Characterization of Software Design and Collaborative Modeling in Open Source Projects

Khandoker Rahad[a], Omar Badreddin[b] and Sayed Mohsin Reza[c]
Computer Science, University of Texas at El Paso, El Paso, Texas, U.S.A.

Keywords: UML, UML Usages, Software Design, Software Quality, Software Maintenance, Mining Software Repositories.

Abstract: Software design is fundamental to developing high-quality, sustainable, maintainable software. Design languages, such as UML, have become the defacto standard in software design, but their infiltration in the mainstream practices remains vague. Recent studies suggest significant and increasing uptake in mainstream and open source spheres. Mining repositories and the software modeling artifacts often underpin the findings of these studies and focus on counting the instances of modeling artifacts as an indicator for adoption. This study aims to characterize this uptake in greater depth by focusing on analyzing the instances of models in open source projects. The goal is to uncover the profiles of developers who tend to create modeling artifacts, and those developers who maintain them throughout the project life cycle and to uncover the timelines of model creation and manipulation in reference to project evolution. This study sheds light on the nature of model-based collaboration and interactions and characterizes the role of model-based artifacts well beyond mining their presence in open source repositories. The study finds that, despite the nominal increase in the presence of model-based artifacts, these artifacts are rarely maintained and are typically created by a small and unique set of practitioners. Models are often created early in the project life cycle and do not play any significant role in the collaborative development activities of the subject projects. Life span of these model files is relatively shorter than the code file life span. Unexpectedly, models tend to be more frequently updated and maintained when the project has a relatively fewer number of models.

1 INTRODUCTION

Software systems and software-intensive systems continue their exponential growth in size and complexity. Such systems require enhanced levels of collaborations and collective comprehensibility and designs (Podgorelec and Heričko, 2007). Continuous software design and modeling throughout the project life cycle are proven to enhance collaboration, maintainability, and the sustainability of these systems.

Recent research that investigates the practices of software design and the adoption of several design languages tend to suggest a consistent increase in the adoption of design and modeling practices. These studies tend to rely on two primary sources of data; mining open source repositories and empirical investigations of the practitioners. Despite the advances in software repository mining tools and techniques, these studies tend to rely on nominal numbers of modeling artifacts presence as an indicator for the level of adoption of design practices. The focus on counting the number of modeling artifacts in repositories leaves much more uncovered, such as the profiles of those engineers who create the models, when are those models created, and how often are these models are maintained.

The goal of this study is to focus on relatively smaller instances of repositories, and investigate their models in greater depth. The investigations focus on uncovering the specifics of those models, their creators and maintainers, their lifespans in reference to the project overall evolution, and the frequency with which these models are updated in reference to other repository artifacts.

In this study, we identify seventeen (17) software repositories selected from an extensive and customized query from GitHub. The query identifies most active repositories with significant presence of design artifacts. From these repositories, we extract every software design artifact and collect extensive contextual data sets that include, the profile of engi-

[a] https://orcid.org/0000-0001-7481-8954
[b] https://orcid.org/0000-0002-8851-4131
[c] https://orcid.org/0000-0003-3379-6319

Rahad, K., Badreddin, O. and Reza, S.
Characterization of Software Design and Collaborative Modeling in Open Source Projects.
DOI: 10.5220/0010266802540261
In *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2021)*, pages 254-261
ISBN: 978-989-758-487-9

neers who created and maintained those models, the timelines of model creations and maintenance, the frequency of their updates in reference to other repository artifacts, among other.

2 RELATED WORK

Studies on software modeling uses are conducted by surveys and questionnaires (Torchiano et al., 2013). However, very few studies are focused on the usages of models in open source (Ho-Quang et al., 2017).

Software modeling practices are investigated by a few industries. Gorschek et al. focused on a sample population, who are programmers, partially working in the industry, and open source systems (Gorschek et al., 2014). The results show that their sample design models are not used extensively. However, UML models are used mainly for communication purposes. Kobryn et al. reported that UML has been widely accepted throughout the software industry (Kobryn, 2002).

