UML CD images. Accordingly, *LR* is considered as the best classification algorithm for our classifier with mentioned-extraction features.

The confusion matrix in Table 2.5 illustrates the classification result generated by applying the *LR* classifier to our dataset. From total of 1300 images, 1183 images were classified correctly. 596 out of 650 UML CD images were correctly predicted as UML CDs. Also 587 out of 650 non-UML CD images were correctly recommended as non-UML CDs. On the other hand, among 117 incorrectly classified images, there was 54 false positives (predicted as UML CDs, but actually non-UML CDs) and 63 false negatives.

2.6.3 RQ3: Set of features Performance

In this subsection, we describe the sets of features that were used as candidate dataset and the comparison between these sets in terms of the performance that classification algorithms can reach on them. Again, specificity and sensitivity are the two measures that are used to evaluate the performance of the feature set.

For this evaluation, we form four feature-sets by grouping the features into 3, 6, 9 and all features. For groups of feature that have 3, 6, and 9 features, we used *Correlation-based Feature Selection (CFS)* Evaluator to select the suitable features. These sets are as follows:

- Feature set 0 (FS0) = All features
- Feature set 1 (FS1) = F01, F09, F13
- Feature set 2 (FS2) = F01, F02, F09, F13, F18, F20
- Feature set 3 (FS3) = F01, F02, F06, F07, F08, F09, F13, F18, F20

As can be seen in Table 2.6, the set of all feature (*FS0*) shows a more positive result compared with other sets in almost of all classification algorithms. Two out of six classification algorithms gained the best result on both specificity and sensitivity scores with the set of all features. With regards to specificity score, *FS0* is the most suitable feature-set for *SVM*, *LR* and *DT*, while *RF*, *J48* and *DT* perform the best on the 6-feature-set (*FS2*). In terms of sensitivity, *FS0* is considerably higher than other sets as it is the best choice for 4 algorithms.

With focus on the best algorithm (*Logistic Regression*) that is analysed above, *FS0* is the best choice in both specificity and sensitivity, at 91.4% and 90.2%, respectively.

Table 2.6: Specificity and Sensitivity Scores Across Datasets

	SVM	RF	J48	LR	RT	DT
FS0	0.890 +	0.904	0.901	0.914 +	0.901	0.895 +
	0.924 *	0.959 *	0.925 *	0.902 *	0.92	0.919
FS1	0.873	0.898	0.908	0.861	0.903	0.895 +
	0.839	0.926	0.92	0.858	0.919	0.921 *
FS2	0.874	0.907 +	0.916 +	0.882	0.905	0.895 +
	0.894	0.947	0.922	0.853	0.924 *	0.919
FS3	0.885	0.906	0.908	0.901	0.907 +	0.895 +
	0.915	0.949	0.925 *	0.892	0.922	0.919

Note: For each feature set: The first row is specificity and the second row is sensitivity cells that have highest value across all algorithms are highlighted as yellow and orange, respectively. For each algorithm: cells that have highest specificity and sensitivity are marked + and *, respectively.

2.7 Discussion

In this section, we summarize and explain the result in the previous section. We also explain the threats to validity of this study.

2.7.1 Image Processing Time

The image processing and input image-set are described in the Section 2.4.2 and Section 2.5.1, respectively. Overall, the average processing time is 5.84 seconds per image. Those images that have big sizes and large amounts of lines need much time to be processed.

In order to discover extraction times dependence on images size and number of lines, we use Pearsons correlation tests. Obtained results show that there is a moderate relationship between execution time and images pixel size (corr. = 0.535). Meanwhile, execution time has a relatively high correlation, at 0.85, with the number of lines. Figure 2.6 indicates the relationship along with its linear model.

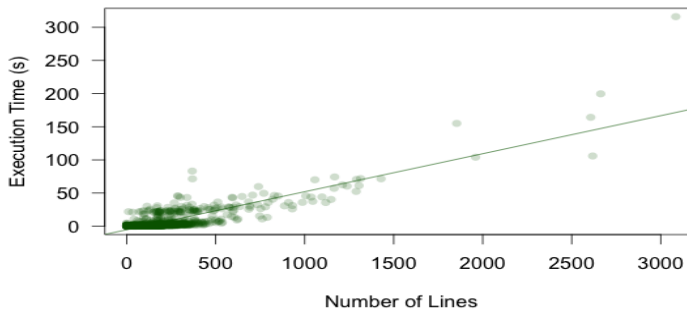


Figure 2.5: Relationship between exec. time and number of extraction lines

As the above discussion attests, processing on architectural diagrams, maps and blueprints might take a lot of time and system resource. Among the 4 most

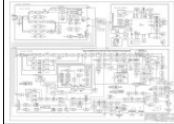


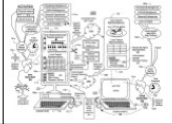
			
t = 315.9 s n = 3085 s = 4758 x 3404	t = 200 s n = 2663 s = 2260 x 1942	t = 164.2 s n = 2607 s = 2106 x 1402	t = 155 s n = 1854 s = 3075 x 2326

Figure 2.6: The most time-consuming cases

(t) extraction time; (n) no. of extraction lines; (s) image size in pixel

time-consuming images showed in the Figure 2.6, only the 2nd one is a UML CD. Therefore, an early recognition of such the images will significantly decrease the execution time. It can be done by applying template-based matching and image pyramids [75].

2.7.2 Image Processing Features Performance

Table 2.7 shows the prediction performance of the six classification algorithms using the features. The classification performance ranges from 89% to 91.4% in terms of specificity and from 90.2% to 95.9% in terms of sensitivity. Therefore, we are certain that the proposed features are suitable for classifying UML diagram based on the input images.

The four features whose InfoGain values equal 0 can be considered of too small of influence and can be excluded from the feature set. We check this exclusion by comparing the performance of the classification algorithms on the two features sets: one is full-feature set (so called *FS23*), and the other is the reduced-feature set (so called **FS19**). The result is shown on Table 2.7 explicitly shows that the exclusion helps the classification algorithms to improve their performance of eliminating non-UML CDs. Comparing with *FS23*, while sensitivity scores recorded on *FS19* slightly decrease with at most 0.2% through all algorithms, specificity values increase from 0.2% to 0.7% on 3 out of 6 algorithms.

Table 2.7: Comparison between FS23 and FS19

	SVM	RF	J48	LR	RT	DT
FS23	0.895	0.902	0.901	0.906	0.899	0.898
	0.929	0.961	0.927	0.903	0.92	0.919
FS19	0.89	0.904	0.901	0.914	0.901	0.895
	0.924	0.959	0.925	0.902	0.92	0.919

On the other hand, results from Table III show that features which relate to class geometric shapes are the most powerful. 8 out of 10 top placed features are about rectangles (distribution, divided lines inside). The two other features are related to connecting lines and information outside rectangles.

2.7.3 Classification Algorithms

The results show that *LR* is a suitable classification algorithm in this study as it produces the best specificity score. However, as can be seen from Table 2.7, *LR* is not the best classification algorithm for all feature sets. There is a remarkable decrease when applying *LR* on *FS1*, as its *sensitivity* is 5.3% less than *FS0*. The most suitable algorithm for *FS1*, *FS2* and *FS3* is *J48* Decision Tree whose *specificity* scores ranges from 90.8% to 91.6%.

In term of *sensitivity*, *RF* maintains its first rank and a reliable performance through all feature sets. Its *sensitivity* scores range from 94.7% to 95.9% with a small standard deviation at 3%. From this analysis, we can conclude that *RF* is the most suitable algorithm for detecting the UML CDs.

2.7.4 Threats to validity

2.7.4.1 Threats to Internal Validity

Image processing phase is done by applying a process mentioned in the section 2.4.2. However, the process itself has some weakness which is formed by *HT*, *S85*, *RDPs* disadvantages [76]. The weakness may causes misdetection of classes and connecting lines. Accordingly, the features that are extracted from the images may not be accurate. Picture F, Figure 2.2 is an example of a misdetection: 2 classes are missing in the final step. Using the algorithms in combination with line-segment merging/grouping algorithms such as [77, 78] should improve the weakness.

2.7.4.2 Threats to External Validity

The class diagrams that we used are collected from the Internet. We believe they are representative for the syntactical representation used in various modeling tools including generic drawing tools. One threat to validity is that we have not included industrial class diagrams. In discussions about this research, people claim these industrial diagrams could be larger in terms of number of classes per diagram. On the other hand, our experience is that large diagrams are decomposed into diagrams that consist of around 10–12 classes per diagram.

2.7.4.3 Threats to Construct Validity

To measure the classification algorithm performance, we use specificity and sensitivity as our evaluation measures. Specificity and sensitivity can be considered as standard measures in data mining [40]. Therefore, we believe there is little threat to construct validity.

2.8 Conclusions and Future Work

This paper presents an automated classification method for images that represent UML Class diagram. To this end, we discuss features extracted from images as input to the classifier for UML class diagrams. In this study, we introduced 23 features that capture statistical and geometric characteristics of diagrams. We find that using these metrics as predictors for the classification

reaches 95.9% and 91.4% (respectively) of correct classification of input images for UML CD and non-UML CD. For this study 1300 different images are collected from the Internet through Google Image Search. We make this dataset available as a benchmark.

We take a step further by examining the classification performance by considering different sets of features. We find out that the full-feature set is the most suitable predictors for most of all classification algorithms. However, we argue that using the full-feature set leads to a time-consuming feature extraction. Therefore, in order to speed up the classification, using other smaller feature-set like *FS2* or *FS3* have only a little lower correct prediction rate, but are faster to compute.

We also study which classification algorithms perform the best on classifying UML CDs. To do that, we calculate and compare their classification performance based on the two measures specificity and sensitivity. Amongst these two measures, specificity is in our case considered more important than sensitivity. Logistic Regression is found to produce the highest correct predication rate, at 91.4%, on identifying non-UML CDs.

Evaluating the performance of classification through the feature sets allows us to identify the most reliable classification algorithms. Random Forest is the most reliable algorithm in term of detecting UML CDs. Meanwhile, J48 Decision Tree obtains top specificity score on 3 subsets of features.

For future work, we will try to improve the performance of the classifier using features based on text-recognition. Another direction would be to try to get a semantic understanding of the diagrams. This could for instance allow the classifier to distinguish organizational charts from class diagrams even if these organizational diagrams cannot formally be discriminated from UML CD syntax.

Also, our classifier allows us to think about a UML CD Crawler which we can use to build a larger collection of UML CDs. Moreover, we consider a classifier for identifying UML sequence diagrams.

Chapter 3

Paper B

The Quest for Open Source Projects that Use UML: Mining GitHub

R. Hebig, T. Ho-Quang, M.R.V. Chaudron, G. Robles, F. Miguel Angel

ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), Saint-Malo, France, October 2 - October 7, 2016.

Abstract

Context: While industrial use of UML was studied, little is known about UML use in Free/Open Source Software (FOSS) projects.

Goal: We aim at systematically mining GitHub projects to answer the question when models, if used, are created and updated throughout the whole project's life-span.

Method: We present a semi-automated approach to collect UML stored in images, .xmi, and .uml files and scanned ten percent of all GitHub projects (1,24 million). Our focus was on number and role of contributors that created/updated models and the time span during which this happened.

Results: We identified and studied 21 316 UML diagrams within 3 295 projects.

Conclusion: Creating/updating of UML happens most often during a very short phase at the project start. 12% of the distinct models occurred several times. Duplicates are in average spread across 1,88 projects. Finally, we contribute a list of GitHub projects that include UML files.

Keywords: UML, open source, free software, GitHub, mining software repositories

3.1 Introduction

The Unified modeling language (UML) provides the facility for software engineers to specify, construct, visualize and document the artifacts of a software-intensive system and to facilitate communication of ideas [79]. For commercial software development, the use of UML has been introduced and commonly accepted to be a prescribed part of a company-wide software development process.

When it comes to Free/Open Source Software (FOSS) development, characterized by dynamism and distributed workplaces, code remains the key development artifact [32]. Little is known about the use of UML in open source. Researchers in the area of modeling in software engineering have performed some efforts to collect examples of models and of projects that use modelling. However the results are often limited [14]. For example, the Repository for Model Driven Development (ReMoDD) [19] is an initiative driven by an international consortium of leading researchers in the field of modeling. Nevertheless its content is growing at a low rate: after 7 years (summer 2014) it contains around 60 models. Industrial projects are very reluctant to share models because they believe these reflect key intellectual property and or insight into their state of IT-affairs.

Due to the so far limited success in identifying open source projects with UML, many researchers (including the authors themselves at the start of this study) are rather pessimistic finding much use of UML in open source projects. Furthermore, since most open source platforms, such as GitHub, do not provide facilities for model versioning, such as tools for model merging, we were even more pessimistic about finding examples of UML models that were updated over time.

The lack of available data is the reason why so far no answers could be given to several basic questions on the amount of UML files in open source projects that are static or updated, the time span during which models are created or updated during the open source project, or the question which of the project's contributors do create models. Thus it seems that UML is not frequently present in FOSS projects. However, there is no exact quantification of its presence.

GitHub hosts around 10 million of non-forked repositories, which makes it a good starting point to obtain an estimation of the use of UML in FOSS projects. GitHub's web search is limited for this type of endeavor as it targets mainly source code searches by developers. However, there are many other ways to access GitHub data (GHTorrent or the GitHub API), but as we will show obtaining data on UML usage is not trivial.

In this paper we present our efforts to mine GitHub in order to gain a list of open source projects that include UML models. Due to the required manual steps, it is not yet feasible to investigate all 12 million GitHub projects. Instead we focus on a random sample of 10% of all GitHub projects (1,24 million of the 12 million repositories). It turned out that for achieving this goal we required to join forces and expertise from different fields. The first challenge is the identification of non-forked repositories in GitHub with the help of the GHTorrent [36] in order to retrieve candidates for files that might include UML diagrams. Since these many of these diagrams are stored in formats that can

also include other information than models, e.g. images or XML based files, it is further necessary to perform an automated recognition of those files that actually are UML. Therefore, it is required to perform two different checks, one for XML based formats and one for images, which is a state of the research technology that just became available in 2014 [80]. Finally, with the retrieved list of UML models, the git repositories of these projects were triggered in order to retrieve information about the repositories and further information about commit and update histories of these models. As a result we gain out of over 1 240 000 repositories a first list of 3 295 projects containing UML models.

The contributions of this paper are: **1.** A first list of 3 295 GitHub repositories including altogether including 21 316 models. This list can be used by other researchers in future to find case studies and experimental data, e.g. for developing model versioning technologies or for studying how design decisions in models transfer to the code. **2.** Based on this data we give for the first time answers descriptive questions about the number of models that are subject to updates, the number of model duplicates that can be found, and the point in a projects life time where models are created and updated. **3.** Furthermore, this research provides the basis to ask when UML models are introduced and updated. Surely the approach has still limitations, for example we will not be able to identify how often the models are read. However, we believe that these first descriptive results are just a starting point. They enable us and other researchers to formulated and address more advanced questions about UML usage and its impacts on a project in future work.

The remainder of our paper is structured as follows. In Section 3.2, we formulate a number of research questions. Section 3.3 shows our review on relevant works. We describe our study approach in detail in Section 3.4. Our findings are presented and discussed in Section 3.5 and Section 3.6, respectively, including the threats to validity. We conclude our paper in Section 3.8.

