Practices and Perceptions of UML Use in Open Source Projects

Truong Ho-Quang*, Regina Hebig*, Gregorio Robles†, Michel R.V. Chaudron*, Miguel Angel Fernandez†
*Chalmers — Göteborg University, Göteborg, Sweden
{truongh, hebig, chaudron}@chalmers.se
†GSyC/LibreSoft, Universidad Rey Juan Carlos, Madrid, Spain
grex@gsyc.urjc.es, mafesan.nsn@gmail.com

***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**

I. 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 [1]. 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 [3], or some companies were created around some communities [4]. Many projects created foundations to serve as an umbrella to collaborate and integrate software industry partners [5].

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 [6]. On the other hand, OSS has embraced practices from industry, such as (modern) code review practices and planning and requirements analysis mechanisms [7]. Companies with a large pool of developers try to have an "internal" OSS-like ecosystem, a concept coined as inner source [8]. 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...) [9]. And the software industry is looking into popular OSS repositories, such as GitHub, to find suitable candidates to fill open development positions [10].

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 [11]. OSS is known to be programming-driven, with other tasks having room for further improvement [12]. However, modeling is used in major companies [14]. 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 [11]. 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 II,

then introduce related work in Section III and describe our research method in Section IV. Section V presents our findings. Our findings, possible threats to validity and implications of our research are discussed in Section VI. Conclusions can be found in Section VII.

II. 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 todays software intensive industry [13]. 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?

III. Related work

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

A. 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 [14]. 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 [16]. Gorschek et al. focused on a different population, which are programmers, partially working in industry and OSS [15]. 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 [17]. 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 [18]. 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 [13].

B. 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 [19]. Similarly, Ding et al. found only 19 projects with UML when manually studying 2,000 OSS projects [20]. However, in our previous work, we presented an approach that allows to find thousands of projects with UML by mining GitHub [11].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 [23]. 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 [24]. Finally, Langer et al. studied the lifespan of 121 enterprise architect models in OSS projects [25].

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.

IV. Research Methodology

In this section, we describe our study method in detail. The overall process is shown in Fig. 1.

A. 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 [11]. 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.

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 [26]. 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.

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 [11]. At the end of this step, we had 93,648 UML files from 24,797 repositories.

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 [28], 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.

B. 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 [29] or identification of contributors [30].

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 then 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 package[1].

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 [30]. After running the script, the original 129,276 contributors result in 99,319 distinct ones.

C. Conducting the survey

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

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)

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.

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.

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

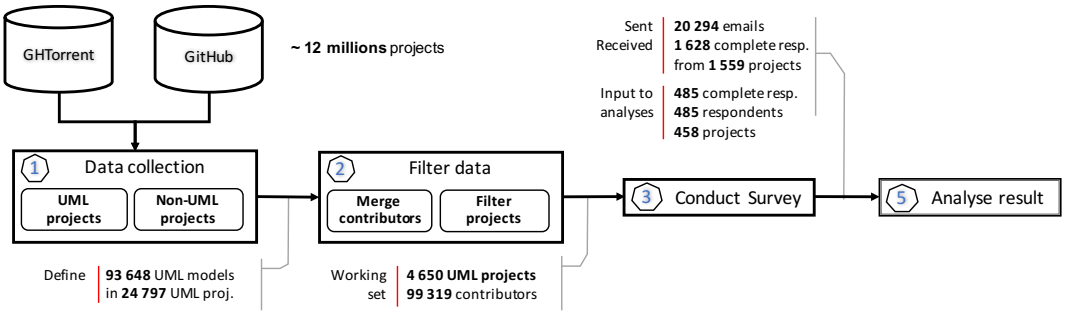


Figure 1: Overall process

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 tool[2] 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.

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.

