The Quest for Open Source Projects that use UML

Mining GitHub

Regina Hebig, Truong Ho Quang,
Michel R.V. Chaudron
Chalmers | Göteborg University
{hebig,truongh,michel.chaudron}@cse.gu.se

Gregorio Robles,
Miguel Angel Fernandez
GSyC/LibreSoft
Universidad Rey Juan Carlos, Madrid, Spain
grex@gsyc.urjc.es, mafesan.nsn@gmail.com

ABSTRACT

Context: While industrial use of UML was studied intensely, 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. For 12% of the models duplicates were found, which 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

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 [2]. 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 [1]. 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 [19]. For example, the Repository for Model Driven Development (ReMoDD)[5] 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.

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. While there are many other ways to access GitHub data (GHTorrent or the GitHub API) obtaining data on UML usage is not trivial (as we will show).

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 [6] 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

MODELS '16, October 02-07, 2016, Saint-Malo, France

DOI: http://dx.doi.org/10.1145/2976767.2976778

formats and one for images, which is a state of the research technology that just became available in 2014 [8]. Finally, with the retrieved list of UML models, the git repositories of these projects were accessed 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 2, we formulate a number of research questions. Section 3 shows our review on relevant works. We describe our study approach in detail in Section 4. Our findings are presented and discussed in Section 5 and Section 6, respectively, including the threats to validity. We conclude our paper in Section 8.

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 results 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. RELATED RESEARCH

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

3.1 Use of UML in FOSS

Studies on the usage of UML are frequently done amongst in industry (mostly through surveys) [16, 20]. However, only few studies focus on freely available models, such as can be found in open source projects. Reggio et al. [16] 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. [11] use a crawling approach to automatically fill an online repository[1] with so far more than 700 model images- 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 [19][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.

Further, there are some works addressing small numbers of case studies of modeling in open source projects. Yatani et al. [23] 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. [15] 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

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

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

models are updated.

Finally, there are three works that actually approach a quantitative investigation of models in open source projects. Chung et al. [3] 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 by Ding et al. [4] 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.

The work that is probably closest to our study is the one of Langer et al. [12] 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 and GitHub. They identified 121 models. They further assessed the model lifespan (between introduction and last update) to be in average 1 247 days. 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.

3.2 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 [18]. By doing so, research has shed some light on the variation and specialization of workload that exist in FOSS communities [21].

The study of specific file formats that are non-source code can be found as well in the research literature: McIntosh et al. [13], [14] have investigated the build system for its evolution and effort, or the analysis of infrastructure as code that has become mainstream in the last years [9].

4. METHODOLOGY

In this section, we describe our study approach. The overall process is shown in Figure 1.

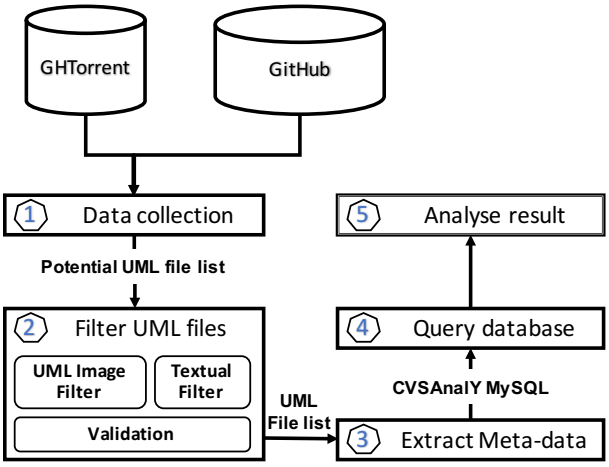


Figure 1: Overall process

First, we obtained a list of 10% of the GitHub repositories from GHTorrent [6] that are not forks. This resulted in a list of files of 1 240 000 repositories, those that had a branch that could be downloaded. 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 4.1 and Section 4.2 describe our approach and used filters in detail.

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. We describe the classification method in Section 4.3.