3.2 Research questions

The data set that we are assessing in this work would allow for a multitude of analysis, e.g. for assessing the distribution of different model types more precisely than it has been done in related work so far. However, answering all questions at once is not possible due to space limitations, but also due to limitation of time. Therefore, we decided to focus in this paper on a set of descriptive questions that had not been addressed in related work so far and that provide a necessary starting point and frame for future analysis:

RQ1: *Are there GitHub projects that use UML? Which are these projects?*

RQ2: *Are there GitHub projects in which the UML models are also updated?*

These first two questions are interesting for two reasons. First, their answer represents a description of the state of practice that was simply not available so far. Second, projects with updates are ideal candidates for future investigations on model usage. For example, they might be used to evaluate facilities for model versioning.

RQ3: *When in the project are new UML models introduced?*

Is it at the beginning of the project or later? What span of the project life time is covered by the phase where UML models are actively created

or modified? Again the descriptive character of this questions is important. Only with the answer, we will be capable to formulate more precise questions on the model usage in future work. For example, whether these figures are homogeneous amongst open source projects or not, will imply directions for future investigations. In long term/ future work this might lead to investigations what form of model usage is most efficient and so on.

RQ4: *What is the time span of “active” UML creation and modification?*

With this question we want to know how long is the time span during which models are in active use during a project? A limitation of our methodology is that we cannot investigate how often and when models are read. However, we can have a look at the time span of active UML creation and modification, i.e., the time between the first introduction of a UML file and the last introduction or update of UML files within a project.

RQ5: *Are UML files originals?* Special model versioning techniques such as model merging are not explicitly supported by GitHub. Therefore, we are interested in the question how many of the found models are duplicates of other models.

Despite the big interest in these questions, it was until now not possible to answer them. The reason is that simply no systematic knowledge exists about UML in open source projects. Furthermore, even if projects are known, it requires advanced mining of the repository in order to get related information about changes and contributors.

3.3 Related research

This paper builds on previous research done in two research communities: the software modelling- and the mining software repositories communities.

3.3.1 Use of UML in FOSS

Studies on the usage of UML are frequently done amongst in industry (mostly through surveys) [26,81]. However, only few studies focus on freely available models, such as can be found in open source projects. Reggio et al. [81] investigated which UML diagrams are used based on diverse available resources, such as online books, university courses, tutorials, or modeling tools. While this work was done mainly manually, Karasneh et al. use a crawling approach to automatically fill an online repository¹ with so far more than 700 model images [20] Both works focus on the models only and do not take their project context into account. Further, they do not distinguish between models that stem from actual software development projects and models that are created for other reasons, e.g. teaching.

An index of existing model repositories can be found online [14]². However, in addition to their small size, these repositories seldom include other artifacts than the models, making it impossible to study the models in the environment of actual projects.

¹<http://models-db.com/>

²Index of model repositories <http://www2.compute.dtu.dk/~hsto/fmi/models.html>

Further, there are some works addressing small numbers of case studies of modeling in open source projects. Yatani et al. studied the models usage in Ubuntu development by interviewing 9 developers. They found that models are forward designs that are rarely updated [21]. Osman et al. investigated 10 case studies of open source projects from Google-code and SourceForge that use UML. They focused on identifying the ratio of classes in the diagrams compared to classes in the code. They find only seldom cases where models are updated [22].

Finally, there are three works that actually approach a quantitative investigation of models in open source projects. Chung et al. questioned 230 contributors from 40 open source projects for their use of sketches [23] and found that participants tend to not update these sketches. A study that focuses on software architecture documentation in open source projects was performed by Ding et al. They manually studied 2 000 projects from SourceForge, Google code, GitHub, and Tigris. Amongst those projects that used such documentations they identified 19 projects that actually use UML [24].

The work that is probably closest to our study is the one of Langer et al. They searched for files conforming to the enterprise architect file format (which is a format that can be used to store UML files) within Google code, assembla, and GitHub. They identified 121 models. They further assessed the model lifespan (between introduction and last update) to be in average 1 247 days [34]. However, studying a single file format is a rather limited view on UML. Furthermore, the project perspective is not considered and they rather put a focus on the used UML concepts.

3.3.2 Mining

Mining software repositories has mainly focused on aspects related directly to (programming) source code. However, projects may include non-source-code sources such as images, translation, documentation or user interface files, that can be usually identified by their extension [82]. By doing so, research has shed some light on the variation and specialization of workload that exist in FOSS communities [83].

The study of specific file formats that are non-source code can be found as well in the research literature: McIntosh et al. have investigated the build system for its evolution [84] and effort [85], or the analysis of infrastructure as code that has become mainstream in the last years [86].

3.4 Methodology

In this section, we describe our study approach. The overall process is shown in Figure 3.1.

First, we obtained a list of 10% of the GitHub repositories from GHTorrent [36] that are not forks. This resulted in a list of files of 1 240 000 repositories, those who had a downloadable branch. From this list, potential UML files were collected using several heuristic filters based on the creation and storage nature of UML files (Step 1). Section 3.4.1 and Section 3.4.2 describe our approach in detail.

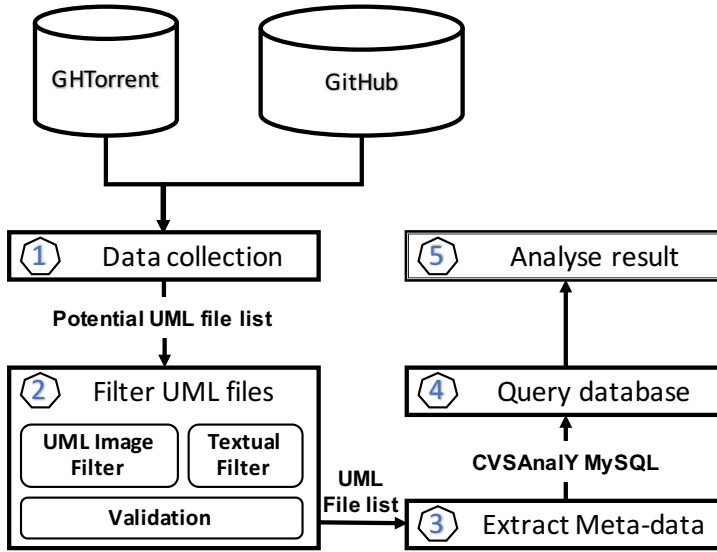


Figure 3.1: Overall processes

An automated process was built to examine the existence of UML notation in the obtained files (Step 2). A manual validation step is taken in order to consolidate the classification result. In the end, we had 21 316 files that contain UML diagrams. We describe the classification method in Section 3.4.3.

We have then obtained the meta-data from those repositories where a UML file has been identified by means of using the CVSAAnaly tool [87] (step 3). Section 3.4.4 discusses tool’s settings and the meta-data structure.

In step 4, we queried the metadata (taken in Step 3) with respect to our research questions. We answer the research questions by analyzing the result (Step 5). Note that during the data analysis further files got lost for diverse reasons (see discussion section 3.6). Thus, we were finally able to analyze a set of 21 316 UML model files.

A replication package of our analysis is available online [88].

3.4.1 Occurrence of UML

To understand how we searched for files containing UML, it is important to understand how these files are created and stored. Figure 3.2 illustrates the different sources of UML files (at the bottom in green). UML models might be created by manual drawing (sketching). Possibilities to create models directly with a computer are the usage of tools that have drawing functionality, such as Inkscape, or dedicated modeling tools, such as Modelio or Argo UML. Some of the modeling tools even provide the possibility to generate UML models, e.g. based on source code. This differences in tool support lead to a wide variety of ways in which UML models are represented by files. The different possibilities are illustrated in blue at the top of Figure 3.2: Firstly, manual sketches are sometimes digitized with the help of scanners or digital cameras and thus lead to image files of diverse formats. Secondly, tools with drawing capabilities can

either store the UML models as images, such as .jpeg and .png or .bmp, or may have file specific formats, e.g. ”pptx”. Thirdly, dedicated modeling tools work with tool specific file formats, e.g. the Enterprise Architect tool stores files with a “.eap” extension. Also some tools work with ‘standard’ formats for storing and exchanging UML: “.uml” and “.xmi”. Yet, modeling tools with specific formats often allow to export and import these standard formats and allow to export the models as images. As a consequence, when searching for UML many different file types need to be considered.

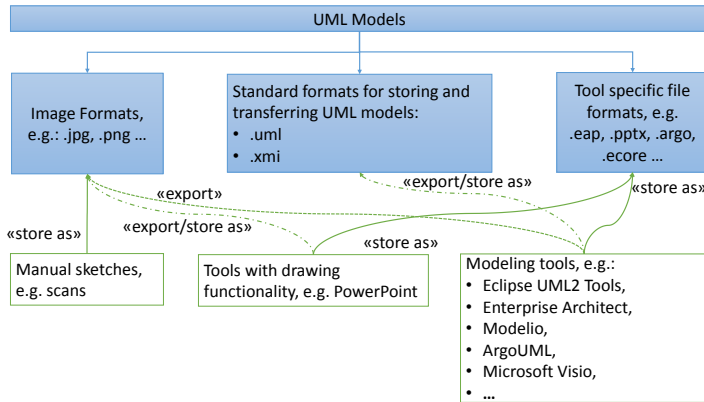


Figure 3.2: There is a large variety of tools for creating and formats for storing UML models

3.4.2 Data Collection

For all repositories from GHTorrent [36] that are not marked as forks, we used the GitHub API: i) To obtain file list for **master** branch; ii) If no master branch found, ask for **default** branch; iii) To obtain the file list from **default** branch. With up to three GitHub calls (i, ii and iii) for each repository, given the GitHub API limitation of 5 000 requests/hour, it took over two weeks to retrieve the complete file list once the machinery was set up.

As explained in section 3.4.1, different file formats need to be taken into account. However, as not every image file is UML, also not every xmi file or files with the endings of tool specific format extensions are UML. Therefore, the filtering process does not only consist of the collection of files with a specific extension, but also of a check whether the collected files are really UML files. It makes no sense to collect files in the first step, for which we have no automated support for the second step.

Since image files as well as standard formats are more common and are created by most modeling tools. For each the development of approaches to identify UML has a good cost-benefit ratio. The applied methods are explained below in section 3.4.3. However, for tool specific formats this ratio can be very low. Therefore, we searched only files of those formats where we could exclude two cases:

- The format is used within the tool exclusively for UML models.

- The file extension of the format is not used by other tools. For example the extension of Enterprise architect files (“.eap”) is also used for Adobe Photoshop exposure files.

To identify these formats we used as a starting point the list of UML modeling tools collected on Wikipedia³, which we as experts consider as one of the most complete lists available. We checked whether the file formats used by these tools do not fulfill the two obstacles mentioned above.

Thus, we search for following file types:

- Images: Common filenames for UML files (such as “xmi”, “uml”, “diagram”, “architecture”, “design”) that have following extensions (“xml”, “bmp”, “jpg”, “jpeg”, “gif”, “png”, “svg”)
- Standard formats: [“uml”, “xmi”]
- UML file extensions that solely relate to specific UML Editor tools: [“aird”, “argo”, “asta”, “dfClass”, “dfUseCase”, “ecore”, “mdj”, “simp”, “txvcls”, “umlx”, “ump”, “uxf”, “zargo”, “zuml”, “zvpl”, “plantuml”]

Hence we do not consider document formats such as word (.doc(x)), .pdf and powerpoint (.ppt(x)). The main reason is that currently technology is not yet capable of extracting UML models out of such general documents.

3.4.3 UML filters

At this stage, the files obtained from Step 1 were checked if they really contain UML notation. More specifically, the files which solely belong to specific UML editor tools were automatically added to the final UML file list.

3.4.3.1 Identify UML images

Firstly, all images were automatically downloaded. Files that could not downloaded or unreadable were eliminated (Result: Successfully downloaded files downloads: 55 747; errors: 1 819). In addition, observations on downloaded images showed a remarkable number of icons and duplicate images. While it’s mostly impossible to find reasonable UML content in icon-size images, including duplicate images in candidate set could definitively cause redundancies to classification phase. Therefore, we eliminated icon-size images. Duplicate images were proceeded as: i) Duplicate images were automatically detected; ii) Representative images were added to classification candidate list; iii) After classification phase, duplicate images of an image will be marked as the same label as the image.

In particular, 15 726 images that have icon-dimension-size no bigger than 128 x 128 were excluded. Subfigures 3.3a, 3.3b and 3.3c show examples of such images.

In order to detect duplicate images, we created a simple detection tool by using an open source .NET library “Similar images finder”⁴. Given two

³List of modeling tools https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools, Last visited 9th December 2015

⁴<https://similarimagesfinder.codeplex.com/>

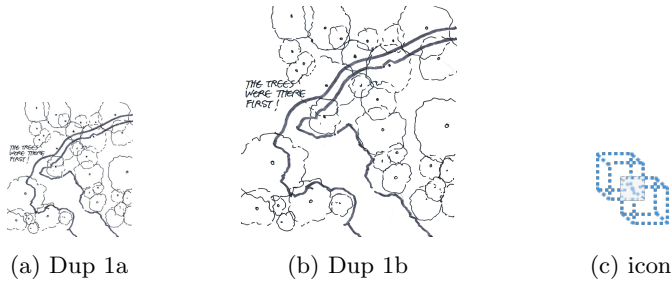


Figure 3.3: Example of duplicates and icon-size images

images, the tool calculates differences between their RGB projections to say how similar they are. In our case, we chose a similarity threshold at 95% since it gave the best detection rate through a number of tests on a subset of our images. Downloading of images took 27 hours.

The final image set of 19 506 images were classified as UML or non-UML images by using a classifier from our prior research [80]. The classifier was trained by a set of 1 300 images (650 UML-CD images and 650 non-UML-CD images). The Random Forest algorithm was chosen since it performed the best in term of minimizing the amount of false-positive rate (expecting below 4%). The automated classification too 26.5 hours. In order to eliminate false-positive and false-negative cases, we manually checked the whole image set. It took 6 working days of effort of an UML expert to complete the checking. This manual check allowed us to prove our classification method and to consolidate classification results. It turned out that the automated analysis had a 98.6% precision and 86.6% recall. The false positives and negatives could be identified due the the manual check.

Gradually, we manually picked up UML in other types (i.e., Sequence Diagram - SD, Component Diagram - CPD, Deployment Diagram - DLD, State Machines - SM and Use-case - UC). UML files that are sketches (SKE) were counted, too. The list of images was marked with a number of labels: "UNREAD", "SVG", "SMALL", "DUP", "CD", "SD", "CPD", "DLD", "SM", "UC" and "SKE".

3.4.3.2 Identify UML files among .xmi and .uml files

Both .xmi and .uml files are specific XML formats. The later ones can include uml models, only and we found 10 171 of them. XMI is a standard format that should enable exchange of models between different tools. In theory it should be simple to identify whether an XML file in general contains a UML model: the schema reference in the XML file defines the content's format.

We performed the analysis in 3 steps:

- [a] In practice the schema reference are often generated in different forms by tools. For example, we found following three schema references to the UML: "org.omg/UML", "omg.org/spec/UML", and "http://schema.omg.org/spec/UML". Therefore we first of all searched with a simple search function for the string "UML" and "MOF" (the meta meta model

of the UML language) in a random subset of the models. This way we could come up with a list of 7 strings representing UML schema references.

- [b] In a second step we automatically downloaded the identified xmi files and parsed them for the schema references. We could identify 876 files with UML schema references. However, 359 files could not be downloaded automatically (for diverse reasons). Here we applied a manual check for the schema references.
- [c] In a last step we wanted to double check that the existence of such a schema reference is sufficient to assume that the file includes UML. Therefore, we took a sample of four open source projects containing together 53 (between 1 and 33 respectively) links to xmi files. In addition to the check for schema references, we went manual through the content of the 53 files to assess whether and what kind of models they include. A comparison of the results with the data from the step above confirmed that the existence of an UML/MOF schema is a reliable indicator for rating a file as UML: of the 53 xmi files, 30 had been rated by both approaches as UML, while the other 23 were rated as non-UML.

