CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

UML-Ninja: Automatic Assessment of Documentation and UML Practices in Open Source Projects

Bachelor of Science Thesis in Software Engineering and Management

Aras Bazyan
Nimish Krashak

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

ARAS BAZYAN
NIMISH KRASHAK

Supervisor: MICHEL CHAURDRON, TRUONG HO-QUANG
Examiner: ROGARDT HELDAL

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden May 2018

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

UML Ninja: Automatic Assessment of Documentation and UML Practices in Open Source Projects

Aras Bazyan
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
aras.bazyan@gmail.com

Nimish Krashak
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
nimish.krashak@gmail.com

***Abstract*—Context: Assessment of software documentation practices in open source projects is important because, besides the source code, information regarding the open source project is in the documentation. Quality of documentation can help in determining the quality of the project. Goal: The goal of this paper is to automate the process of assessing the quality of documentation and UML in open source projects. Method: We conduct a design science research study and evaluate the outcome based on 14 interviews with researchers and practitioners. Result: The result of this paper is UML-Ninja, which is a web tool to automatically assesses quality of documentation in open source projects based on indicators. Conclusion: Both researchers and practitioners found the approach of UML-Ninja tool good and appropriate.**

***Index Terms*—Documentation, UML, Automation, OSS**

I. Introduction

Software architecture (SA) documentation provides a blueprint of a system and represents the structure of a system [1]. SA document contains models and design decisions made during the architecture design process and it is an essential part of a project as it provides access to the necessary information to stakeholders [2], [3]. Besides source code, SA documentation is one of the fundamental sources of information regarding the project. The models in SA are mostly presented using the Unified Modelling Language (UML) [25]. UML is widely used when it comes to modelling. It has become the accepted standard for modelling in system and software development [4]. A field study done by Regina et al. proclaims that from the 1240000 projects on GitHub that were mined in their research to count the number of projects that contained UML documents, around 1% of the projects were deemed to contain UML documents. This means around 5000 projects were found with UML documents and therefore depict high numbers of UML projects. This research only took GitHub projects into consideration whereas there are organisations that use UML in their projects without hosting them on GitHub.

There are a few studies on the use of UML in Open Source Projects (OSS) [6], [8]. In OSS, SA descriptions and models are used for communication purposes as well as understanding the project. When it comes to the assessment of documentation or UML in OSS projects, the approach taken in the available literature is field study or case study [9], [11]. Good quality of documentation is important in software development, in fact, software quality improves when the quality of documentation improves [7].

In order to maintain good quality of software documentation, having an easy access to an overview of the current state of software documentation quality will help in identifying areas of improvement. There is lack of tools that automatically assess quality of documentation in OSS. This paper introduces UML-Ninja[1] a web tool that automatically assesses UML and documentation practices in OSS and presents the data in a dashboard. This is done by collecting and analysing data regarding a project and showing the data in a well-structured and easy to use dashboard view. The dashboard view is the main component of UML-Ninja, it offers a quick way to see an overview of the project quality in terms of documentation, including UML models. The quality check is on the model level and not code level, because the focus of UML-Ninja is assessment of non-source code content, the documentation specifically.

The goal of the tool is to motivate users to improve the quality of their documentation. The motivation for developing UML-Ninja is the hope that it enables many use cases for many stakeholders in the field of software engineering. For example:

- Developers: As a developer of a project you would like to see the current status of the project‘s documentation quality and be able to recognise and rectify any issues with documentation.
- Student: As a student you would like to check the quality of your documentation and UML in your course projects so that you can improve it.
- Researcher: As a researcher you would like to collect data for further research and analysis or collect data regarding quality of documentation in software projects for empirical studies. Using this solution, the researcher

[1] http://204.48.31.89

would be able to get the required data in an automatic and structured manner. A researcher could also be wanting to compare one project with another in terms of documentation.

The rest of this paper is structured as follows. In section II, we present the research questions of this paper. Section III explains and reviews background and related works. Section IV describes the research methodology of this paper. In section V we introduce and describe UML-Ninja in detail. Section VI evaluates UML-Ninja and describes validity threats

to this study. Section VII discusses the findings and relate them to the related works. Section VII highlights the conclusions of this paper and presents possible future works.