D. 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 [31]. 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".

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

	Founder			Non-Founder			SUM
	IUC	*NUC*	*UC*	*IUC*	*NUC*	*UC*	
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.	*84*	*79*	*27*	*176*	*80*	*39*	*485*
Percent(%)	17.3	16.3	5.6	36.3	16.5	8.0	100

V. Results/Findings

A. 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

[2]LimeSurvey homepage: https://www.limesurvey.org/

of respondents from 458 projects. Among the 485 respondents, 190 (about 40%) are founders of an OSS project and 159 (32.8%) are non-UML contributors (Table I). Regarding the educational background (as shown in Fig. 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. 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.

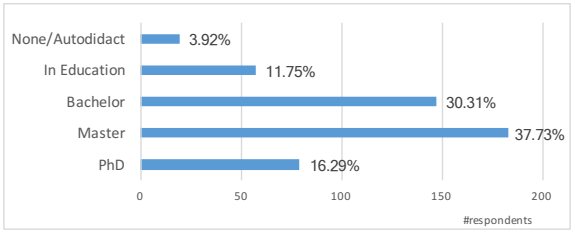


Figure 2: Distribution of respondents based on their highest educational background

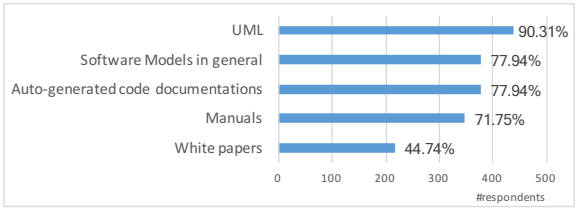


Figure 3: Familiar architecture document formats (multiple choices were allowed)

B. Why is UML used?

1) What are the motivations to use UML modeling?:

Fig. 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. 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.

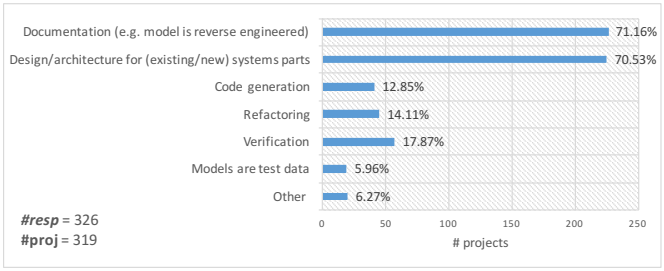


Figure 4: Intent of UML models that were added/updated

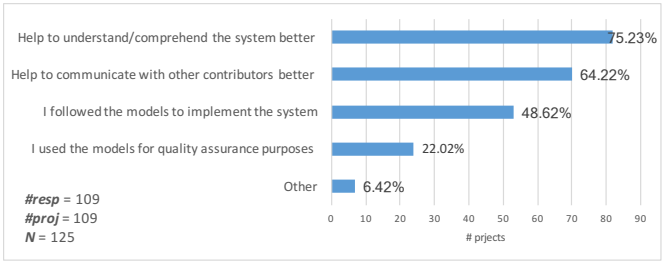


Figure 5: How did UML help non-UML contributors?

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.*

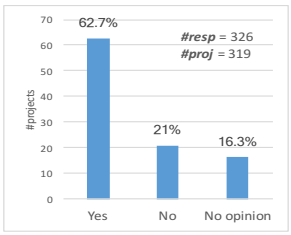
2) What are the reasons not to use UML in projects?: To complement our finding on the motivations to introduce/use UML, we asked the 16 NUCs who did not find UML models useful the reasons for this. Respondents from 6 projects actually had not used models, finding themselves not required to learn/use 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).

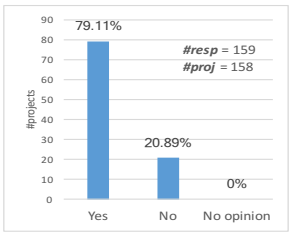
Results for $SQ_{1.2}$*: Only a small number of respondents found UML not useful.*

C. Is UML part of the interaction of contributors?

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. 6a). In



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



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

Figure 6: Awareness of developers about the existence of UML in their projects (by project)

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. 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.*

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. 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. 8).