Finally we run a duplicate detection on .xmi and .uml files by calculating and comparing hash values of the file contents.

3.4.4 Metadata Extraction and Querying

We downloaded all repositories where at least one (real) UML file was identified and extracted its metadata with the help of CVSanaly [87]. 100 repositories from the initial list could not be retrieved, due to various reasons, from some giving errors to others having changed to private repositories.

In average, around 30.000 projects per day were downloaded for each Github account. Taking these results a time span of 14 months ((12.847.555 projects / 30000) / 30) would be required for the analysis, when using one single Github account. As this would have made this study infeasible, we parallelized the retrieval of the JSON files through many Github accounts, which were donated during this process. This reduced the time span to approximately one month. While the download is an automated process, but the parallelization is not. It took around 1 h 30' each day to run and check each set of repositories, using up to 21 Github accounts. Altogether this process took 6 weeks.

After this process, we had 21 316 of the identified UML files from 3 295 repositories and the corresponding meta-data in a SQL database. A new SQL table was added to the ones provided by CVSanaly with just the UML files for easy and efficient querying. A set of Python scripts were then used to answer the RQs stated in this paper by querying the database and aggregating the data accordingly. This final step took 14 days.

3.5 Results

This section presents the results of our investigation. Together with this research an ample amount of data have been used, usually handled by scripts

developed by the authors. Detailed information of the former and the code of the latter can be obtained in the replication package⁵.

3.5.1 RQ1: UML in GitHub projects

We downloaded 1 240 000 non-forked GitHub repositories obtained from GHTorrent. After filtering the data for potential UML files based on type, we retrieved a list of 100 702 links. Of those, 21 316 were classified as UML.

The further extraction of model related data, turned out to be an additional filter, since details could not be extracted for all files. The reason for this is due to the fact that our retrieval procedure takes so much time that context changes. So, for instance, in the time that goes from the retrieval of information of the files they are included in a project (July/August 2015) to the time where the git repositories were downloaded (November/December 2015), some of them were renamed, deleted or made private.

In consequence, 21 316 files could be retrieved for the following analysis (as summarized in Table 3.1). These files belong to 3 295 GitHub projects. Of these 1 947 include a single UML file, only and 1 169 projects include between 2 and 9 UML files. Furthermore, we identified 158 projects with 10 to 99 UML files and 4 projects with more than 100 UML files. In the following analysis, the later 21 projects are taken separately, when statistics per model are shown. The reason is that they show very different characteristics and would, with their large number of models⁶, strongly bias and hide trends that occur within the other projects. This first list of identified GitHub projects that include UML can be found online [88].

Table 3.1: Found distribution of model files by formats

	xmi	uml	jpeg	png	gif	svg	bmp
Share	3.4%	44.9%	4.7%	29.6%	16.6%	0.6%	0.2%

Results for RQ1: The here identified repositories with UML files represent already 0.28% of the GitHub repositories. Of these, two thirds of the projects contain a single UML file.

3.5.2 RQ2: Versions of UML models

The next important question was whether models are 'read-only' or also sometimes updated.

Table 3.2 summarizes the distribution of model files by number of updates per model. Our results show that the vast majority of the UML files (18 867) are never updated. Nonetheless, we found that more than 11% of the UML files in our sample (2 449 models) were updated one or more times. Further, the number of updates of models that are updated is on average 3,0 times (although the median, which is more significant given the skewed distribution,

⁵Replication package <http://oss.models-db.com>

⁶One of the projects is "eclipse/emf.compare/", which includes more than 6 000 models. We strongly assume that many of these models are generated, e.g. for tests.

is 1 time). Furthermore, Table 3.2 summarize the distribution of projects by sum of model updates or all models of a project.

26.67% of the projects in our sample include at least one model update. Models are less often updated in projects that have more than 100 models (38.09% in our sample), in contrast to 26.60% of the models in projects with less than 100 models are updated. There are only 11 projects that include more than 100 model updates.

Table 3.2: Distribution of files / projects by number of updates

number of up- dates	models in projects with 1 to 99 models	models in projects with ≥ 100 models	projects
0	7 947	10 921	2416
1	946	466	332
2	336	42	157
3	151	19	78
4	107	7	64
5	82	2	51
6	67	4	34
7	38	1	18
8	24	3	17
9	24	1	12
10	11	2	8
<20	70	3	50
<30	24	0	25
<40	14	0	8
<50	1	0	6
<60	2	0	2
<70	0	0	0
<80	0	0	2
<90	1	0	3
<100	1	0	1
>100	0	0	11

Results for Q2: Only 26% of the investigated projects updated their UML files at least once.

3.5.3 RQ3: Time of UML model introduction

Figure 3.4 shows the dates of the introduction models considering the amount of days since the start of the project, while Figure 3.5 displays the same information by dividing the duration of the project from the start to nowadays in a normalized way (so, the 50% mark would be half of the project duration since its start until today).

Projects with less than 100 UML models seem to have a tendency to introduce models at the project start. In contrast, the 21 projects with 100 or more models show a different graph. We decided to show the numbers

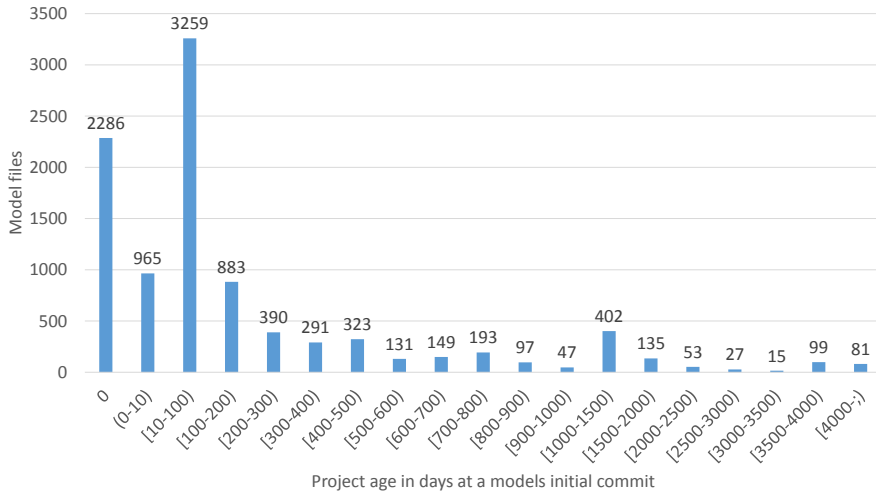


Figure 3.4: Distribution of model files sorted by project's age in days when the diagram was introduced (models within projects that have less than 100 models)

separately, since these projects with partially more than 1 000 models would easily bias the presented view.

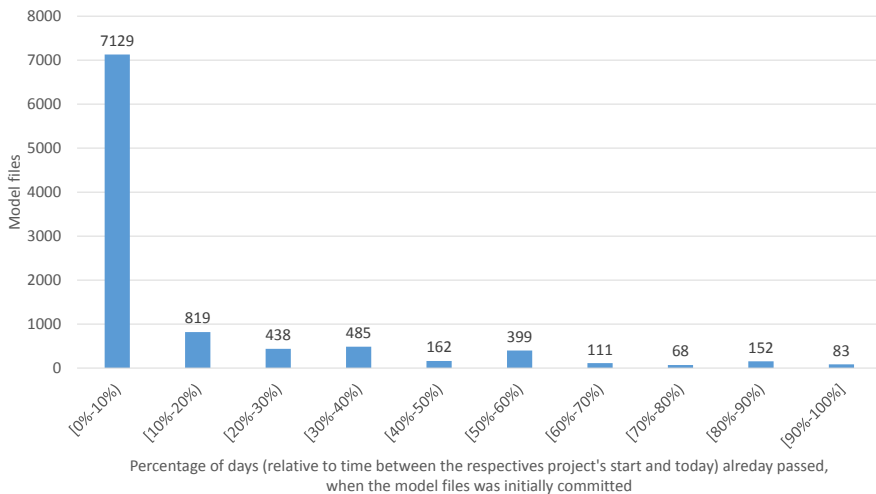


Figure 3.5: Distribution of model files sorted by percentage of project time that passed when the UML file was introduced (for projects that have less than 100 models)

However, we found that calendar time (days) may not be the best way to consider a project's progress, since the amount of activities can highly vary during the lifetime of open source projects. Figure 3.6 shows the distribution of the models based on the time of their introduction when measured by the

percentage of the project's commits.

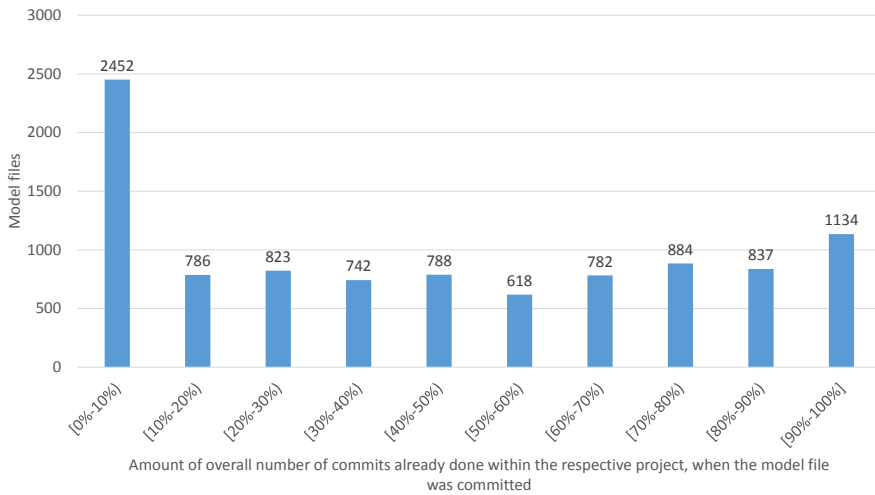


Figure 3.6: Distribution of files sorted by number of overall commits done when the diagram was introduced (For models in projects that have less than 100 models)

An interesting difference between the two views is that the consideration of time in terms of amount of commits shows a much more balanced view. While this may not be the most intuitive notion, it helps to place the modeling activities relative to the active phases of the project. Thus we can see whether model introduction happened before or after a majority of other development activities (such as coding or documenting). In addition, it helps to better represent projects that had their main activity in the past and/or have become inactive. From our results, it can be seen that new models are introduced predominantly in the early phases (above 25% of them in the first 10% of the commits), but that new UML models are introduced in later phases too.

Finally, as mentioned above the results look very different for the 21 projects that have 100 or more models. As Figure 3.7 illustrates there most models are introduced during the last third of the project activities.

Results for Q3: UML models are introduced in all active phases of a project with a tendency towards the early phases.

3.5.4 RQ4: Time span of active UML

In this RQ, we have looked at the time span of active UML creation and modification, i.e., the time between the first introduction of a UML file and the last introduction or update of a UML file within a project.

Figure 3.8 summarizes these time spans. The maximum time span found is thereby 100% of the projects commits, while the median of the time spans is 5.8%. We found that by far most projects seem to introduce (and update) all models within a single day. Model creation and updating plays only in a minority of the projects a role during more than 10% of the project's commits.

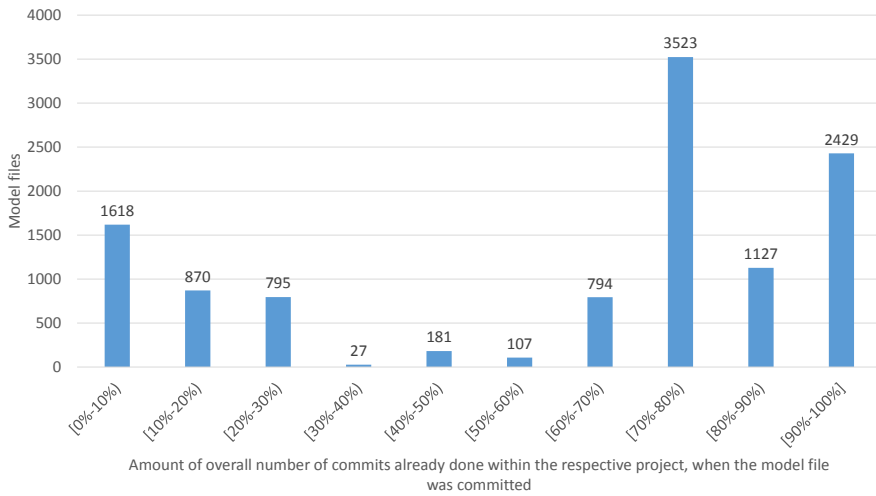


Figure 3.7: Distribution of files sorted by number of overall commits done when the diagram was introduced (For models of the 21 projects that have 100 or more models)

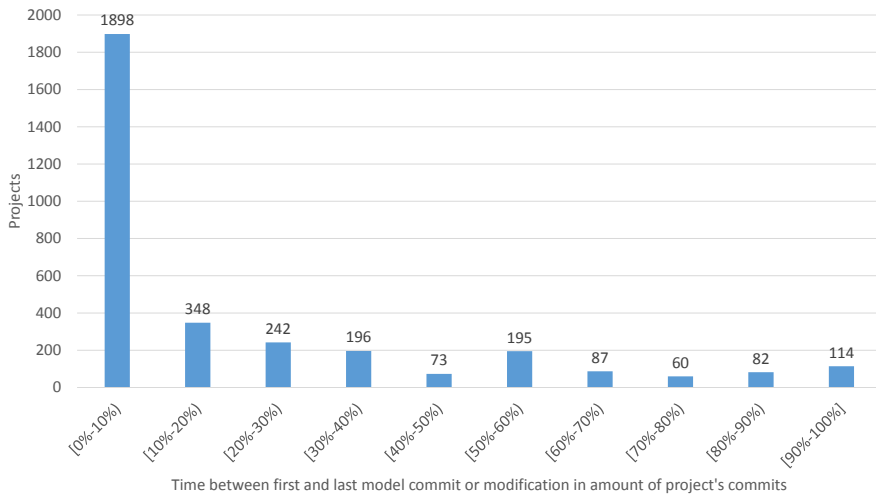


Figure 3.8: Projects by time between first model commit and last model update-or-commit as percentage of project's commits

As with RQ3, we use commits as an alternative measure of the time where UML introductions/updates occur. Figure 3.9 presents the active UML phase for all 3 295 projects from this perspective. The active UML phase of a project is given horizontally in percentages of commits done, starting when the first model is introduced and ending when the last model is introduced or updated. The diagram illustrates nicely the findings from above that a minority of projects (less than 10%) have UML active phases that cover nearly the whole project life time. For a majority of projects the active UML phase is very short, many of them concentrating this activity in the first commits.

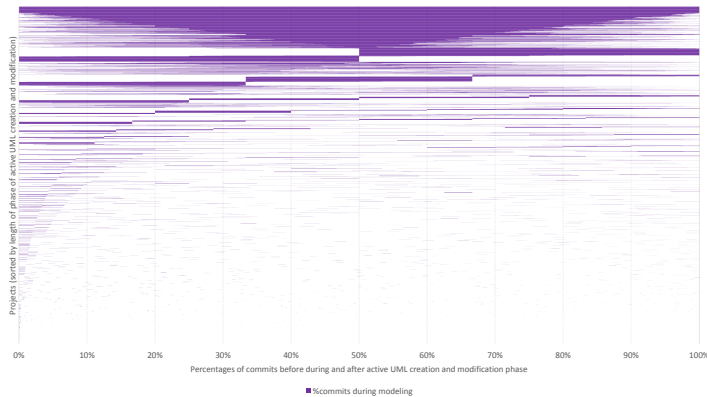


Figure 3.9: Plot of all 3 295 projects illustrating the placement of active UML creation and manipulation phase within the overall project life span. Time is measured in percentages of commits done, when the first model is introduced and the last model is introduced or updated. The projects are sorted by the relative amount of the active modeling phase (projects with a relatively long active modeling phase are at the top, projects with a shorter phase are at the bottom).