We have then obtained the meta-data from those repositories where a UML file has been identified by means of using the CVSAnalY tool [17] (step 3). Section 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 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 [7].

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 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 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 tool 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.

4.2 Data Collection

For all repositories from GHTorrent [6] 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 4.1, different file formats need to be taken into account. However, as not every image file is

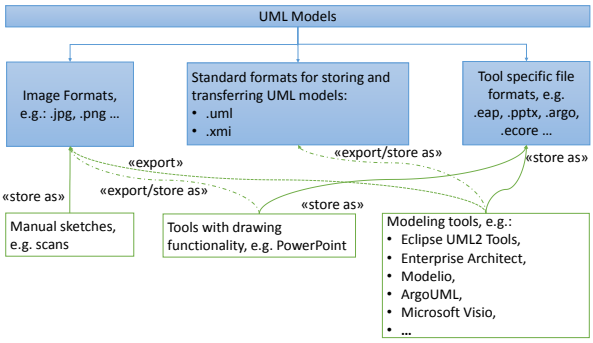


Figure 2: There is a large variety of tools for creating and formats for storing UML models

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.

Image files as well as standard formats are more common and are created by most modeling tools. For such common tools, developing an approaches to identify UML has a good cost-benefit ratio. The applied methods are explained below in section 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 WikiPedia[3], 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"]

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.

4.3 UML filters

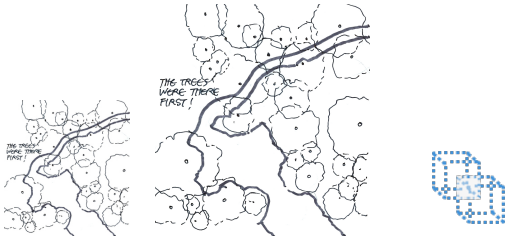
At this stage, the files obtained from Step 1 were checked if they really contain UML notation.

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

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 3a, 3b and 3c show examples of such images.



(a) Dup 1a (b) Dup 1b (c) icon

Figure 3: Example of duplicates and icon-size 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 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 [8]. 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".

[4]https://similarimagesfinder.codeplex.com/

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:

1. 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.
2. 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.
3. 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 comparing hash values of the file contents.

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 [17]. 100 repositories from the initial list could not be retrieved, due to various reasons, e.g. changes from public 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 in feasible, 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 metadata 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 used too query the database and aggregate the data required to answer the RQs. This final step took 14 days.

5. RESULTS

This section presents the results of our investigation. In 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 package[5].

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 the are included in a project (July/August 2015) to the time where the git repositories where 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 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 models[6], 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[7].

Table 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.

5.2 RQ2: Versions of UML models

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

Table 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

[5]Replication package http://oss.models-db.com

[6]One 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.

Table 2: Distribution of files / projects by number of updates

number of updates	models in projects with 1 to 99 models	models in projects with $\geq$ 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

on average 3,0 times (although the median, which is more significant given the skewed distribution, is 1 time). Furthermore, Table 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.

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

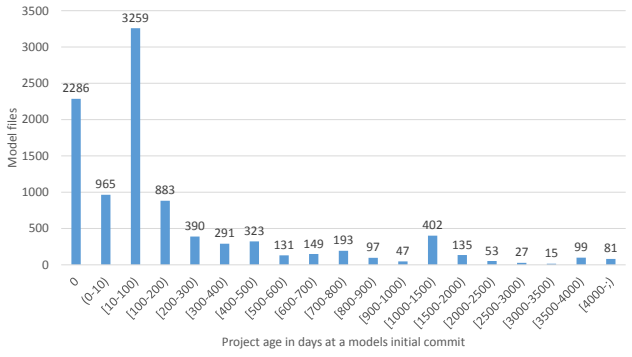


Figure 4: Distribution of model files sorted by project's age in days when the diagram was introduced (models within projects that have less than 100 models)

5.3 RQ3: Time of UML model introduction

Figure 4 shows the dates of the introduction models considering the amount of days since the start of the project, while Figure 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 separately, since these projects with partially more than 1 000 models would easily bias the presented view.