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.*

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

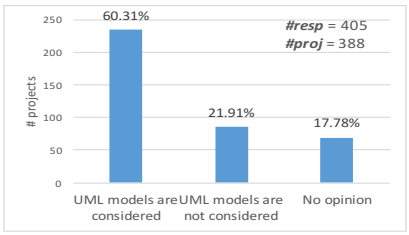


Figure 7: Are the UML model(s) considered in the communication between contributors? (per project)

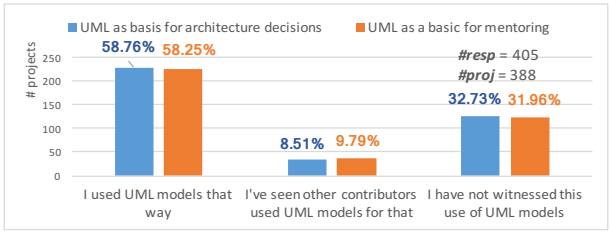


Figure 8: Is UML a basis for architectural decisions or mentoring activities? (per project)

asked the question "Was the UML model adopted during the implementation phase?". Fig. 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%).

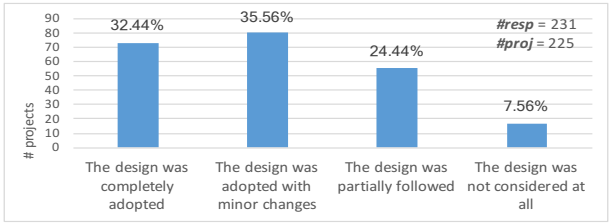


Figure 9: Was the UML model adopted during the implementation phase? (by project)

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. 10 and Fig. 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 that 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.

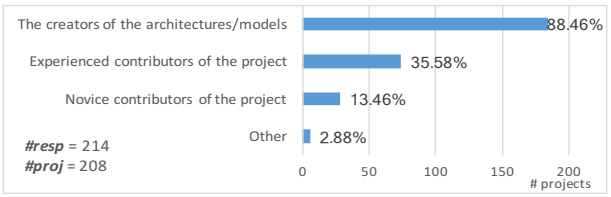


Figure 10: Who implemented the UML models? (by project)

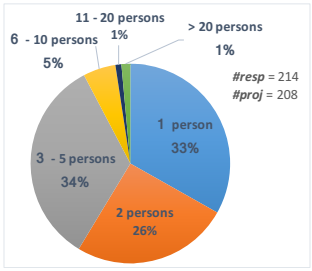
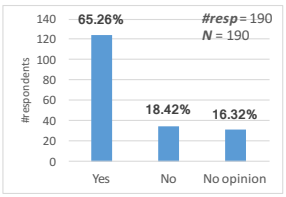
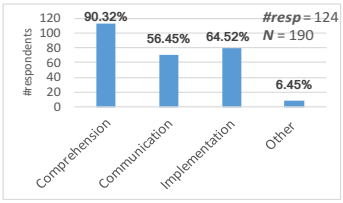


Figure 11: Number of contributors who implemented UML models in a project



(a) Do UML models help new contributors?



(b) For what tasks do models help?

Figure 12: Responses for the questions whether UML models help new contributors to join a project.

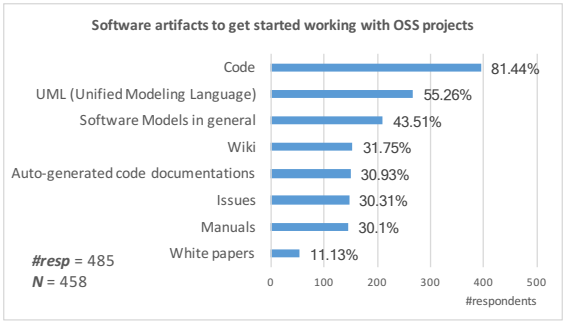


Figure 13: Software artifacts used by respondents to start working in their OSS project (multiple choices were allowed).

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.*

D. What is the impact/benefit of UML?

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. 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 V-A), source code still remains their first choice to start working with an OSS project (81%) - see Fig. 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.