Results for Q4: Few of the studied projects are active with UML during their whole lifetime. In general, the projects work very shortly on UML, usually at the beginning.

3.5.5 RQ5: Duplicates

Our final question was whether the 21 316 found model files are all distinct originals. To answer the question we used automated duplicate detection, as indicated in the method section.

As a result we identified that 16 576 of the 21 316 found models were unique in our sample. The remaining 4 741 model files represent 2300 models of which each occurs at least twice. Thus, 21 316 found model files include together 18 876 distinct models. In Figure 3.10 we summarize how often models with duplicates occurred in our sample. Interestingly, one of the models was found 79 times. In average, models that are duplicated are duplicated 3,63 times.

Furthermore, we investigated, whether duplicates of a model belong to the same project. To our surprise this is only for the half of the models

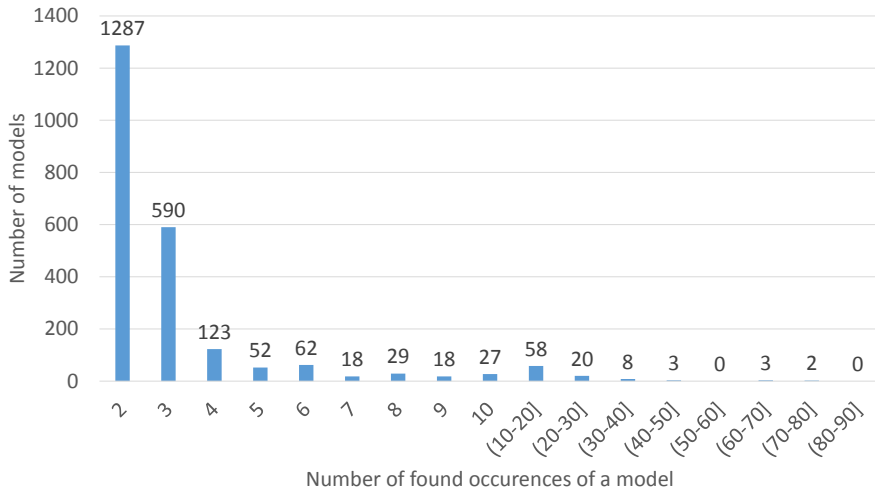


Figure 3.10: Histogram of models that were found at least twice indicating how often models occur.

with duplicates the case. However, the roughly the half of these models have occurrences in multiple repositories (up to 43). In average the number of projects over which duplicates of a model are spread is 1,88. Figure 3.11 summarizes the results in form of a histogram.

While duplicates that occur in the same repository might be result of attempts to model versions, we cannot explain the high number of cases were models occur in multiple projects. A possible explanation might be that models might be stored as part of platforms or plug-ins that are reused in multiple projects. Another explanation could be project forks that are done manually by cloning repositories instead of using GitHubs fork mechanism.

Results for Q5: While most models seem to be unique, a large number of identified distinct models (12%) occur several times. In average duplicates are spread over 1,88 projects.

3.6 Discussion

Considering our initial expectations we were surprised to find such a big number of projects with UML. Surely, 3 295 projects are still a small number compared to the overall number of GitHub projects. Nonetheless, the identification of 21 316 UML models exceeds by far the expectations that we had based on the numbers of models found so far in open source projects in related work, e.g. 121 models by Langer et al. [34] or 19 projects with UML by Ding et al. [24].

Data consistency We want to shortly discuss the type of data that we can get with the presented mining method. The method we applied is not trivial and consist of several steps of data collection. For example, we search for UML candidates using a GHTorrent dump, but accessed the GitHub API to retrieve

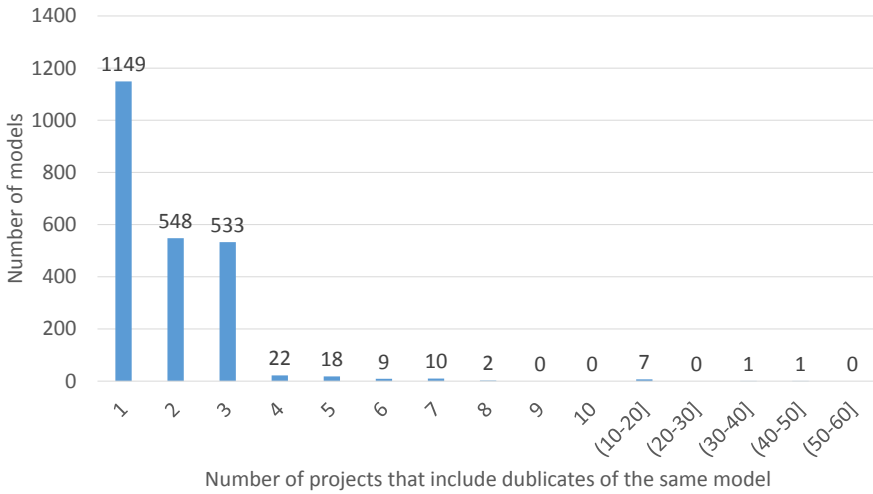


Figure 3.11: Histogram of models that have duplicates in one or more projects. The histogram shows the number of models by number of projects within which occurrences of a model were identified.

further information about model contributors. Due to the difference in time between the creation of the GHTorrent dump and the request to the GitHub API, we had drop outs of identified models/projects during the second step.

In addition, we performed this method for the first time, which had an exploratory component in trying out what kind of data we can (and need to) retrieve. This led to the situation that we accessed the GitHub API several times, leading to different drop-outs in models and projects for the different types of information collected.

A *lessons learned* is that, for the next analysis, we have to make a clear planning of all required data in advance, to ensure that at least the second threat to data consistency can be reduced. For this paper we addressed the problem with a reduction of the finally analyzed data set to models and projects for which we had the data points that are necessary to answer the different research questions.

Static models A finding is that many projects use UML only in a very static way. In such projects models are never updated and often all models are introduced at the same point in time. These results confirm findings from smaller studies such as Yatani et al.’s [21] or our own (Osman et al.’s [22]), who both found that updates of models are rare. This can have different reasons. One optimistic interpretation would be that models are just introduced as first architectural plans that are followed and used as documentations, but never changed. Another rather pessimistic interpretation would be that modeling is just “tried-out” at some point in time and then dropped. An observation that at least supports the idea that the optimistic interpretation plays a role is that in most projects the main activities of introducing models happen during the first half of all commit activities.

Projects with regular model usage Another number that we consider surprisingly high is the number of projects (or models) with more than 20 updates as well as projects with more than 1 year of active UML creation and modeling. Again, compared to the number of overall GitHub projects the here found number seem small. Nonetheless, it was unexpected to find several projects that seem to use modeling on a regular basis.

It has to be noted that the results we found are in contrast to the study of Langer et al. [34] who found an average model lifespan of 1 247 days, while studying 121 enterprise architecture models in open source. We found much lower lifespans. The difference in the findings might be caused by the fact that enterprise architect is a modeling tool that is rather used in an industrial context. Thus, the probability that the projects studied by Langer et al. [34] have industrial support is very high.

Model genesis An aspect that we could not address in this study the source of the models or the reason for model usage. Accordingly, the data set was not filtered to exclude for example student projects. We expect this to influence the the findings in this paper, since student projects might show different patterns of model updates, model introduction time, and life span than non-student projects. Addressing this threat will be subject to future work.

Different populations A finding that is supported by multiple of the figures shown above is that there seem to be different populations of model usage. A first hint that the data set covers different populations can be seen in Table 3.2. There is a difference in the number of model updates between projects with more than 100 model files and projects with less than 100 model files. One reason for different populations could be the actual form of model usage and creation. Models might be created manually or automatically (e.g. through reverse engineering). They might solve as plans for system design or as description for an already existing system. Model updates might be performed in order to make small corrections after an initial creation (leading to updates within in short span of time) or in order to make a documentation up to date after a longer phase of system change. At the current state we do not know whether these populations can actually be distinguished on their characteristic commit and update pattern. However, a further hint that they might play a role can be seen in the relatively constant distribution models by the amount of commits that were already done within a project (see Figure 3.6). We can see model introductions at all project ages. The in average short time of active UML creation and modification speaks against the idea that these introductions at different points in time happen within the same projects. Thus, it seems that we have to deal with different groups of projects introducing their models at different points in time. In future work we plan to have a closer look at the model usage in order to study whether we can associate pattern to different populations of model use.

Duplicates The large number of identified duplicates leads to questions. What are the reasons for duplicates? Missing model versioning techniques alone cannot explain the found results. Furthermore, it is not clear yet whether these duplicates represent a form of model use. E.g. if models are adopted

together with code from other projects, they might be used to understand the alien code that is embedded in a new project.

Paving the way for future research Finally, one of our main contributions is that we presented a method to systematically mine for UML models in GitHub and that this leads to an enormously promising set (much larger than any existing set of projects) for future analysis. On the one hand this will help us to address in future question that arise from the findings of this paper. For example, concerning the model updates, it would be interesting to consider following questions:

- Are models updated by their original authors or by other people?
- In how many projects are UML files obsolete?

Further considering the time of model introduction, we would like to address the following question further: Has the time of introduction an influence on the "success" of an open source project, i.e. the question how many developers join a project? And of course we would like to address the question whether different populations of model-usages can be statistically distinguished.

Even more important, the hereby published list of open source projects using UML can help other researchers to progress in their studies. For example:

- What kind of UML diagrams are used most often?
- What coding languages are used most often in combination with UML?
- What files are changed together with changes in architectural models?
- Can UML help to attract and integrate inexperienced developers?

Furthermore, the data can be used to find case studies for other model or architecture related research, such as:

- Does a good architectural design in models help to create a good architecture in the code?
- Tools for traceability management and model merging can benefit from the real case studies.
- Research that integrates models into fault prediction can be evaluated with the help of that data.

Thus, we believe that the identified initial list of open source projects with UML will be of great help for other researchers, too.

3.7 Threats to validity

We defined a number of threats to our research's validity. We categorized them by using the validity terminology introduced by Wohlin et.al [89]. We identified three types of threats to validity, they are: Construction Validity, External validity and Conclusion Validity.

3.7.1 Threats to construct validity

There were a number of threats that might cause the loss of UML files during data collection phase:

- With regards to the materials that were used to collect data, we used a subset of GHTorrent SQL dump from 2015-06-18 which is out-dated at the current time. Accordingly, newer projects have a higher probability to be dropped out. In addition, the limitation of 5 000 hits per hour of GitHub API made data collection last long. Requests that were done at different points of time during the period could give different outcomes, and probably the loss of potential UML files.
- Our collection method, which made use of a number of heuristic filters, might overlook potential UML files which are not complying with searching terms and file-type list. We noticed some cases where UML files had been named differently such as *act-cartesortir.jpg* and *FrameworkInterface.png*. Further, we restricted the search to file formats for which we had techniques to decided, whether the file includes UML. This excludes a couple of other formats which might include models, such as some formats from modeling or graphic tools (e.g. visio files or enterprise architect files), but also documents that might include models as part of documentations, e.g. pdf and word (docx) files or powerpoint.

The loss of UML files might affect to our analysis in the sense that it could make us underestimate the number of projects with UML models and the number of UML models. Being aware of the above consequences, in this research, we don't use our data to analyze the frequencies of model usage as well as the evolution of model usage in general over *the years*. We were focused on getting an overview of various aspects of the use of UML in GitHub projects. We expect no systematic bias concerning the aspects that we investigated!

The applied mechanisms for duplicate detection allow us to identify duplicates within the same file type. However, we cannot identify whether an image and an .xmi file are duplicates. This might lead to an underestimation of the amount of models in this paper. Despite this limitation, our results are already interesting and we consider them a valuable starting point, towards a better understanding of model usage in FOSS.

Kalliamvakou et.al discuss a number of promises and potential risks that researcher might be faced when mining GitHub repository [90]. We found that the threat that many active projects might not conduct all their software development in GitHub could somehow mitigate our analysis.

3.7.2 Threats to external validity

During data collection phase, in order to minimize the possibility of incorrectly collect non-UML files, we excluded some tool-specific file types form the search for UML models. This might reduce the generalization of our results with respect to these UML tools. However, most of these tools, e.g. Enterprise Architect, are commercial. It is to be investigated in future work whether they are used in open source projects to a similar degree as non-commercial formats.

Data in this research was only taken from GitHub, but not other OSS hosts/platforms such as SourceForge, Google Code, etc. As they differ to each other in terms of size, functionality, users and user's behaviors, the results of this paper can hardly be generalized to the other platforms. It is possible that UML is used in a different ration within projects at other platforms. However, as GitHub is one of the biggest player in the field, we strongly believe that our investigation gives valuable insights to a majority of the OSS community.

A manual glance at the retrieved list of UML models shows that several project paths include names such as "Assignment" or "master's thesis". While this is no direct threat to our results, it limits the generalizability. For example, it is possible that many of the projects that include single UML files only, actually are result of university teaching.

Last but not least, outcomes of this research can not be generalized to closed source community.

3.7.3 Threats to conclusion validity

As described above, the data has some limitations which permit to do analysis of frequencies, since we expect to have only discovered a part of the overall set of UML models and respective projects. In particular we have not considered powerpoint, pdf, and word-formats of documentation in which UML models may be embedded. For that reason we do not do statistical analysis or even predictions, but stay on a descriptive level in this paper. Nonetheless, we are convinced that this descriptive analysis already represents a valuable contribution to the research community.

3.8 Conclusions

In this paper we joined forces in repository mining and model identification in order to identify open source projects on GitHub that contain UML models.

As a result we can present a list of 3 295 open source projects which include together 21 316 UML models. This is the first time the modeling community can establish a corpus comparable to collections already exist for source code only, such as *QualitasCorpus*⁷. Furthermore, the relatively low amount of UML projects amongst the investigated GitHub projects (0.28%) reconfirmed that our systematic mining approach was required in order to establish the corpus.

We analyzed the data to gain first descriptive results on UML model usage in open source. One finding is that the majority of models is never updates, but that projects exist that do update their models regularly. Furthermore, we learned that models can be introduced during all possible phases in the lifespan of an open source project. Nonetheless a peak of model introduction is during the first 10% of the duration of projects.

A few projects are active with UML during their whole lifetime. However, most projects work very shortly actively on UML, usually at the beginning. We found that 12% of the distinct models occurred several times. Duplicates are in average spread across 1.88 projects.

⁷QualitasCorpus <http://qualitascorpus.com/>

In the future we plan to further explore the possibilities that arise with the here presented new method to collect data about UML usage in open source projects. For example we plan to analyze the impact of model usage on project dynamics, such as the number of people joining projects. We are planning to proceed with mining GitHub in future work. Based on the now investigated 10% of GitHub we expect that GitHub includes around 34 000 projects with UML and together around 200 000 UML models. Furthermore, we will investigate possibilities to identify UML models that are embedded in other files such as manuals stored in pdf.