Often, research would identify a specific project or organization for in-depth analysis of the practices. These studies often focus on one or a few cases for investigations. There are some works addressing small numbers of case studies of modeling in open source projects. Yatani et al. (Yatani et al., 2009) reported on the model usage in Ubuntu development by interviewing 9 contributors. The authors found that models are not actively used and updated.

In open source software development, code remains the key development artifacts (Badreddin et al., 2013). Researchers know very little about the use of UML models in open source. However, some software engineering researchers performed some efforts to identify UML models and their usages. Nonetheless, the data set is very small and not up to date. ReMoDD (France et al., 2006) is a repository for UML models, but their repository collection is not extensive. Hebig et al. (Hebig et al., 2016) reported on a study that investigates how UML models are used. The results of the study suggest that UML models are generally used at the beginning of the project start.

3 STUDY DESIGN

In this study, we investigate 17 repositories listed in Table 1 that uses and practices UML modeling in open source. The goal of the study is to understand the uses of UML modeling in open source. We used a tool ModelMine (Reza et al., 2020) to identify UML model and their repositories. There are some repository selection criteria that we used to identify UML models such as programming languages, number of commits, number of contributors, and popularity in GitHub. After Identifying models from these selected repositories we analyzed the UML models. We identified the total commits in each repository. Commits are analyzed to understand the updates that are made on the UML files. The raw data is published in (Khandoker Rahad, 2020). This study answers the following research questions.

RQ1. How Prevalent Are Model-based Artifacts in Open Source Projects?
To answer this research question, we investigate the UML based repository in open source. We use the ModelMine tool that has a feature to retrieve repository from open source using different file extension and number of commits and the project update date and other criteria. This research question investigates how often UML models are used in open source repositories.

RQ2. How Are UML Models Maintained in Open Source Projects?
To answer this research question, we investigate UML model commits. Model commits are provided with updates that are performed by the contributors. This information also provides information on the model updates in relation to the project update. This will give us information on how frequently models are used and get updated over time.

RQ3.What Are the Profiles of Those Engineers Who Create, Share, and Maintain Model Artifacts?
To answer this research question, we obtained the repository contributor's information. We also collect profile information including the practitioners' activities, contributions, and experiences in open source projects. We identify the total number of model contributors in each repository.

RQ4. What Is the Lifespan of Model-based Artifacts in Relation to the Project Lifecycle?
To investigate how UML models are maintained by the repository contributors, we mine model and repository commits. We compare the total number of model commits with the total number of repositories commits.

3.1 Data Collection Methodology

This section presents the data collection of this study. We use a mining tool to collect UML based repositories from open source platforms. There are several open source platforms that contain UML based repository. GitHub is one of the popular open sources

Table 1: Repository Information.

Repo #	Name	Contributors	Commits	Language	GitHub URL
1	contrib	7	209	Java	https://github.com/europeana/contrib
2	SnakeInTheDark	6	416	Java	https://github.com/MrStrings/SnakeInTheDark
3	nanoengineer	6	11688	Python	https://github.com/kanzure/nanoengineer
4	luciddb	7	5107	Java	https://github.com/LucidDB/luciddb
5	MobSens	7	776	Java	https://github.com/Institute-Web-Science-and-Technologies/MobSens
6	lambda-alligatoren	6	1786	Java	https://github.com/vincent23/lambda-alligatoren
7	mars_city	21	1304	C++	https://github.com/mars-planet/mars_city
8	arcemu	11	4307	C++	https://github.com/arcemu/arcemu
9	pse_allocation	6	1418	Java	https://github.com/EmilDohse/pse_allocation
10	h5cpp	12	1317	C++	https://github.com/ess-dmsc/h5cpp
11	netty	11	2247	Java	https://github.com/ChenLuigi/netty
12	hotel-california	6	433	Java	https://github.com/verath/hotel-california
13	dataDictionary	8	1316	C++	https://github.com/openETCS/dataDictionary
14	TDA593-18	7	312	Java	https://github.com/Jaxing/TDA593-18
15	model-driven-software-development	7	478	Java	https://github.com/Centril/model-driven-software-development
16	FinalPortalURL	8	254	Java	https://github.com/ahuitz/FinalPortalURL
17	moodles	9	222	Java	https://github.com/hedmanw/moodles

repositories. However, there are deficiencies identifying UML models in GitHub because GitHub mining is a nontrivial task. Nonetheless, there are few publicly available repositories that list UML projects. However, this collection is relatively small and not up to date. In this study, we use a model mining tool named ModelMine (Reza et al., 2020) to retrieve the up to date UML based repositories with recent metadata information. The following sections discuss the mining repositories, identifying UML based repositories and data extraction of UML files.