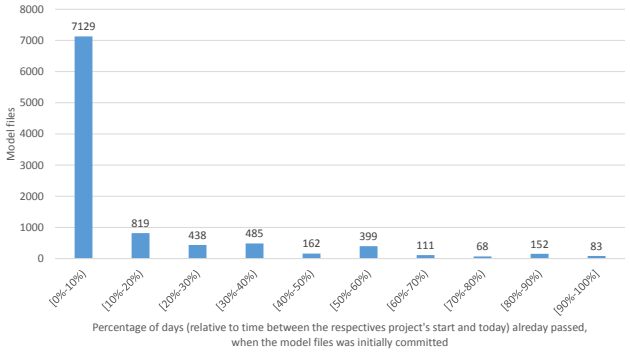


Figure 5: Distribution of model files sorted by percentage of project time that passed when the UML file was introduced (for projects that have less than 100 models)

However, we found that calender 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 6 shows the distribution of the models based on the time of their introduction when measured by the percentage of the project's commits.

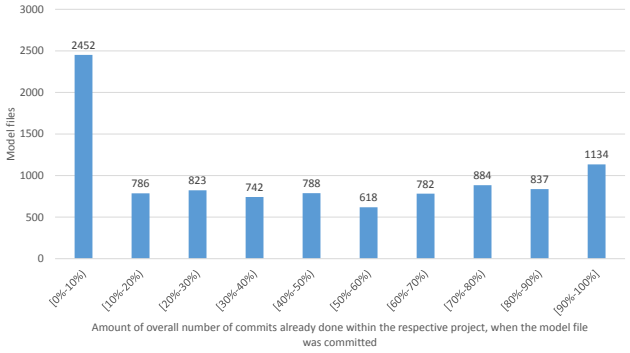


Figure 6: Distribution of files sorted by number of overall commits done when the diagram was introduced (For models in projects that have less than 100 models)

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 7 illustrates there most models are introduce during the last third of the project activities.

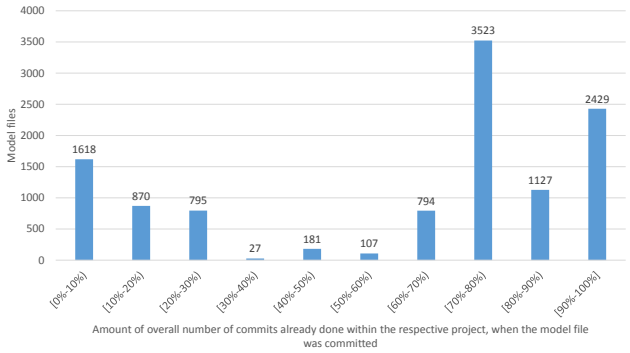


Figure 7: Distribution of files sorted by number of overall commits done when the diagram was introduced (For models of the 21 projects that have 100 or more models)

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

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.

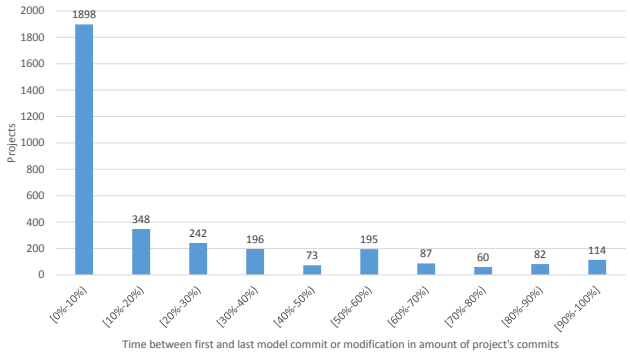


Figure 8: Projects by time between first model commit and last model update-or-commit as percentage of project's commits

Figure 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.

As with RQ3, we use commits as an alternative measure of the time where UML introductions/updates occur. Figure 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 the above finding 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 and often concentrated in the first commits.

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.