Chapter 4

Paper C

Practices and Perceptions of UML Use in Open Source Projects

T. Ho-Quang, R. Hebig, G. Robles, M.R.V. Chaudron, F. Miguel Angel

Accepted at 39th International Conference on Software Engineering - Software Engineering in Practice Track (ICSE SEIP 2017), Buenos Aires, Argentina, May 20 - May 28, 2017.

Abstract

Context: Open Source is getting more and more collaborative with industry. At the same time, modeling is today playing a crucial role in development of, e.g., safety critical software.

Goal: However, there is a lack of research about the use of modeling in Open Source. Our goal is to shed some light into the motivation and benefits of the use of modeling and its use within project teams.

Method: In this study, we perform a survey among Open Source developers. We focus on projects that use the Unified Modeling Language (UML) as a representative for software modeling.

Results: We received 485 answers of contributors of 458 different Open Source projects.

Conclusion: Collaboration seems to be the most important motivation for using UML. It benefits new contributors and contributors who do not create models. Teams use UML during communication and planning of joint implementation efforts.

Keywords: UML; architecture documentation; OSS projects; GitHub; motivation; communication; effectiveness of UML

4.1 Introduction

Open Source Software (OSS), which has its roots in the free software movement, started partially as a counter-movement to the software industry in the 80s and 90s [91]. Even though, there was a clear border between OSS and industry, the situation started to change in the late 90s and early 2000s. In those years, some industry started to early adopt the OSS movement practices, collaborating with communities [92], or some companies were created around some communities [93]. Many projects created foundations to serve as an umbrella to collaborate and integrate software industry partners [94].

Thus, we have witnessed a process and technology transfer between OSS and industry that has made the line between both be vague nowadays. Notable contributions from OSS to industry have been technologies, such as git and GitHub, and community-managing practices, although the list of adoptions is much larger [95]. On the other hand, OSS has embraced practices from industry, such as (modern) code review practices and planning and requirements analysis mechanisms [96]. Companies with a large pool of developers try to have an “internal” OSS-like ecosystem, a concept coined as inner source [97]. Many OSS practices are commonly taught at universities, and young graduates start their professional careers with experience in OSS, whether in languages (Python, Perl, Ruby...), products (jQuery, Hadoop...) and tools (GCC compiler tool chain, git and GitHub...) [98]. And the software industry is looking into popular OSS repositories, such as GitHub, to find suitable candidates to fill open development positions [99].

In this regard, we have seen a clash of two worlds, resulting in new practices where industry sometimes has adopted elements from OSS and vice versa. As the trend seems to go on, we would like to draw attention on modeling, specifically on the use of Unified Modeling Language (UML) in OSS. UML has been around as a graphical language for modeling software systems for about 25 years. As far as it is known, UML is not yet frequently used in OSS projects, with a rather marginal use [100]. OSS is known to be programming-driven, with other tasks having room for further improvement [82]. However, modeling is used in major companies [26]. Modeling is, thus, an area where we can find a gap between OSS and industry. Given that the use of UML in OSS is not very well-known, we would like to shed some light into this issue with the aim of discovering how UML is used and whether it is considered useful. We hope that the results will help to understand whether the use of UML in OSS helps these projects and whether industry working with OSS projects should promote its use.

To this end, we used a technique that we developed to find UML use in GitHub projects [100]. This effort showed the feasibility of our approach and triggered us to come up with various research questions addressed in this paper, where we scanned through the majority of non-forked GitHub projects (over 12 million of projects) and identified which of them use UML.

We performed a large scale survey directed at those projects that use UML, with focus on how it is used and impacts development activities. The contributions of this research are: i) the identification of a large set of OSS projects that use UML, and ii) insights from a large scale survey of OSS developers that use UML. Amongst other insights, we have found that UML is

used to coordinate the development. Furthermore the use of UML seems to help new contributors to get started, although it does not seem to attract new contributors. The set of projects we identify are a valuable resource for future empirical studies regarding UML.

The rest of this paper is constructed as follow: We formulate a number of research questions in Section 4.2, then introduce related work in Section 4.3 and describe our research method in Section 4.4. Section 4.5 presents our findings. Our findings, possible threats to validity and implications of our research are discussed in Section 4.6. Conclusions can be found in Section 4.7.

4.2 Research Question

To better understand the use of UML in OSS, we formulate the following three main research questions:

RQ₁: *Why is UML used in OSS projects?*

To get an impression of the role of UML models in OSS projects, we formulate this first question as the following.

- *SQ_{1.1}:* What are the motivations to use UML modeling?
- *SQ_{1.2}:* What are the reasons not to use UML in projects?

RQ₂: *Is UML part of the interaction of (a team of) contributors?*

Teams and interaction between developers play an important role within today's software intensive industry [31]. Models are used as basis for planning and work coordination. However, it is an open question, whether UML models fulfill a similar role in OSS projects. We approach this question from three aspects: 1) awareness of developers about the existence of UML models within the project, 2) the use of UML during project planning and communication, and 3) the role of UML during joined implementation efforts. These three sub-research questions are structured as follows:

- *SQ_{2.1}:* Are developers aware of the existence of UML in their projects?
- *SQ_{2.2}:* Are UML models used during communication and team decision making?
- *SQ_{2.3}:* Are modeled designs adopted afterward during the implementation phase by teams of OSS contributors?

RQ₃: *What is impact/benefit of UML?* Much research has been performed to identify benefits of UML usage in industry. However, it is not yet clear whether UML usage impacts or even benefits development in OSS. Again, we consider three different perspectives: 1) the role of UML for novice contributors, 2) the impact of UML on the working routine, and 3) the impact of UML on the attractiveness of a project for potential contributors. The following sub-research questions are structured:

- *SQ_{3.1}:* Can UML models support new contributors?
- *SQ_{3.2}:* What are the impacts of using UML in OSS projects?
- *SQ_{3.3}:* Can UML models help to attract new contributors?

4.3 Related work

In the following we discuss related studies about UML or modeling in industry and OSS.

4.3.1 Modeling in Industry

Modeling has been widely studied in industry, in particular in several surveys. Torchiano et al. found that models help to improve design and documentation [26]. However, they also found that model usage is connected to extra effort, especially due to a lack of supporting tooling. Forward et al. find that models are primarily used for design and documentation, while code generation is rather seldom [27]. Gorschek et al. focused on a different population, which are programmers, partially working in industry and OSS [28]. Within their sample design models are not use very extensively. However, models and UML are found to be used mainly for communication purposes. Further, they report on a higher use of models for less experienced programmers.

Case studies have also been performed in order to investigate the impact of modeling/UML usage. For example, Baker et al. found an increase of productivity when using UML in Motorola [4]. Nugroho et al. investigated an industrial case study and found that UML usage has the potential to reduce the defect density and, thus, increase the quality of software [30]. Just as in the case described by Kuhn et al., most of the case studies draw a picture of model use, where models are actually artifacts that are produced and consumed by different people [31].

4.3.2 Modeling in Open Source Software

Much less work has been done on UML use in OSS. One reason for this is the challenge to actually find cases that can be studied. For example, Badreddin et al. studied 20 projects without finding UML, and concluded that it is barely used in OSS [32]. Similarly, Ding et al. found only 19 projects with UML when manually studying 2,000 OSS projects [24]. However, in our previous work, we presented an approach that allows to find thousands of projects with UML by mining GitHub [100]. There are several investigations of single or very small numbers of cases of OSS projects that use UML, e.g. by Yatani et al., who found that models are used to describe system designs, but are rarely updated [21]. Osman et al. studied to what extent classes in the diagrams are implemented in the code [22]. Finally, Kazman et al. investigate the Hadoop Distributed File System to learn how documentation impacts communication and commit behavior in the open source system [33]. There are some studies that approach the use of models in OSS with a quantitative perspective, studying a large number of projects. For example, to study the use of sketches, Chung et al. collected insights from 230 persons contributing to 40 OSS projects [23]. Finally, Langer et al. studied the lifespan of 121 enterprise architect models in OSS projects [34].

However, to the best of our knowledge there is so far no quantitative study targeting the use of UML within the team communication and its effects.

4.4 Research Methodology

In this section, we describe our study method in detail. The overall process is shown in Fig. 4.1.

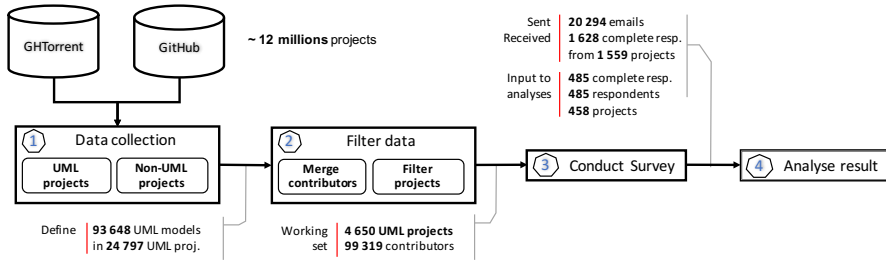


Figure 4.1: Overall process

4.4.1 Data Collection

The first step is to identify UML files in GitHub repositories. In our previous work, we analyzed 1.2 million GitHub repositories to identify UML files in them [100]. In this study, we have extended the data collection to the whole GitHub database. A number of changes have been made in order to adapt our method to the retrieval and analysis of such a big dataset. In this section, we briefly summarize the data collection steps and the changes that were made.

4.4.1.1 Obtaining the full list of GitHub projects

To obtain the list of projects, we used the data from the February 1st 2015 dump of GHTorrent [36]. From this dataset we identified a list of projects that were not deleted and non-forks. As GHTorrent does not contain information on the files in the repositories, we made use of the GitHub API to retrieve the list of files, for a total number of 12,847,555 repositories. The result is a JSON file per repository with information on the files hosted in the *master* (or *default*) branch of the repository.

4.4.1.2 Identifying UML files

The next step was to identify UML files from the file list. First, potential UML files were collected using several heuristic filters based on the creation and storage nature of UML files. After that, an automated process was applied to examine the existence of UML notation in the obtained files. A manual validation was made to consolidate the results. Details about the identification procedure are described in Section 4 in [100]. At the end of this step, we had 93,648 UML files from 24,797 repositories.

4.4.1.3 Extracting meta-data

For all projects that contain a UML file, development meta-data from the repositories has been retrieved. Therefore, we use `perceval`, an evolution of the well-known `CVSanaly` software [87], that allows to obtain these data in JSON files, allowing to perform the data extraction process in parallel. It took

the five instances of the tool over 4 weeks to complete this task. At the end, and after removing 240 JSON files that contained 404 Not Found responses, we had 24,125 JSON files that were parsed and normalized, and finally converted into SQL.

4.4.2 Filtering the obtained projects and contributors

In this phase, we aimed at mitigating a number of known threats to validity when mining GitHub, i.e., sample/short-time projects [90] or identification of contributors [101].

4.4.2.1 Filtering short-time projects

For this paper we aim at projects that are interesting from an industry perspective. Thus, we focus on projects that are not short-term and that do not consist of a single contributor. We define short-time projects as those projects that have: i) active time (time between the first and the latest commits) less than 6 months, OR ii) less than 2 contributors, OR iii) less than 10 commits. After classifying and filtering short-time projects, 4,650 UML-projects (out of 24,125, we use the term *UML project* to refer to GitHub projects that contain UML file(s)) and 2,701 (out of 17,101) non-UML projects met our requirements. The final list of the projects is shared in our replication package¹.

4.4.2.2 Merging duplicate contributors

A contributor can use different emails or usernames during the project time, and thus a procedure has to be applied to merge all the identities into a unique one. In our work, developers who have different identities are merged when they have the same e-mail address or the same full name. In the case of the full name, we consider them to be the same if at the full name is composed of at least two words or of only a word and numbers, e.g., "arg123". This is a rather conservative approach, but it minimizes the number of false positives [101]. After running the script, the original 129,276 contributors result in 99,319 distinct ones.

4.4.3 Conducting the survey

In the following, we give a short overview about how we conducted the survey.

4.4.3.1 Participant

To ensure that we obtain a balanced picture, we had to consider the role that contributors play within the OSS projects with UML. Two dimensions of roles are important (each questioned person would fulfill a combination of roles in these two dimensions):

- Founder (F) vs. non-founder (NF)
- Non-UML Contributor (NUC) vs. first UML Contributor (1UC) vs. UML updater (non-1st contributor) (UC)

¹The replication package for this paper can be found at <http://oss.models-db.com/2017-icse-seip-uml/>

Consequently, each interviewed participant fulfills one of the following six roles: F-1UC, F-NUC, F-UC, NF-1UC, NF-NUC, NF-UC. For each project, we randomly selected three contributors, to whom we sent the questionnaire. The selected three contributors had to fulfill one of the following three constellations of roles.

- F-NUC, NF-1UC, NF-UC
- F-1UC, NF-UC, NF-NUC
- F-UC, NF-1UC, NF-NUC

For those projects where we could not identify any NUC or UC (e.g., projects that have only one UML contributor), we contacted less contributors.

4.4.3.2 Questionnaire

The questionnaire has been designed to meet the following requirements:

Multiple roles We send different question sets depending on the role of the contributor. For example, NUCs are asked whether they are aware that UML models exist in the project, while UCs are asked if they think that NUCs are aware of them. Thus, depending on the role, participants received between 5 (NF-NUC) and 19 (F-1UC) questions.

Exploration We use a funneling approach (from broad to narrow) when designing the survey. For example, if a UC uses a UML model for architecture/design purpose, we would ask if the model is adopted, and eventually, who implemented the model. Accordingly, the number of questions will not only differ among different roles, but also among respondents who have the same role. In addition, to gain more insights, we use a mix of close-ended and open-ended questions in the survey.

Personalized Contact To ensure that participants know what projects and UML models we are referring to, we personalized the email with which we contacted potential survey participants by concretely referring to his/her GitHub identification, the name of project of interest, and (if applicable) an URL to his (first) UML commit or to a UML file committed by someone else. By following the URL (e.g., <https://github.com/rvs-fluid-it/wizard-in-a-box/blob/master/src/doc/wizard-in-a-box-design.png>), participants could get further contextual information about the UML models, for example commit messages, commit date, etc.

We used the Lime Survey tool² as it offers the possibility to perform on-line surveys. Our Lime Survey server is hosted at <http://survey.models-db.com/>. Details about survey settings (questionnaire and its data-flow diagram) and email templates can be found in the replication package.

²LimeSurvey homepage: <https://www.limesurvey.org/>

4.4.3.3 Sending out the survey

We sent 20,294 survey emails to OSS contributors in 6 days, from July 21 to July 26, 2016. More than 1,000 emails were not sent because of various problems, including out-dated email addresses, etc. We sent reminder emails after one week, and finally closed the survey in August 4, 2016. Altogether, we received 2,230 responses, being 1,628 completed. After filtering responses that belonged to short-term projects, we had 485 survey responses of respondents from 458 projects.

Table 4.1: Number of emails sent, number of responses and number of responses after filtering by participant categories

	Founder			Non-Founder			SUM
	<i>1UC</i>	<i>NUC</i>	<i>UC</i>	<i>1UC</i>	<i>NUC</i>	<i>UC</i>	
Sent emails	4509	3891	713	6737	3221	1223	20294
#full resp.	373	293	68	564	210	120	1628
#inc. resp.	167	105	24	214	56	36	602
#fil. resp.	<i>84</i>	<i>79</i>	<i>27</i>	<i>176</i>	<i>80</i>	<i>39</i>	<i>485</i>
Percent(%)	17.3	16.3	5.6	36.3	16.5	8.0	100

4.4.4 Data Analysis

First, we take into account completed responses only. Second, we do not consider short-time projects.