3.2 Mining Open Source Repositories

In this section, we present the UML based repository collection methodology from open source GitHub. To access the resources from GitHub, we use ModelMine tool (Reza et al., 2020) which fetches models with filtered criteria. To identify model based repositories that use UML, we identify 8 filtered variables that identify well maintained and popular model based repositories listed in Table 2.

There are two types of artifacts (keywords and qualifiers). This is because of the tool requirements for search query construction. It requires at least one search term as keywords to be included in addition to that add qualifiers. This artifacts (keywords or qualifiers) allow us to limit our search to specific areas of open source repositories. For example, minimum 200KB memory size is used to identify non-trivial projects in Table 2.

We use a .uml file extension in the search and retrieve 465 UML based repositories. These repositories include a minimum of one UML model in the repository. This is a large number of repositories to analyze. We use a set of repository selection criteria shown in fig 1 to make the list short. The criteria are as follows: minimum 200 commits, 5 contributors, and primary programming languages of the repository must belong to Python, Java or C++. Finally, we remove duplicate repositories to identify 17 repositories that has a minimum of one UML modeling. This exclusion criteria also ensure that we do not include trivial model based repositories.

3.3 Identification of Potential UML based Repositories

After the mining process, we identify 17 UML based repositories. Primarily, the .uml file extension is used because they are defacto standard (Ozkaya and Erata, 2020). We select .uml file extension to retrieve UML model files with their repository information. Moreover, all UML files are not actual UML files. Therefore, we use a manual filtering process to identify actual UML models to ensure expert opinion on the selection.

This filtering process has been applied after collecting all the UML models. Identifying actual UML models will be impossible if we do not classify UML models at the beginning of repository selection.

3.4 UML Metadata Extraction

After identifying 17 repositories that used UML modeling, we found a total number of 886 UML files. After filtering out the duplicate model files we collected 92 unique model files. To understand the usages of these model files we analyze the commits of these model files. ModelMine tool is used to retrieve the commits of those model files. The tool is able to provide the commits, author information, file creation date, modification date, etc.

3.5 UML Model File Assessment Criteria

We conduct analysis on UML model files. We compare code file commits with the average number of UML model file commits in respective repositories.

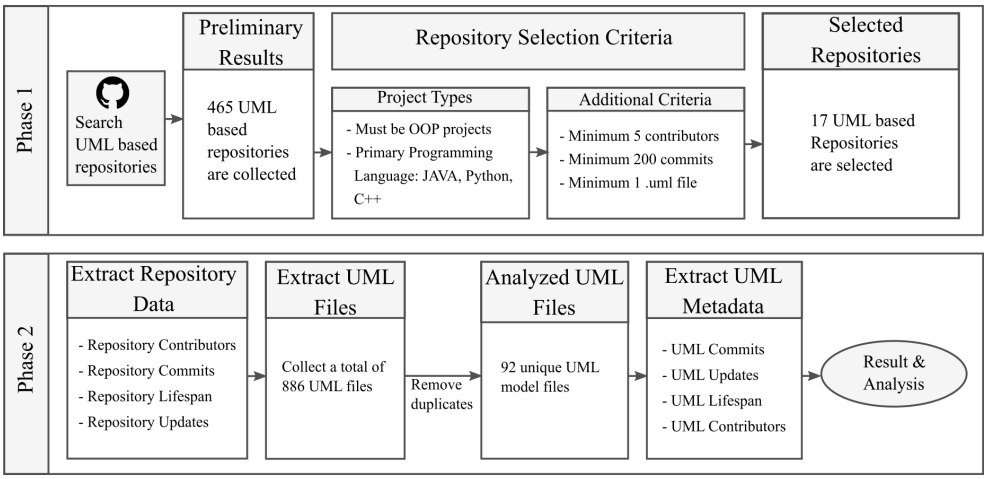

Figure 1: Data Collection Flow.