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 above.

As a result we identified that 16 576 of the 21 316 found models where unique in our sample. The remaining 4 741 model files represent 2 300 models of which each occurs at least twice. Thus, 21 316 found model files include together 18 876 distinct models. In Figure 10 we summarize how often models with duplicates occurred in our sample. Interestingly, one of the models was found 79 times. In average, models (if duplicated) are duplicated 3.63 times.

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

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.[12] or 19 projects with UML by Ding et al.[4].

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

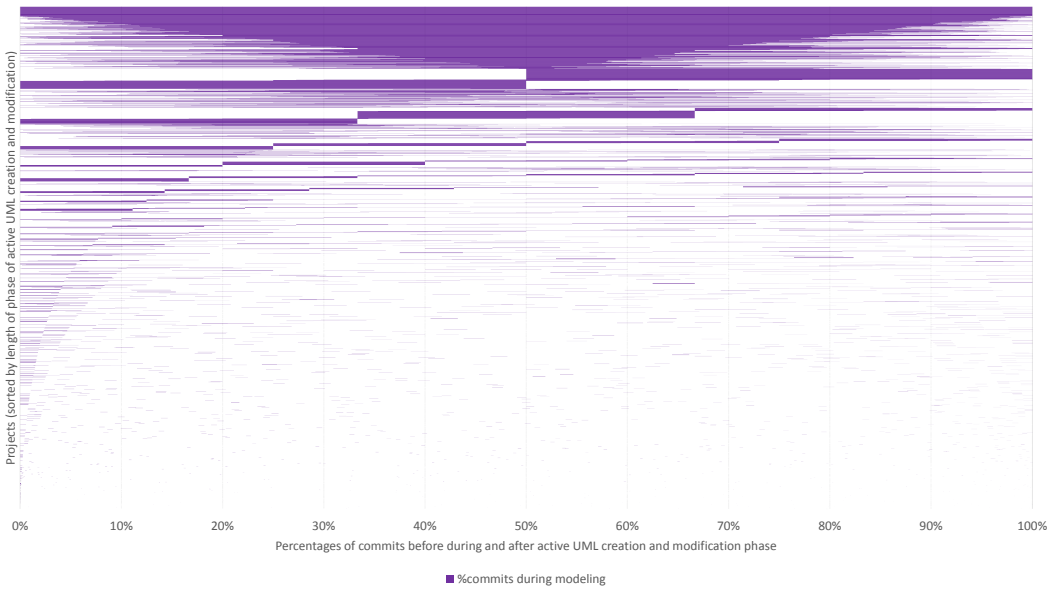


Figure 9: Plot of all 3 295 projects illustrating the placement of active UML creation and manipulation phase within the overall project life span. Time is measured in percentages of commits done, when the first model is introduced and the last model is introduced or updated. The projects are sorted 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).

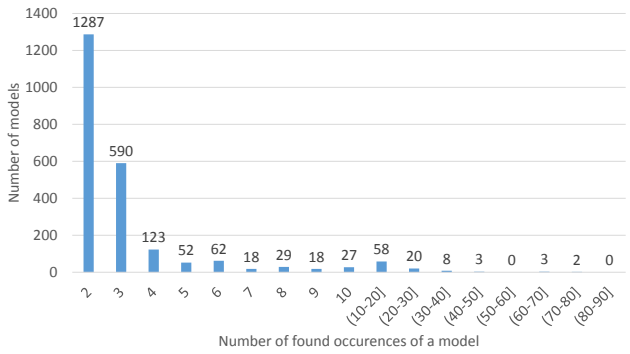


Figure 10: Histogram of models that were found at least twice indicating how often models occur.