Part of the questionnaire are free-text questions. We use these questions to learn about phenomena for which we do not know a fixed set of answers yet. The goal of analyzing the data is to identify re-occurring themes. Therefore, we used a coding technique, following the constant comparison method as described by Seaman [102]. We decided to use an empty starting set of codes and develop them during the coding. For each of the question two of the authors coded the answers independently. In a second step we inspected the codes together to identify and if necessary resolve differences in the selected codes and application of the coding. Afterward, we went a second time through the data in order to ensure that the now fixed set of codes was assigned consistently. We did this i) to increase the quality of the coding and ii) to decrease the probability that we miss interesting aspects. As a final step we checked whether codes occurred for more than one project, in order to prioritize those themes that are of greater relevance.

Furthermore, we took those cases where we got multiple responses for the same project and aggregated them. This aggregation was done as follows: we interpret observation based questions (i.e., whether UML is used for communication) as reports about a project. Thus, aggregating a “yes” and a “no” answer for the same project to a “yes” to indicate that there is a report about a phenomenon for that project. Similarly, we prioritized “no” over “I have no opinion”. “I do” and “I have seen other people doing” are merged to “I do”.

4.5 Results/Findings

4.5.1 Respondent Demographics

A total of 2,230 respondents from 91 countries began the survey, with 1,628 completed compulsory questions of the survey. After filtering out survey responses from short-time project participants, we ended up with 485 survey responses of respondents from 458 projects.

Table 4.2 (Appendix 1) and Figure 4.16 (Appendix 2) show the distribution of the respondents by country and continent, indicating the majority of the responses originating from Europe with 57.52%, followed by North America and South America.

Among the 485 respondents, 190 (about 40%) are founders of an OSS project and 159 (32.8%) are non-UML contributors (Table 4.1). Regarding the educational background (as shown in Fig. 4.2), 37.73% of respondents had a Master's degree, 30.31% a Bachelors, 16.29% a Ph.D., and 11.75% were still in education. About 4% of the respondents identified themselves as autodidacts. A vast majority of the respondents reported to be familiar with architecture documentation in different formats, mostly UML (90.31%), then auto-generated code documentation and software models in generic formats (78%) (Fig. 4.3). Only a half of them (45%) were familiar with architectural notations on white papers. There are programming languages where UML is more frequently found (Smalltalk, Java, C# and C++). On the other side, UML has not that much impact in the Objective-C and the Ruby community.

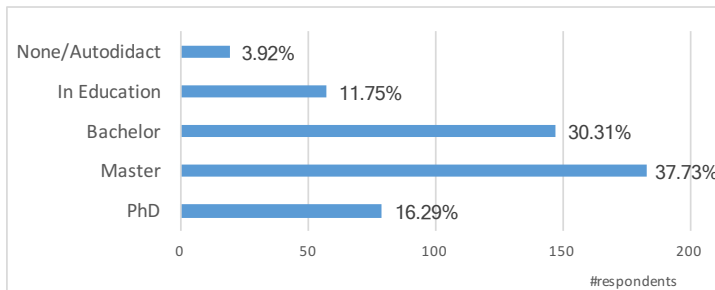


Figure 4.2: Distribution of respondents based on their highest educational background

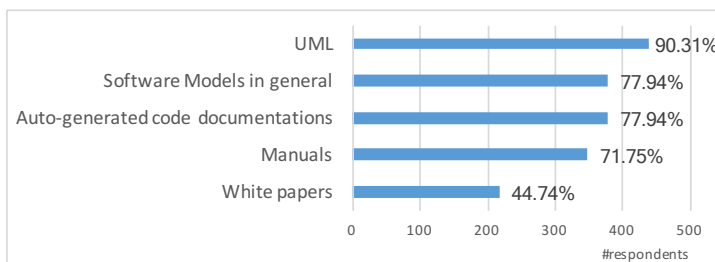


Figure 4.3: Familiar architecture document formats (multiple choices were allowed)

4.5.2 Why is UML used?

4.5.2.1 What are the motivations to use UML modeling?

Fig. 4.4 shows the answers from 326 UCs (from 319 projects) about the *intent* of UML files they added/updated. Most of UML files served for design/architecture and documentation purposes, with 70% and 71% of votes, respectively. For about 18% of the projects, software verification was mentioned as one of the main purposes. Refactoring and code generation was less usual (14.11% and 12.85% of the projects).

Among 125 NUCs that claimed to be aware of the existence of UML models, 109 people (from 109 projects) reported to find UML helpful (Fig. 4.5). 79% of the respondents found UML useful for understanding the OSS systems. They also found UML models helpful as the models assisted in improving communication within their project, guiding implementation and managing quality of the project.

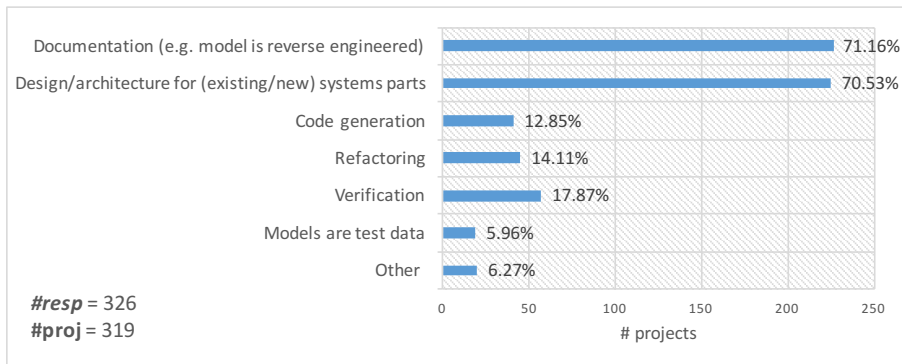


Figure 4.4: Intent of UML models that were added/updated

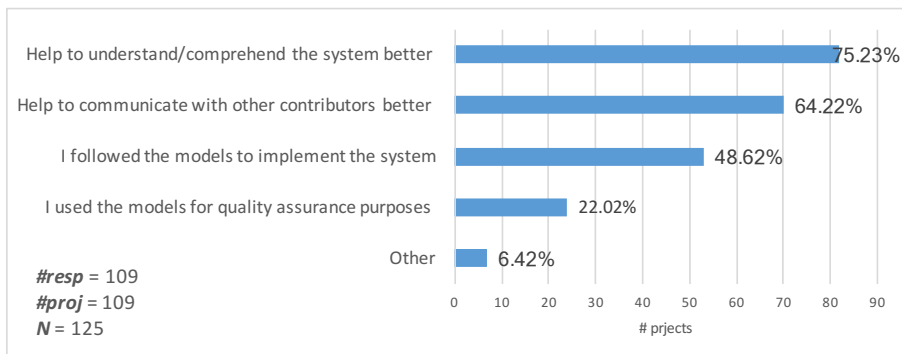
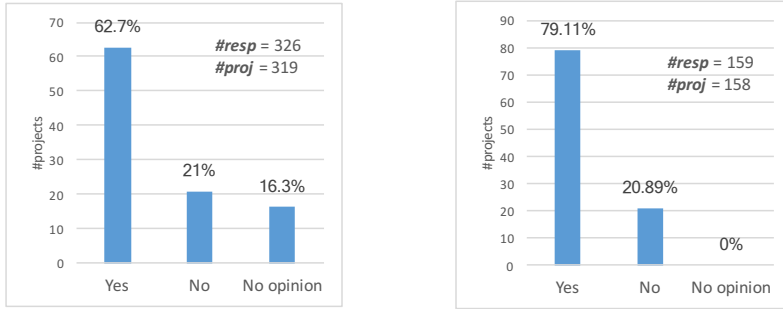


Figure 4.5: How did UML help non-UML contributors?

Results for SQ_{1.1}: The majority of models are intended for creating software designs and documenting software systems. Non-UML Contributors (NUCs) benefit from UML models when it comes to understand a system and to communication.



(a) Do UCs think that other contributors are aware of UML?

(b) Are NUCs aware of the existence of the UML models?

Figure 4.6: Awareness of developers about the existence of UML in their projects (by project)

4.5.2.2 What are the reasons not to use UML in projects?

To complement our finding on the motivations to introduce/use UML, we asked the 16 NUCs who did not find UML models useful the reasons for this. Respondents from 6 projects actually had not used models, finding themselves not required to learn/use UML (e.g., “*there was no demand to do so*”). Interestingly, in no case license problems for modeling tools were a problem.

In 4 cases, the UML files were outdated. Other reasons that were brought up in free-texts are: missing support for versioning models, a failed attempt to understand the models, a preference for other means of communication (face to face), a preference for other forms of modeling/sketching, a preference for reading code rather than spending time for UML models, and the dislike of UML (anti-UML attitude).

Results for SQ_{1,2}: Only a small number of respondents found UML not useful.

4.5.3 Is UML part of the interaction of contributors?

4.5.3.1 Developer’s awareness about the existence of UML in their projects

To answer this question, we first asked creators/maintainers of UML models whether they think that the models are known by developers of the projects (summarized in Fig. 4.6a). In 62.7% of the 319 projects with responses, the UCs/1UCs believed that UML models are known by the developers of the projects. Second, we asked NUCs of projects that use UML if they are aware of the existence of UML models in their projects (Fig. 4.6b). Surprisingly, for the vast majority of projects (80%) NUCs stated that they are aware of UML models.

To better understand the difference between the answers of UCs and NUCs,

we looked in detail into the 24 projects for which we received responses from NUCs and UCs. In 10 out of 24 projects, NUCs and UCs differed. Interestingly, UC(s) did not expect their UML to be known by other developers although NUCs were aware of it in 8 of them. It seems that model creators tend to underestimate the spread of their models.

Results for SQ_{2.1}: A majority of non-UML contributors are aware of the UML models in their projects. Awareness is higher than the one expected by the authors of the models.

4.5.3.2 Are UML models used during communication and team decision making?

In a first step we asked founders and UCs whether UML models are considered in the communication between contributors. Fig. 4.7 summarizes the 405 individual responses from 388 projects. According to the responses, UML models were considered in communications in a large majority of the participated projects (60%).

As a step further, we asked whether UML models were used as a basis for architectural decision making or mentoring activities. Respondents from a majority of the projects recalled that they had used the UML models for making architectural decisions (58.7%) and to explain each other different aspects of the system (58.25%) (Fig. 4.8).

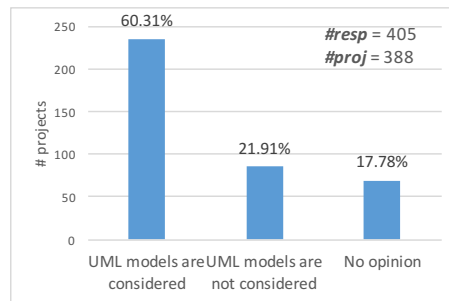


Figure 4.7: Are the UML model(s) considered in the communication between contributors? (per project)

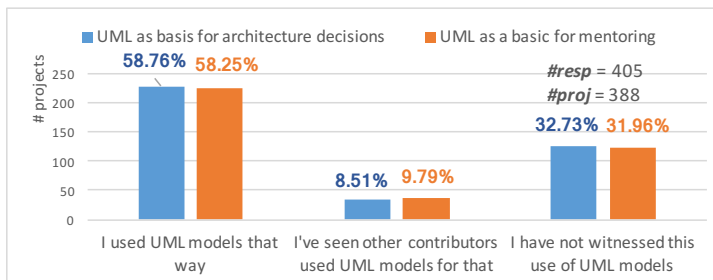


Figure 4.8: Is UML a basis for architectural decisions or mentoring activities? (per project)

Results for SQ_{2.2}: UML models were considered as a mean of communication, as a basis for architectural decisions, and for mentoring in a majority of the projects.

4.5.3.3 Are modeled designs adopted afterwards, during the implementation phase by teams of OSS contributors?

For those projects that claimed to have design models, we asked the question “Was the UML model adopted during the implementation phase?”. Fig. 4.9 summarizes the answers of the 231 respondents from 225 projects. In most cases UML models were adopted partly or completely during the implementation phase (about 92%).

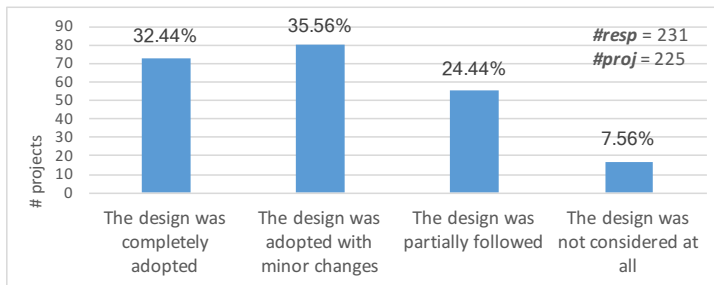


Figure 4.9: Was the UML model adopted during the implementation phase? (by project)

If the answers were that UML models were at least partially adopted, we asked further questions to find out who and how many contributors implemented the modeled designs. Fig. 4.10 and Fig. 4.11 summarise the responses per project (based on 214 individual responses for 208 projects).

Creators of UML models are greatly involved in implementing the modeled designs (in 88.5% of the projects). Experienced contributors helped in 35.5% of the cases and novice contributors helped in around 13% of the cases.

In the majority of the projects (around 66%) more than one person participated in the implementation of previously modeled designs. However, only 7% of the projects reported to have more than 5 contributors involved in such joint implementation efforts.

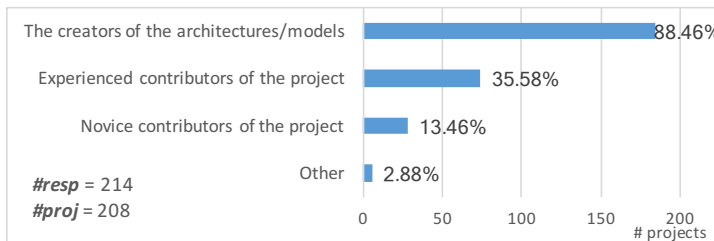


Figure 4.10: Who implemented the UML models? (by project)

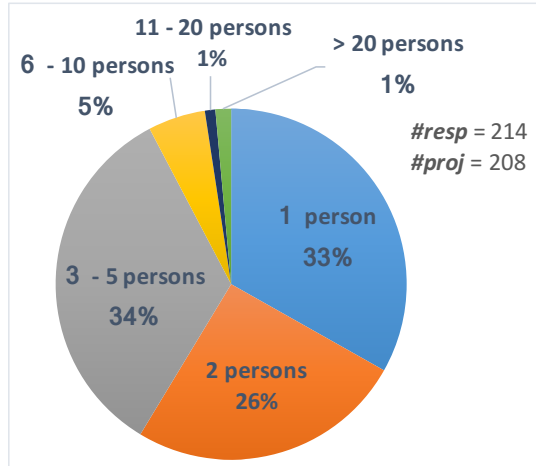
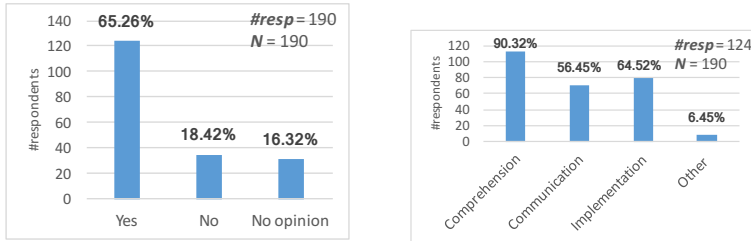


Figure 4.11: Number of contributors who implemented UML models in a project



(a) Do UML models help new contributors? (b) For what tasks do models help?

Figure 4.12: Responses for the questions whether UML models help new contributors to join a project.