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

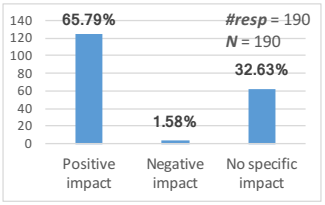
2) What are the impacts of using UML in OSS projects?:

Because of their overview about the projects, we asked founders for their impression about the impacts of introducing UML into their project. Fig. 14a and Fig. 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.

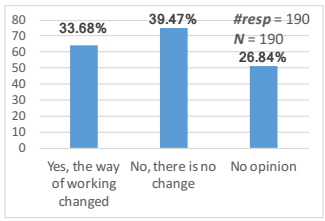
To find out more about the changes, we asked those who observed changes to describe the way the working routine had changed. We received 31 responses to the open ended question. Comments positive to UML can be summarized in following groups: i) Hiding complexity/improved overview (mentioned 18 times); 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).

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.*



(a) Overall impact



(b) Impacts on working routine

Figure 14: Impacts of introducing UML in OSS projects

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. 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"*.

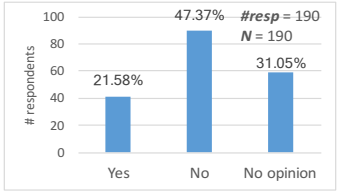


Figure 15: Do UML models attract new contributors to the project?

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

VI. 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.

A. 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. [23] and sketches Chung et al. [24]. Furthermore, the results fit with the insights of Gorschek et al. [15], 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 [24]. Gorschek et al. found similar tendencies in their survey, where the use of models was found to be higher for novices [15].

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. [14] and Forward et al. [16].

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. [13].

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.

B. Implications

1) OSS practitioners: Use UML to coordinate team work! We know that UML is used in industry within teams - communicating and coordinating their work [33]. 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.

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.

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) 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.

C. Threats to Validity

In the following, we discuss internal and external threats to validity of our study as introduced by Marczyk et al. [32].

Internal validity: Some threats that are generic to research that use GitHub data, as discussed by Kalliamvakou et al. [29], 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.

VII. 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.

Acknowledgment

We are very grateful to all participants of the study for taking the time and sharing their experience. G. Robles acknowledges the eMadrid (S2013/ICE-2715) and NUBOMEDIA (FP7-ICT-2013-1.6, GA-610576) projects.

REFERENCES

[1] Cristina Gacek and Budi Arief. The many meanings of open source. *IEEE software*, 21(1):34–40, 2004.

[2] Kevin Carillo and Chitu Okoli. The open source movement: a revolution in software development. *Journal of Computer Information Systems*, 49(2):1–9, 2008.

[3] Brian Fitzgerald. The transformation of open source software. *Mis Quarterly*, pages 587–598, 2006.

[4] Daniel M German. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice*, 8(4):201–215, 2003.

[5] Dirk Riehle. The economic case for open source foundations. *Computer*, 43(1):0086–90, 2010.

[6] Øyvind Hauge, Claudia Ayala, and Reidar Conradi. Adoption of open source software in software-intensive organizations–a systematic literature review. *Information and Software Technology*, 52(11):1133–1154, 2010.

[7] Kevin Crowston, Kangning Wei, James Howison, and Andrea Wiggins. Free/libre open-source software development: What we know and what we do not know. *ACM Computing Surveys*, 44(2):7, 2012.

[8] Klaas-Jan Stol, Muhammad Ali Babar, Paris Avgeriou, and Brian Fitzgerald. A comparative study of challenges in integrating open source software and inner source software. *Information and Software Technology*, 53(12):1319–1336, 2011.

[9] Diomidis Spinellis and Clemens Szyperski. How is open source affecting software development? *IEEE Software*, 21(1):28, 2004.

[10] Claudia Hauff and Georgios Gousios. Matching GitHub developer profiles to job advertisements. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 362–366. IEEE Press, 2015.

[11] Regina Hebig, Truong Ho-Quang, Gregorio Robles, Michel R.V. Chaudron, and Miguel Angel Fernandez. The quest for open source projects that use UML: Mining GitHub. *International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, 2016.