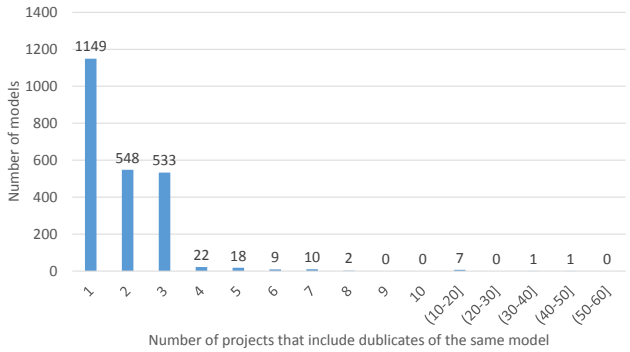


Figure 11: Histogram of models that have duplicates in one or more projects. The histogram shows the number of models by number of projects within which occurrences of a model were identified.

collection. For example, we search for UML candidates using a GHTorrent dump, but accessed the GitHub API to retrieve 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 [23] or our own (Osman et al.'s [15]), 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. [12] 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. [12] 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 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 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.

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 [22]. We identified three types of threats to validity, they are: Construction Validity, External validity and Conclusion Validity.

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 staring 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 [10]. We found that the threat that many active projects might not conduct all their software development in GitHub could somehow mitigate our analysis.

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.

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.

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.

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.

[7]QualitasCorpus http://qualitascorpus.com/

9. REFERENCES

[1] O. Badreddin, T. C. Lethbridge, and M. Elassar. Modeling practices in open source software. In *Open Source Software: Quality Verification*, pages 127–139. Springer, 2013.

[2] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

[3] E. Chung, C. Jensen, K. Yatani, V. Kuechler, and K. N. Truong. Sketching and drawing in the design of open source software. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pages 195–202. IEEE, 2010.

[4] W. Ding, P. Liang, A. Tang, H. Van Vliet, and M. 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.

[5] R. France, J. Bieman, and B. H. Cheng. Repository for model driven development (remodd). In *Models in Software Engineering*, pages 311–317. Springer, 2007.

[6] G. Gousios and D. Spinellis. Ghtorrent: Github's data from a firehose. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 12–21. IEEE, 2012.

[7] R. Hebig, T. Ho Quang, G. Robles, and M. R. Chaudron. List of identified projects with uml and replication package. http://oss.models-db.com.

[8] T. Ho-Quang, M. R. V. Chaudron, I. Samúelsson, J. Hjaltason, B. Karasneh, and H. Osman. Automatic classification of uml class diagrams from images. In *Proceedings of the 2014 21st Asia-Pacific Software Engineering Conference - Volume 01*, APSEC '14, pages 399–406, Washington, DC, USA, 2014. IEEE Computer Society.

[9] 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*, pages 45–55, 2015.

[10] 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*, MSR 2014, pages 92–101, New York, NY, USA, 2014. ACM.

[11] B. Karasneh and M. R. Chaudron. Online img2uml repository: An online repository for uml models. In *EESSMOD@ MoDELS*, pages 61–66, 2013.

[12] P. Langer, T. Mayerhofer, M. Wimmer, and G. Kappel. On the usage of uml: Initial results of analyzing open uml models. In *Modellierung*, volume 19, page 21, 2014.

[13] 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*, pages 42–51. IEEE, 2010.

[14] S. McIntosh, B. Adams, T. H. Nguyen, Y. Kamei, and A. E. Hassan. An empirical study of build maintenance effort. In *Proceedings of the 33rd international conference on software engineering*, pages 141–150. ACM, 2011.

[15] M. 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), Miami, USA, October 1, 2013.*, pages 23–32, 2013.

[16] 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*, pages 149–165. Springer, 2014.

[17] G. Robles, J. M. González-Barahona, D. Izquierdo-Cortazar, and I. Herraiz. Tools for the study of the usual data sources found in libre software projects. *International Journal of Open Source Software and Processes*, 1(1):24–45, 2009.

[18] 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*, 79(9):1233–1248, 2006.

[19] 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*, pages 36–40, 2014.

[20] 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*, 86(8):2110–2126, 2013.

[21] 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*, 19(4):955–1008, 2014.

[22] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[23] K. Yatani, E. Chung, C. Jensen, and K. 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.