Results for SQ_{2.3}: Designs introduced with UML are in most cases adopted during the implementation phase (fully or with slight changes). Most often these designs are implemented by groups of 2-5 developers.

4.5.4 What is the impact/benefit of UML?

4.5.4.1 Can UML models support new contributors?

We used two perspectives to approach the question whether UML models support new contributors.

First, we ask founders if they think that UML models help new contributors to join their projects. We received 190 responses from 84 F-1UCs, 79 F-NUCs and 27 F-UCs. For those who agreed, we further asked with what tasks models help. Fig. 4.12 shows the responses in detail. 124 out of 190 respondents (65.26%) agreed that UML models can help new contributors when joining projects. They expected models to assist new contributors in comprehending the system (90%), during implementation phases (65%), and when communicating with other contributors (56.5%).

Second, we asked each contributor what software artifacts he/she used when

they got started with the project. 485 contributors answered this question. Despite the fact that most of respondents were familiar with architectural documents (as shown in Section 4.5.1), source code still remains their first choice to start working with an OSS project (81%) - see Fig. 4.13. Remarkably, UML and software models in general were reported to be starting points for 55% and 43.5% of the respondents, respectively. This is more than the proportion of contributors who started using wikis, issues, manuals, and auto-generated code documentation. This conforms with the answers given by the founders about new contributors.

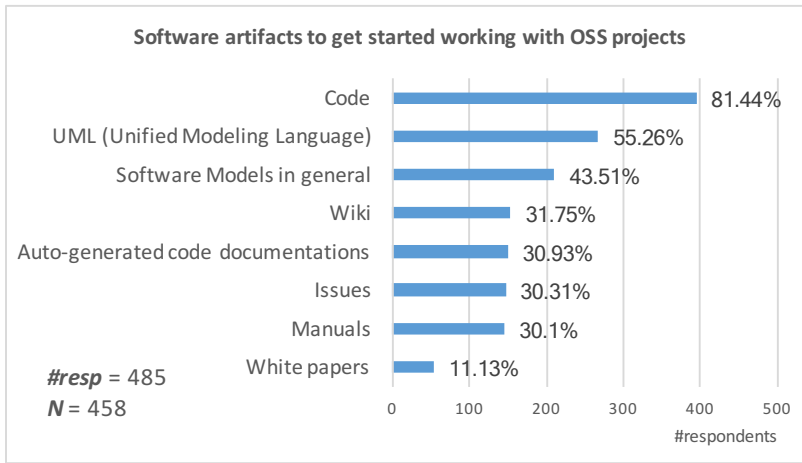


Figure 4.13: Software artifacts used by respondents to start working in their OSS project (multiple choices were allowed).

Results for SQ_{3.1}: The results suggest that UML is helpful for new contributors to get up to speed.

4.5.4.2 What are the impacts of using UML in OSS projects?

Because of their overview about the projects, we asked founders for their impression about the impacts of introducing UML into their project. Fig. 4.14a and Fig. 4.14b summarize the 190 answers for the two questions. A majority of respondents (65.79%) reported positive impacts, while only a few founders (<2%) encountered negative impacts. Only, 34% of the founders saw changes in the way the contributors worked after UML was introduced.

To find out more about the changes, we asked those who observed changes to describe the way the working routine had changed. We received 31 responses to the open ended question. Comments positive to UML can be summarized in following groups: i) Hiding complexity/improved overview (mentioned 18 times); ii) Improved communication/ reduced ambiguity (6 times); iii) Prevention of sub-standard implementations (5 times); iv) Improved scoping and partitioning of work (3 times); v) Improved/easier to implement designs (9 times); vi) Improved quality assurance (1 time); vii) Reduced architecture degradation (1 time).

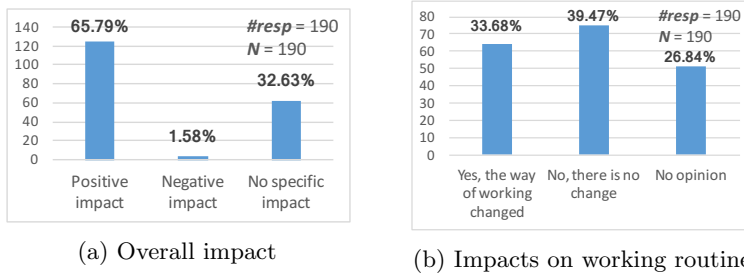


Figure 4.14: Impacts of introducing UML in OSS projects

We also received two answers describing negative changes, complaining about more work and the need for developers to learn UML notation.

Results for SQ_{3.2}: One third of respondents reported changes of the working routine due to UML, mainly in the planning phase, the development process and in communication. Most of the reported changes can be considered positive.

4.5.4.3 Can UML models help to attract new contributors?

We ask founders if they think that UML models help to attract new contributors to their projects. 190 founders answered this question. Fig. 4.15 shows the responses in detail. Only a few of the respondents (21.58%) believe that UML models can attract new contributors, while most of them think UML is not an attractive factor (47.37%).

We asked those who think UML models attract new contributors for reasons behind their thoughts. We received only 25 answers, including following arguments: a) UML models make the project and its goals easier to understand (mentioned 13 times), b) the potential of UML to help new contributors (by code comprehension) (7 times), c) visual documentation is considered attractive (3 times), and d) UML can support communication between old and new members (2 times).

It is worth mentioning that two of the projects have been based on executable UML diagrams (xtUML), therefore the diagrams were considered a magnet to contributors.

Two of the respondents who answered previously that UML is an attracting factor, mentioned additional factors, i.e., the personality, the quality of the model, and complexity of the project, e.g., “*I feel that it depends on two things: how perceptive the contributors are, and how elegantly and interesting the models [were] structured*”.

Results for SQ_{3.3}: Few founders think UML models attract new contributors to their projects.

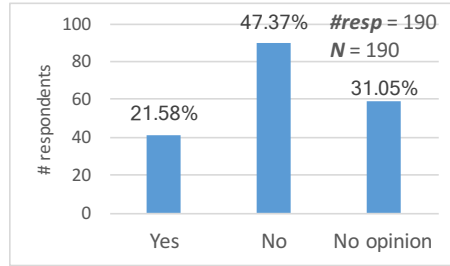


Figure 4.15: Do UML models attract new contributors to the project?

4.6 Discussions

In the following we discuss our insights in context of related works and implications of our results. Furthermore, we present the threats to validity.

4.6.1 Comparison to Insights to Related Works

In this section, our observations are compared with findings from related works.

Communication: The finding that UML is used for communication purposes within OSS fits with observations that were already made about the use of documentation by Kazman et al. [33] and sketches Chung et al. [23]. Furthermore, the results fit with the insights of Gorschek et al. [28], who also observed a use for communication within industrial and OSS programmers.

New contributors: The observation that new contributors seem to benefit from the use of UML confirms the first anecdotal evidence that Chung et al. collected [23]. Gorschek et al. found similar tendencies in their survey, where the use of models was found to be higher for novices [28].

Design and documentation: We could uncover a main similarity in the use of UML in OSS and industry, as we observed that UML is mainly used for design and documentation, and less for code generation within OSS. Similar observations had been made for industrial usage by Torchiano et al. [26] and Forward et al. [27].

Role splits: However, we also found a hint of a contrast in the use of UML. While we observed that the architectures defined within UML models are often implemented by multiple developers, as it happens within industry, we also observed that in most cases all these contributors had participated in the model creation. This seems to be in contrast to the practice in many industrial cases, where those who create the models are not necessarily the ones who create the code, as, e.g., observed by Kuhn et al. [31].

Finally, we made two observations that should be further studied, also in industry. *Passive benefits:* Many participants who do not create UML models consider its existence in the project beneficial. *Partial adoption:* Many models are only partially adopted during implementation. It would be interesting to see whether this conforms or is in contrast to industrial practice.

4.6.2 Implications

4.6.2.1 OSS practitioners

Use UML to coordinate team work! We know that UML is used in industry within teams - communicating and coordinating their work [6]. The insights from this paper indicate that this practice might actually also work to coordinate joint efforts within OSS teams with often remotely located developers.

4.6.2.2 OSS seniors

Provide UML to support your junior peers! In most investigated aspects the answers given by NUCs showed a slight tendency to be more positive about UML than the answers of UML contributors. Thus, it seems that models have an impact on teams that affects not just the model creators positively. We hope that OSS contributors feel motivated by these results to contributing more models. Furthermore, it seems that the usage of UML helps new contributors to get productive. This might be seen as an incentive for the introduction of UML.

4.6.2.3 Industrial companies

Adopt team-modeling! The observed contrast that most people implementing a model also participated in its creation, might be an interesting option for industrial practice, too. Especially, when agile practices are applied, models can be taken into the loop, e.g., as part of planing during Scrum meetings.

4.6.2.4 University teachers

Promote consumption as first experience when learning UML! Again, the mentioned slight tendency of NUCs to be more positive about UML is worth noting. It seems that the benefits of UML are more positive for consumers than for creators. This is to be confirmed in future studies. It can have today an impact on the way we teach modeling. Students still tend to learn modeling by creating models. Our results imply that it might be a good idea to let them consume models first.

4.6.3 Threats to Validity

In the following, we discuss internal and external threats to validity of our study as introduced by Marczyk et al. [103].

Internal validity Some threats that are generic to research that use GitHub data, as discussed by Kalliamvakou et al. [90], concern our study, too: First, a large amount of GitHub projects are not software development projects or have very few commits, only. Furthermore most GitHub projects are inactive (Kalliamvakou et al. guess that the amount of active projects is around 22%). To mitigate the impact of these threats on our study, we filtered the projects based on the number of commits and size. Since such filters are always just heuristics, it is probable that some of the remaining projects still are toy or educational projects. However, we consider the remaining threat acceptable,

since we can assume that the vast majority of the here studied projects are *real* software development projects.

We focus on projects that do use UML only, to ensure that questioned developers have the experience of working in a project with UML. To ensure nonetheless that persons that prefer to not use UML are not underrepresented, we sent the questionnaire not just to persons who manipulated UML, but also to contributors who did not change or introduce UML files (NUCs). Therefore, we believe that our results still provide valuable insights.

External validity Our study focuses on OSS projects in GitHub. While we do not expect a direct generalization of our results to closed source projects, we expect them to be mostly generalizable to OSS projects. 16.29% of the survey respondents had a PhD degree. This rate is higher than industry average. We expect them to be more positive about UML, making them more likely to have answered our questionnaire. Thus, there might be a selection bias towards projects that have PhDs as contributors. We do not know whether these projects are different in nature concerning our results. However, since this concerns only 16.29% of our data points, we believe that our results are nonetheless representative.

We did not limit the domain. However, there might be a bias towards the domain that comes with the use of UML. Since we study the impact of UML, when it is used, we consider our results valuable despite the possible bias in study domains. We only have a look at UML models that are stored as specific file formats. Although, it would be better to have a look at all possible representations of UML models that exist, the selected set of formats comprehends the standard ones (.uml and .xmi) and image files, being already broad and allows a first valuable insight. Finally, in this study, we do not distinguish between UML diagram types. We therefore do not conclude for single UML types but for UML in general.

4.7 Conclusion and Future work

In this paper we study the use of UML in open source, in order to identify commonalities and differences to the use of UML in industry. Therefore, we performed a survey with contributors from 458 GitHub projects that include UML files. Our study delivers some first insights that might help companies to decide whether to promote UML usage in open source projects. In favor of UML are the observations that UML actually helps new contributors and is generally perceived as supportive. However, UML does not seem to have the potential to attract new contributors. Further, we found that the use of UML in open source projects is partially similar to industrial use. However, there are also differences that should be considered when joining industrial projects with open source efforts. For example, the fact that there seems to be barely a split of roles between model creator and person implementing the modeled system. Furthermore, we found that many modeled designs are only partially followed during implementation.

Future works We only use a part of survey responses in this study (ignoring responses of short-time projects). In the future, we plan to compare whether the results for these projects are different from the ones we found. Furthermore, we plan to use meta data to investigate whether different aspects such as size, active time, and number of contributors of a project affect the use of models and the perception of developers within the projects. Nonetheless, our findings from this study are drawn for UML in general. We would love to enrich our dataset by classifying UML diagrams by diagram type. This will enable to see whether diagram types affect the use of UML, and what UML diagrams are in widest use.

4.8 Appendix 1. Distribution of survey respondents by countries

Table 4.2: Respondents by countries (Top 26)

Country	No. responses	Country	No. responses
United States	72	Russia	9
Germany	49	Austria	8
France	46	China, People's Republic of	8
Brazil	35	Czech Republic	8
Spain	26	India	8
United Kingdom	21	Belgium	7
Switzerland	20	Colombia	6
unknown	15	Slovakia	6
Canada	14	Sweden	6
Italy	12	Bulgaria	5
Netherlands	11	China, Republic of (Taiwan)	5
Argentina	9	Denmark	5
Poland	9	Finland	4

4.9 Appendix 2. Distribution of survey respondents by continents

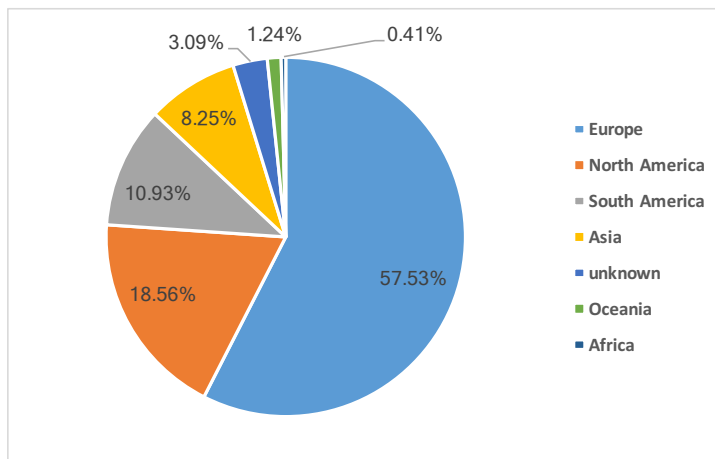


Figure 4.16: Distribution of survey respondents by continents

Bibliography

- [1] G. Booch, J. Rumbaugh, and I. Jacobson, “The Unified Modeling Language For Object-Oriented Development, Documentation Set Version 1.0,” 1997.
- [2] B. Anda, K. Hansen, I. Gullesen, and H. K. Thorsen, “Experiences from Introducing UML-based Development in a Large Safety-critical Project,” *Empirical Softw. Engg.*, vol. 11, no. 4, pp. 555–581, dec 2006.
- [3] B. Dobing and J. Parsons, “How UML is Used,” *Commun. ACM*, vol. 49, no. 5, pp. 109–113, may 2006.
- [4] P. Baker, S. Loh, and F. Weil, “Model-Driven Engineering in a large industrial context Motorola case study,” *Model Driven Engineering Languages and Systems*, pp. 476–491, 2005.
- [5] M. Grossman, J. E. Aronson, and R. V. McCarthy, “Does UML make the grade? Insights from the software development community,” *Information and Software Technology*, vol. 47, no. 6, pp. 383–397, 2005.
- [6] C. F. J. Lange, M. R. V. Chaudron, and J. Muskens, “In practice: UML software architecture and design description,” *IEEE Software*, vol. 23, no. 2, pp. 40–46, 2006.
- [7] W. J. Dzidek, E. Arisholm, and L. C. Briand, “A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 3, pp. 407–432, may 2008.
- [8] A. Nugroho and M. R. V. Chaudron, “A Survey into the Rigor of UML Use and Its Perceived Impact on Quality and Productivity,” in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2008, pp. 90–99.
- [9] G. Scanniello, C. Gravino, and G. Tortora, “Investigating the Role of UML in the Software Modeling and Maintenance-A Preliminary Industrial Survey.” in *ICEIS (3)*, 2010, pp. 141–148.
- [10] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, and R. Pretorius, “Empirical evidence about the UML: A systematic literature review,” pp. 363–392, 2011.
- [11] M. Petre, “UML in practice,” in *Proceedings - International Conference on Software Engineering*, 2013, pp. 722–731.