Table 2: Variable Identifying Model Based Repository.

Number	Type	Variable Name	Description
1	Keywords (Required) UML	Context	Identify repositories that matches given context.
2	Qualifiers (Optional)	Language	Identify repositories with given language such as java.
3		Extension	Identify repositories with given file extension.
4		Memory size	Identify repository which have minimum 200 KB
5		Life Span	Identify repositories that have minimum project life span
6		Popularity	Identify repositories that have minimum number of stars as popularity index.
7	Commits	Number of Commits	Identify the total number of commits in a repository
8	Contributors	Number of contributors	Identify the total number of active contributors in a repository

Commits percentage and average commits are used to visualize the comparison. Similarly, the life span of model file is computed. Total number of model file and model file updates is counted for all seventeen repositories. Additionally, we assumed software development is performed in three development phases (First phase, Second phase, Third phase). This assumption is made to understand when models are created and whether they are maintained after creation. These three phases are identified by dividing the entire project life into three parts equally.

4 RESULTS

This section presents the results of this study such as how often models are used in open source, who create, contribute, and manage those models. Moreover, we presented the total number of model file contributors compared to code file contributors. This section presents how those models are updated by the contributors and whether maintained in a timely manner. Finally, we presented the timeline of model file creation in reference to the project timeline and time span of model updates in the software life cycle.

4.1 How Models Are Used in Open Source

For the analysis, we downloaded 465 non-forked GitHub repositories with ModelMine tool (Reza et al., 2020). After filtering out the data for potential UML files, we retrieved a list of 17 repositories and their GitHub link. 886 files were classified as UML model files in 17 repositories, this includes duplicate models. To filter out unique models files, we performed the extraction of model related data which are unique. Unique model files are identified after the data collection. This filter resulted in 92 unique UML files.

We obtained the dataset from GitHub between Jan 2012 to August/September 2019. All the commits and repository information lies between this time frame. The reason for selecting this time span is due to the fact that our retrieval procedure takes so much time that context changes. Therefore, for instance, in the time that goes from the retrieval of information of the files that are included in a project (August/ 2019) to the time where the git repositories were downloaded (November/December 2019), some of them were renamed, deleted or made private.

In consequence, 92 unique model files are used for analysis. These files belong to 17 GitHub projects. Of these 3, include a single UML file, only and 14

projects include between 2 and 9 UML files. We perform our analysis in 92 unique model files and their repositories.

4.2 How Models Are Updated and Maintained

This research question answers whether models get updated and if so, how often models get updated. We computed the model commit in reference to project commits in fig 2. For example, in repository 17, 8.5% model commits is performed compared to repository commits.

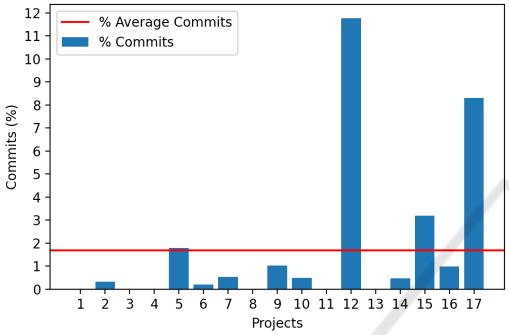

Figure 2: UML Model Commits.

Additionally, we computed the total number of model file updates for those repositories that have equal or less than 10 model files and those have more than 10 model file in fig 3. In fig 3, the total number of model file updates are grouped in eight categories where X-axis and Y-axis presents the total number of model file. Our results indicate that many model files never get updated (fig. 3). The results show that projects with less than 10 model files are frequently updated than projects containing greater than 10 model files. This result is reflected in Fig 3.