[12] Gregorio Robles, Jesus M Gonzalez-Barahona, and Juan Julian Merelo. Beyond source code: the importance of other artifacts in software development (a case study). *Journal of Systems and Software*, 79(9):1233–1248, 2006.

[13] Adrian Kuhn, Gail C Murphy, and C Albert Thompson. An exploratory study of forces and frictions affecting large-scale model-driven development. In *International Conference on Model Driven Engineering Languages and Systems*, pages 352–367. Springer, 2012.

[14] Marco Torchiano, Federico Tomassetti, Filippo Ricca, Alessandro Tiso, and Gianna Reggio. Relevance, benefits, and problems of software modelling and model driven techniques - A survey in the Italian industry. *Journal of Systems and Software*, 86(8):2110–2126, 2013.

[15] Tony Gorschek, Ewan Tempero, and Lefteris Angelis. On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, 95:176–193, 2014.

[16] Andrew Forward, Omar Badreddin, and Timothy 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*, 2010.

[17] Paul Baker, Shiou Loh, and Frank Weil. Model-driven engineering in a large industrial contextmotorola case study. In *International Conference on Model Driven Engineering Languages and Systems*, pages 476–491. Springer, 2005.

[18] Ariadi Nugroho and Michel RV Chaudron. Evaluating the impact of UML modeling on software quality: An industrial case study. In *International Conference on Model Driven Engineering Languages and Systems*, pages 181–195. 2009.

[19] Omar Badreddin, Timothy C. Lethbridge, and Maged Elassar. *Modeling Practices in Open Source Software*, pages 127–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] Wei Ding, Peng Liang, Anthony Tang, Hans Van Vliet, and Mojtaba Shahin. How do open source communities document software architecture: An exploratory survey. In *Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on*, pages 136–145. IEEE, 2014.

[21] Koji Yatani, Eunyoung Chung, Carlos Jensen, and Khai N Truong. Understanding how and why open source contributors use diagrams in the development of Ubuntu. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 995–1004. ACM, 2009.

[22] Mohd Hafeez Osman and Michel 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)*, pages 23–32, 2013.

[23] 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*, 42(3):220–260, 2016.

[24] Eunyoung Chung, Carlos Jensen, Koji Yatani, Victor Kuechler, and Khai N Truong. Sketching and drawing in the design of open source software. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 195–202. IEEE, 2010.

[25] Philip Langer, Tanja Mayerhofer, Manuel Wimmer, and Gerti Kappel. On the usage of UML: Initial results of analyzing open UML models. In *Modellierung*, 19, page 21, 2014.

[26] Georgios Gousios and Diomidis Spinellis. GHTorrent: GitHub's data from a firehose. In *9th IEEE working conference on Mining Software Repositories*, pages 12–21. 2012.

[27] Hudson Borges, André C. Hora, and Marco Tulio Valente. Understanding the factors that impact the popularity of GitHub repositories. *CoRR*, abs/1606.04984, 2016.

[28] Gregorio Robles, Jesús M González-Barahona, Daniel Izquierdo-Cortazar, and Israel Herraiz. Tools for the study of the usual data sources found in libre software projects. *International Journal of Open Source Software and Processes (IJOSSP)*, 1(1):24–45, 2009.

[29] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining GitHub. In *Proc of the 11th Working Conference on Mining Software Repositories*, pages 92–101, New York, NY, USA, 2014. ACM.

[30] Igor Scaliante Wiese, Igor Steinmacher, Christoph Treude, Jose Teodoro Da Silva, and Marco Gerosa. Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant. In *Proc of the 32nd Intl Conf on Software Maintenance and Evolution*, 2016.

[31] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering*, 25(4):557–572, 1999.

[32] Marczyk G, DeMatteo D, Festinger D. *Essentials of research design and methodology*. John Wiley & Sons Inc., 2005.

[33] C. F. J. Lange, M. R. V. Chaudron and J. Muskens. *In practice: UML software architecture and design description*, in IEEE Software, vol. 23, no. 2, pp. 40-46, March-April 2006.