- [12] J. Lung, J. Aranda, S. M. Easterbrook, and G. V. Wilson, "On the Difficulty of Replicating Human Subjects Studies in Software Engineering," in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 191–200.
- [13] J. Liebowitz, *Strategic intelligence: business intelligence, competitive intelligence, and knowledge management*. CRC Press, 2006.
- [14] H. Störrle, R. Hebig, and A. Knapp, "An Index for Software Engineering Models," in *International Conference on Model Driven Engineering Languages and Systems (MoDELS) 2014*, 2014, pp. 36–40.
- [15] C. Kobryn, "UML 2001: A Standardization Odyssey," *Commun. ACM*, vol. 42, no. 10, pp. 29–37, oct 1999.
- [16] G. Booch, *Object Oriented Design with Applications*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1991.
- [17] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-oriented Modeling and Design*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.
- [18] I. Jacobson, *Object-oriented software engineering: a use case driven approach*. Pearson Education India, 1993.
- [19] R. France, J. Bieman, and B. H. C. Cheng, *Repository for Model Driven Development (ReMoDD)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 311–317.
- [20] B. Karasneh and M. R. V. Chaudron, "Online Img2UML Repository: An Online Repository for UML Models." in *EESMOD@ MoDELS*, 2013, pp. 61–66.
- [21] K. Yatani, E. Chung, C. Jensen, and K. N. Truong, "Understanding how and why open source contributors use diagrams in the development of Ubuntu," *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, p. 995, 2009.
- [22] H. Osman and M. R. V. Chaudron, "UML Usage in Open Source Software Development : A Field Study," in *Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013)*, 2013, pp. 23–32.
- [23] E. Chung, C. Jensen, K. Yatani, V. Kuechler, and K. N. Truong, "Sketching and Drawing in the Design of Open Source Software," in *Proc. VL/HCC*, 2010, pp. 195–202.
- [24] W. Ding, P. Liang, A. Tang, H. Van Vliet, and M. Shahin, "How do open source communities document software architecture: An exploratory survey," in *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 2014, pp. 136–145.

- [25] A. M. Fernández-Sáez, M. Genero, and M. R. Chaudron, “Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study,” *Information and Software Technology*, vol. 55, no. 7, pp. 1119–1142, 2013.
- [26] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, “Relevance, benefits, and problems of software modelling and model driven techniques - A survey in the Italian industry,” *Journal of Systems and Software*, vol. 86, no. 8, pp. 2110–2126, 2013.
- [27] A. Forward, O. Badreddin, and T. C. Lethbridge, “Perceptions of software modeling: a survey of software practitioners,” in *5th workshop from code centric to model centric: evaluating the effectiveness of MDD (C2M: EEMDD)*, 2010.
- [28] T. Gorschek, E. Tempero, and L. Angelis, “On the use of software design models in software development practice: An empirical investigation,” *Journal of Systems and Software*, vol. 95, pp. 176–193, 2014.
- [29] A. Nugroho and M. Chaudron, “A Survey of the Practice of Design – Code Correspondence amongst Professional Software Engineers,” *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 467–469, 2007.
- [30] A. Nugroho and M. R. V. Chaudron, “Evaluating the Impact of UML Modeling on Software Quality : An Industrial Case Study,” *Springer-Verlag*, pp. 181–195, 2009.
- [31] A. Kuhn, G. C. Murphy, and C. A. Thompson, *An Exploratory Study of Forces and Frictions Affecting Large-Scale Model-Driven Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 352–367.
- [32] O. Badreddin, T. C. Lethbridge, and M. Elassar, “Modeling Practices in Open Source Software,” *Open Source Software: Quality Verification - 9th IFIP WG 2.13 International Conference, OSS 2013*, pp. 127–139, 2013.
- [33] R. Kazman, D. Goldenson, I. Monarch, W. Nichols, and G. Valetto, “Evaluating the Effects of Architectural Documentation: A Case Study of a Large Scale Open Source Project,” *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 220–260, mar 2016.
- [34] P. Langer, T. Mayerhofer, M. Wimmer, and G. Kappel, “On the usage of UML: Initial results of analyzing open UML models,” *Modellierung 2014*, vol. P225, pp. 289–304, 2014.
- [35] G. D. Crnkovic, *Constructive Research and Info-computational Knowledge Generation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 359–380.
- [36] G. Gousios and D. Spinellis, “GHTorrent: Github’s data from a fire-hose,” in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*. IEEE, 2012, pp. 12–21.

- [37] G. Robles, S. Koch, J. M. González-BARAHONA, and J. Carlos, “Remote analysis and measurement of libre software systems by means of the CVSAnalY tool,” in *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS)*. IET, 2004, pp. 51–56.
- [38] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [39] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, *Selecting Empirical Methods for Software Engineering Research*. London: Springer London, 2008, pp. 285–311.
- [40] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques (3rd edition)*. Elsevier, 2011.
- [41] B. Karasneh, M. R. V. Chaudron, F. Khomh, and Y. G. Gueheneuc, “Studying the Relation between Anti-Patterns in Design Models and in Source Code,” in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, mar 2016, pp. 36–45.
- [42] C. F. J. Lange and M. R. V. Chaudron, “Managing model quality in UML-based software development,” in *Proceedings - 13th IEEE International Workshop on Software Technology and Engineering Practice, STEP 2005*, ser. STEP '05, vol. 2005. Washington, DC, USA: IEEE Computer Society, 2005, pp. 7–16.
- [43] ISO 25010:2011, “Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” International Organization for Standardization, Geneva, CH, Standard ISO 25010:2011, mar 2011.
- [44] J. Bansiya and C. G. Davis, “A hierarchical model for object-oriented design quality assessment,” *IEEE Transactions on software engineering*, vol. 28, no. 1, pp. 4–17, 2002.
- [45] J. A. McCall, P. K. Richards, and G. F. Walters, “Concepts and definitions of software quality,” *Factors in Software Quality, NTIS*, vol. 1, 1977.
- [46] B. W. Boehm, J. R. Brown, and M. Lipow, “Quantitative evaluation of software quality,” in *Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press, 1976, pp. 592–605.
- [47] P. Pietsch, D. Reuling, U. Kelter, J. Folmer, and B. Vogel-Heuser, “Experiences on the Quality and Availability of Test Models for Model Differencing Tools,” in *FMI 2014-Free Models Initiative Workshop Proceedings*, 2014, p. 11.
- [48] J. Kramer, “Is abstraction the key to computing?” *Communications of the ACM*, vol. 50, no. 4, pp. 36–42, 2007.

- [49] D. R. Stikkolorum, C. Stevenson, and M. R. V. Chaudron, “Assessing Software Design Skills and their Relation with Reasoning Skills.” in *EduSymp@ MoDELS*, 2013, pp. 1–8.
- [50] F. Leung and N. Bolloju, “Analyzing the quality of domain models developed by novice systems analysts,” in *System Sciences, 2005. HICSS’05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE, 2005, pp. 188b—188b.
- [51] B. Karasneh, R. Jolak, and M. R. V. Chaudron, “Using Examples for Teaching Software Design: An Experiment Using a Repository of UML Class Diagrams,” in *Software Engineering Conference (APSEC), 2015 Asia-Pacific*. IEEE, 2015, pp. 261–268.
- [52] K. C. Thramboulidis, “Using UML in control and automation: a model driven approach,” in *Industrial Informatics, 2004. INDIN ’04. 2004 2nd IEEE International Conference on*, jun 2004, pp. 587–593.
- [53] C. Secchi, C. Fantuzzi, and M. Bonfe, “On the Use of UML for Modeling Physical Systems,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, apr 2005, pp. 3990–3995.
- [54] R. P. L. Buse and T. Zimmermann, “Information Needs for Software Development Analytics,” in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE ’12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 987–996.
- [55] “Enterprise Architect,” <http://www.sparxsystems.com/>.
- [56] “Visual Paradigm,” <http://www.visual-paradigm.com/>.
- [57] B. Karasneh and M. R. V. Chaudron, “Extracting UML models from images,” in *2013 5th International Conference on Computer Science and Information Technology*, mar 2013, pp. 169–178.
- [58] B. Karasneh and M. R. V. Chaudron, “Img2UML: A System for Extracting UML Models from Images,” in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, sep 2013, pp. 134–137.
- [59] E. P. Costa, A. C. Lorena, A. Carvalho, and A. A. Freitas, “A review of performance evaluation measures for hierarchical classifiers.” in *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, C. Drummond, W. Elazmeh, N. Japkowicz, and S. A. Macskassy, Eds. AAAI Press, jul 2007, pp. 182–196.
- [60] D. Blostein, E. Lank, and R. Zanibbi, *Treatment of Diagrams in Document Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 330–344.
- [61] D. Lu and Q. Weng, “A Survey of Image Classification Methods and Techniques for Improving Classification Performance,” *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, jan 2007.

- [62] J. A. Shine and D. B. Carr, "A comparison of classification methods for large imagery data sets," *JSM*, pp. 3205–3207, 2002.
- [63] A. Mishchenko and N. Vassilieva, "Model-based chart image classification," in *International Symposium on Visual Computing*. Springer, 2011, pp. 476–485.
- [64] B. T. Messmer and H. Bunke, "Automatic learning and recognition of graphical symbols in engineering drawings," in *International Workshop on Graphics Recognition*. Springer, 1995, pp. 123–134.
- [65] E. Lank, J. S. Thorley, and S. J.-S. Chen, "An interactive system for recognizing hand drawn UML diagrams," in *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 2000, p. 7.
- [66] T. Hammond and R. Davis, "Tahuti: A geometrical sketch recognition system for uml class diagrams," in *ACM SIGGRAPH 2006 Courses*. ACM, 2006, p. 25.
- [67] E. Lank, J. Thorley, S. Chen, and D. Blostein, "On-line recognition of UML diagrams," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*. IEEE, 2001, pp. 356–360.
- [68] L. Fu and L. B. Kara, "From engineering diagrams to engineering models: Visual recognition and applications," *Computer-Aided Design*, vol. 43, no. 3, pp. 278–292, 2011.
- [69] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.
- [70] S. Suzuki and Others, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.
- [71] J. C. Russ, *The Image Processing Handbook (3rd Ed.)*. Boca Raton, FL, USA: CRC Press, Inc., 1999.
- [72] M. A. Hall, "Correlation-based feature selection for machine learning," Hamilton, Tech. Rep., 1999.
- [73] "Waikato Environment for Knowledge Analysis (WEKA)," <http://www.cs.waikato.ac.nz/ml/weka/>.
- [74] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.
- [75] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

- [76] A. Herout, M. Dubská, and J. Havel, “Review of hough transform for line detection,” in *Real-Time Detection of Lines and Grids*. Springer, 2013, pp. 3–16.
- [77] D. Lagunovsky and S. Ablameyko, “Fast line and rectangle detection by clustering and grouping,” in *International Conference on Computer Analysis of Images and Patterns*. Springer, 1997, pp. 503–510.
- [78] K. Murakami and T. Naruse, “High speed line detection by Hough transform in local area,” in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 3. IEEE, 2000, pp. 467–470.
- [79] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.
- [80] T. Ho-Quang, M. Chaudron, I. Samuelsson, J. Hjaltason, B. Karasneh, and H. Osman, “Automatic classification of UML Class diagrams from images,” in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 1, 2014.
- [81] G. Reggio, M. Leotta, and F. Ricca, “Who Knows/Uses What of the UML: A Personal Opinion Survey,” in *Model-Driven Engineering Languages and Systems*. Springer, 2014, pp. 149–165.
- [82] G. Robles, J. M. Gonzalez-Barahona, and J. J. Merelo, “Beyond source code: the importance of other artifacts in software development (a case study),” *Journal of Systems and Software*, vol. 79, no. 9, pp. 1233–1248, 2006.
- [83] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, “On the variation and specialisation of workload - a case study of the Gnome ecosystem community,” *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, 2014.
- [84] S. McIntosh, B. Adams, and A. E. Hassan, “The evolution of ant build systems,” in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 2010, pp. 42–51.
- [85] S. McIntosh, B. Adams, T. H. D. Nguyen, Y. Kamei, and A. E. Hassan, “An empirical study of build maintenance effort,” in *Proceedings of the 33rd international conference on software engineering*. ACM, 2011, pp. 141–150.
- [86] Y. Jiang and B. Adams, “Co-evolution of Infrastructure and Source Code - An Empirical Study,” in *12th {IEEE/ACM} Working Conference on Mining Software Repositories, {MSR} 2015, Florence, Italy, May 16-17, 2015*, 2015, pp. 45–55.
- [87] G. Robles, J. M. González-Barahona, D. Izquierdo-Cortazar, and I. Her- raiz, “Tools for the study of the usual data sources found in libre software projects,” *International Journal of Open Source Software and Processes*, vol. 1, no. 1, pp. 24–45, 2009.

- [88] R. Hebig, T. Ho-Quang, G. Robles, and M. R. V. Chaudron, “List of identified projects with UML and replication package,” <http://oss.models-db.com>.
- [89] 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.
- [90] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The Promises and Perils of Mining GitHub,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 92–101.
- [91] C. Gacek and B. Arief, “The many meanings of open source,” *IEEE software*, vol. 21, no. 1, pp. 34–40, 2004.
- [92] B. Fitzgerald, “The transformation of open source software,” *Mis Quarterly*, pp. 587–598, 2006.
- [93] D. M. German, “The GNOME project: a case study of open source, global software development,” *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 201–215, 2003.
- [94] D. Riehle, “The economic case for open source foundations,” *Computer*, vol. 43, no. 1, pp. 86–90, 2010.
- [95] Ø. Hauge, C. Ayala, and R. Conradi, “Adoption of open source software in software-intensive organizations—A systematic literature review,” *Information and Software Technology*, vol. 52, no. 11, pp. 1133–1154, 2010.
- [96] K. Crowston, K. Wei, J. Howison, and A. Wiggins, “Free/Libre open-source software development: What we know and what we do not know,” *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, p. 7, 2012.
- [97] K.-J. Stol, M. A. Babar, P. Avgeriou, and B. Fitzgerald, “A comparative study of challenges in integrating Open Source Software and Inner Source Software,” *Information and Software Technology*, vol. 53, no. 12, pp. 1319–1336, 2011.
- [98] D. Spinellis and C. Szyperski, “How is open source affecting software development?” *IEEE Software*, vol. 21, no. 1, p. 28, 2004.
- [99] C. Hauff and G. Gousios, “Matching GitHub developer profiles to job advertisements,” in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 362–366.
- [100] R. Hebig, T. Ho-Quang, M. Chaudron, G. Robles, and M. Fernandez, “The quest for open source projects that use UML: Mining GitHub,” in *Proceedings - 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2016*, 2016.

-
- [101] I. S. Wiese, I. Steinmacher, C. Treude, J. T. D. Silva, and M. Gerosa, “Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant,” in *Proceedings of the 32nd International Conference on Software Maintenance and Evolution*, 2016.
- [102] C. B. Seaman, “Qualitative methods in empirical studies of software engineering,” *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.
- [103] G. Marczyk, D. DeMatteo, and D. Festinger, *Essentials of research design and methodology*. John Wiley & Sons Inc, 2005.