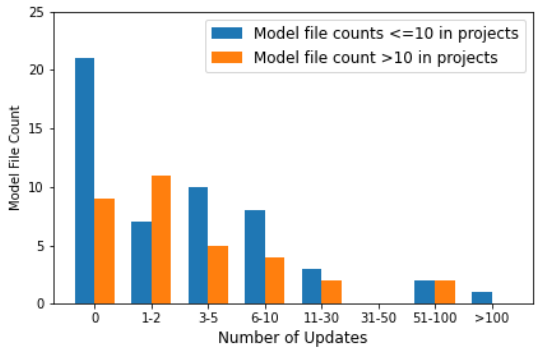

Figure 3: Model File Distribution Per Update.

Fig 2 depicts all that the repositories have lower number of average model commits than the projects file commits. This explores the phenomenon that models are less updated and maintained than code files. The highest percentage of model commits performed in repository number 12 which is approximately 12%. However, repository 1,3,4,8,11 have approximately 0% of model commits compare to repository commit.

4.3 Who Creates, Contributes, and Manages Models

This research question investigates model contributor's profiling information and the total number of model contributor in relation to repository contributor. Table 3 presents model contributors profiling data in GitHub (Only publicly available data have been collected). This reveals the model contributor's software development experience in GitHub. This also includes the total number of contributed repositories in GitHub and the total number of contributions in GitHub. Table 3 data shows that a large number of model contributors are experienced and have a minimum number of contributions in GitHub except for model contributor number 5,6,7 in repository LucidDb (Repo # 4) which is shown in bold. These three model contributors have less than one year of software development experience and additionally, their total number of contributions in GitHub is one.

Results also show that very few contributors contribute to the models in fig 4. Fig 4 depicts that the number of model contributors is always lower than repository contributors. We measured the average number of contributors and found that the average number of model contributors is lower than the repository contributors. For instance, the average number of repository contributor is 9 whereas the average number of model contributors is 4. The average contributor number is measured by simply adding all the contributors and divided by the total project count.

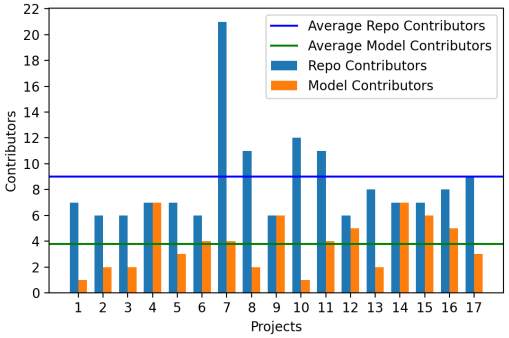

Figure 4: Number of Model and Repository Contributors.

Table 3: Model Contributor's Profiling Information.

Repo #	Repo Name	Model Contributor	Number of Contributed Repos	Years of Experience in Github	Total Number of Contributions
1	Contrib	#1	12	10	1352
2	ShakeInTheDark	#1	9	6	533
		#2	4	6	75
3	nanoengineer	#1	1	10	2
		#2	11	10	169
4	LucidDb	#1	17	9	2127
		#2	15	12	1627
		#3	1	8	2
		#4	37	11	1051
		#5	**1**	**0**	**1**
		#6	**1**	**0**	**1**
		#7	**1**	**0**	**1**
5	Mobens	#1	6	7	125
		#2	28	7	1557
6	lambda-alligatoren	#1	19	9	692
		#2	20	8	1434
7	mars_city	#1	26	8	583
		#2	68	6	1075
8	arcemu	#1	20	11	4498
		#2	6	12	1254
9	pse_allocation	#1	14	7	424
		#2	4	4	209
		#3	2	4	233
10	h5cpp	#1	5	8	4218
		#2	13	7	5605
11	netty	#1	39	15	11031
12	hotel-california	#1	10	7	868
		#2	41	11	3118
		#3	8	8	370
		#4	6	7	238
		#5	7	8	365
13	dataDictionary	#1	4	8	496
14	TDA593-18	#1	23	6	648
		#2	30	6	1605
		#3	37	6	2431
		#4	4	6	309
15	model-driven-software-development	#1	7	7	1091
		#2	4	9	562
		#3	13	7	5182
		#4	2	7	88
		#5	2	7	289
		#6	119	9	13886
16	FinalPortalURL	#1	15	5	394
		#2	8	6	171
		#3	6	5	842
17	moodles	#1	12	7	688
		#2	32	7	2469
		#3	35	7	4564

4.4 When Models Are Created and Updated in Software Life Cycle

The last research question investigates model life cycle in relation to the project life cycle. Fig 5 shows that 66.3% of models are created and get updated in the first phase of software development. Only, 24.4% model files are created and get updated in the second phase of the development and 9.3% in the third phase. This reveals an interesting finding that models are introduced at the beginning of the development whereas at the end very few models are introduced.

Fig 6 shows the life span of the model file in relation to the project life span. We measured the life span (model and project) by simply identifying the first creation date and last update date. Results in fig 6 reveal that the average model life span compares to the project life span is only approximately 15%.

5 ANALYSIS

Considering our initial expectation we found a small number of model files in open source. 17 selected repositories contained 886 models. After the filtering process, we identified 92 unique model files which is a small number of models compared to code files. In the following, some interpretations are presented from this study.

5.1 Models Maintenance

The interpretation is that models are introduced very early in the development and rarely maintained by the developers. Several studies support this claim such as Hebig et al. found that creating/updating of UML happens most often during a very short phase at the project start (Hebig et al., 2016). Moreover, Ho-Quang et al. reported that modeled designs are only partially followed during implementation (Ho-Quang

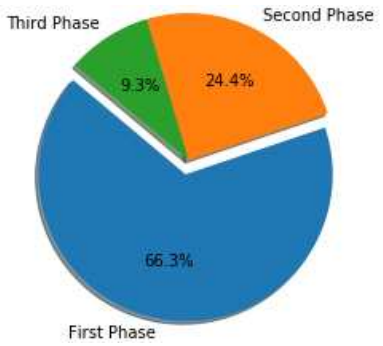

Figure 5: Distribution of model files when first introduced in project life cycle.

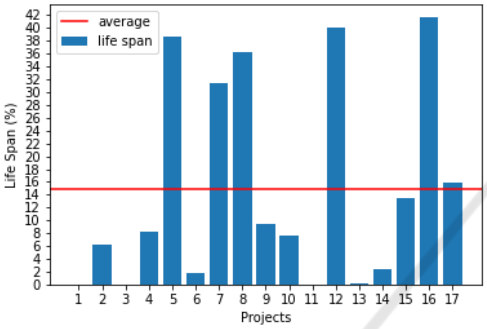

Figure 6: Life Span of UML Models.

et al., 2017). Models are just used in designing or visualizing the system, which could be one of the reasons behind this. Another interpretation is that maintaining models is an extra work that developers are not willing to perform because of time constraint.

Fig 3 shows different model file distributions per edit. This explores that projects with fewer model files are updated frequently than projects with a higher number of model files. The reason behind this could be there is no systematic way to maintain model files such as if any changes needed to make model files require to regenerate the model file.

However, introducing models is a positive aspects on the usages of modeling in open source. This could be a starting point of the use of models in open source.

5.2 Model Contributors

Results show that model contributors are a subset of repository contributors. It is also evident that very few contributors take part in modeling compare to code file contribution. The interpretation behind this is contributors who are experienced with modeling are able to contribute to model creation and updates. Table 3 data verifies this claim. However, Gorschek et al. found a different result that modeling is mostly done by the novice programmers rather than experienced software engineer (Gorschek et al., 2014).

5.3 Model Life Cycle

Model life cycle refers to the time between model creation and end. In the analysis, it is found that models are created a very early stage of developments and get maintained after that for a short period of time (Fig. 5). However, at the end of development, there is very little updates evident in the model files. The interpretation is that models are only used for system designing purpose and become obsolete over time. Other possibilities, such as the fact that the models could also be used for code generation, and the code simply do not need to be further adapted. Supporting this claim Osman et al. reported that the frequency of updating UML models is low (Osman and Chaudron, 2013). Authors found two triggers for updating UML diagrams: 1) if there are changes in the features of the system, and 2) if there is a group of newcomers joining the project.

5.4 Reasons of Model Usages

One of the threats that are remained in this study is that we do not know the source of the actual usages of these UML modeling. Additionally, we do not have sufficient evidence on whether these models are introduced because of designing purpose or introduced to contribute to the actual development. We plan to further investigate these phenomena in our future research.

6 THREATS TO VALIDITY

There are internal and external threats to validity in this study. Internal threats refer to risks within the study whereas external threats refer threats outside of the study design.

External Threats. We do not claim that the set of repositories presented in this paper include all UML based repository in open source. Though this study is worth to investigate because the results provide insightful information on UML uses in open source.

There is another bias in this study that is whether identified UML models are actual UML models. To mitigate this risk we manually checked the identified UML files that really contain any UML modeling.

The study is conducted within the context of GitHub, but not other open source platforms such as BitBucket, SourceForge. Therefore, the results of this study can not be generalized to the other platforms. It is possible that UML is used in a different ration within projects at other platforms. However, GitHub

is one of the popular and most used open source platform (Noten et al., 2017), hence the contribution of this research is useful for fellow researchers in this field.

Internal Threats. Internal risks of this study is that considering UML files as the only UML model based repository. We used UML files since they are defacto standard (Ozkaya and Erata, 2020). There are many other file formats are available that also present UML modeling.

Risks exist in profiling information for the software engineers who participate in software modeling. It could be the case that the targeted software engineers do not perform modeling frequently and this is the reason, we find fewer updates on models.

Another risk is that the identified UML models are created only for teaching or academic purpose for a short period time. And these models are not representative for the models in open source projects. To mitigate this risk we manually checked the repositories and confirmed that the selected repositories are non trivial.

7 CONCLUSION

In this paper, we investigate a sub set of projects that has UML modeling and the usages of model files in the open source. Model file commits provide useful information on the updates and maintenance of the UML models. Additionally, this study investigate how often contributors update model files. Study results show that the models are created and maintained by experienced software engineers.

Further, this study reveals that often UML model files are created at the very first phase of software development. Modification or updates are made on the files mostly at the beginning and in the middle of software development phase. Study results show that the model life span is shorter compared to project life span. Often, repositories with fewer model files get updated frequently than repositories with higher number of model files.

This study investigate 17 repositories which are a small data set representing open source projects. In the future, we have a plan to include more repositories in our data set which will be a broad representation of models usages in the open source.

REFERENCES

Badreddin, O., Lethbridge, T. C., and Elassar, M. (2013). Modeling practices in open source software. In *IFIP International Conference on Open Source Systems*, pages 127–139. Springer.

France, R., Bieman, J., and Cheng, B. H. (2006). Repository for model driven development (remodd). In *International Conference on Model Driven Engineering Languages and Systems*, pages 311–317. Springer.

Gorschek, T., Tempero, E., and Angelis, L. (2014). On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, 95:176–193.

Hebig, R., Quang, T. H., Chaudron, M. R., Robles, G., and Fernandez, M. A. (2016). The quest for open source projects that use uml: mining github. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pages 173–183.

Ho-Quang, T., Hebig, R., Robles, G., Chaudron, M. R., and Fernandez, M. A. (2017). Practices and perceptions of uml use in open source projects. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 203–212. IEEE.

Khandoker Rahad (2020). Raw data: Characterization of software design and collaborative modeling in open source projects,.

Kobryn, C. (2002). Will uml 2.0 be agile or awkward? *Communications of the ACM*, 45(1):107–110.

Noten, J., Mengerink, J. G., and Serebrenik, A. (2017). A data set of ocl expressions on github. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 531–534. IEEE.

Osman, M. H. and Chaudron, M. R. (2013). Uml usage in open source software development: A field study. In *EESSMOD@ MoDELS*, pages 23–32.

Ozkaya, M. and Erata, F. (2020). A survey on the practical use of uml for different software architecture viewpoints. *Information and Software Technology*, 121:106275.

Podgorelec, V. and Heričko, M. (2007). Estimating software complexity from uml models. *ACM SIGSOFT Software Engineering Notes*, 32(2):1–5.

Reza, S. M., Badreddin, O., and Rahad, K. (2020). Modelmine: A tool to facilitate mining models from open source repositories. In *2020 ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM.

Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., and Reggio, G. (2013). 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.

Yatani, K., Chung, E., Jensen, C., and Truong, K. N. (2009). 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.