II. Research Questions

This section presents the research questions (RQ) this paper is answering. The research questions consist of one main question followed by two sub-questions (SQ). The two sub-questions complement the main research question. The research questions are formulated as follows:

RQ1: How can we automatically assess the quality of UML and documentation practices in open source projects?

RQ1 aims to find a new approach and it is formulated in the following two sub-questions:

- SQ1,1: How can the assessment of documentation be automated?
- SQ1.2: How can the assessment of UML documentation be automated?

The scope of SQ1,1 refers to documentation process and documentation content. Documentation process includes e.g. frequency of document updates. Documentation content includes e.g., the layout of the document.

The scope of SQ1.2 refers to UML process and UML content. UML process includes e.g., contribution ratio to see how many people actively contribute to UML. UML content includes e.g., number of dependencies vs. number of classes in a class diagram.

III. Background and Related Work

This section discusses background and related work about the assessment of documentation and automatic assessment of software quality.

A. Background Work

This research is an extension of our supervisor, professor Chaudron‘s research regarding use of UML in the open source industry and its benefits of use [16]. The overall vision of his research is aimed towards improving quality of documentation in software projects which is applicable to this research as well. In the research [16], they extracted 93,648 UML files from 24,797 GitHub projects. They conducted a survey with contributors of 458 GitHub projects in order to identify the use of UML in OSS projects. It was found that UML is generally perceived as supportive to new project contributors, but it has not impact on attracting new contributors [16]. Considering the findings from that research, our research uses the findings as a foundation upon which UML-Ninja is created to motivate the users of UML-Ninja to achieve the overall vision of improving documentation quality.

B. Related Works

Documentation plays an important role in software development to describe the product at all levels of development [31]. Moreover, UML documentation of software projects helps representing the software system on an abstract level [14] which helps software developers in understanding the software system [15].

In terms of analysing quality of documentation, several researches have been done that have produced a list of metrics that could accurately depict quality of documentation [9], [32]. The research done by Aversano et al. [9] describes an approach to identify metrics related to software documentation that contributes towards improving software quality. In contrast to our study, Aversano‘s research only produces a list of indicators for documentation without implementing them in a tool for demonstration purposes. Moreover, his research does not detail UML documentation indicators that are an essential part of documentation. Thus, the analysis of documentation by using only documentation indicators stated by Aversano‘s research, would be an incomplete approach to assess quality of documentation.

Another research by Carvalo et al. [32] focuses purely on software documentation. This research produced various indicators about the software documentation but on the other hand does not emphasize UML indicators in contrast to our research. Indicators for UML are a key focus of our research.

To analyse UML models in OSS, extraction and classification of those models is a required step. To achieve this, a field study was done by Chaudron et. al [11], which was focused on the usage of UML in open source projects. The authors demonstrated their approach through manual identification of commonly used UML models in software projects that are present in the documentation of open source software. The main findings included the type of UML models used, relation between size of design and size of implementation along with timing of code changes related to document changes. Our study is different from this study as we are taking a further step to assess the documentation in the project and not solely identify key patterns, and relations found between change in implementation to change in documentation/UML documentation.

Regarding analysis UML models, Tsiolaki [12] produced metrics for analysis of UML Class Diagrams based on the model elements such as class, operations, associations etc. In comparison to our research, this research does not focus on automation of analysis and mainly uses mathematical representations through graph transformation to calculate the metrics whereas our research focuses on automating the process of the assessments.

A research has also been done by Ericsson et al. [13] that focuses on assessing technical documentation of software projects. Metrics such as clone detection, usage profile, structure, and metric analysis were used to analyse the documentation. The study mainly focussed on composing and identifying various metrics used to assess documentation quality.

Regarding automatic assessment of UML models, a related work is SDMetrics [17]. SDMetrics is an object-oriented design measurement tool that is used to measure structural properties of UML models. It is capable of measuring 34 types of class diagram metrics [17]. SDMetrics is only focused on content of UML models. Our study is different from the previous two related works as our study focuses on both documentation and UML aspects of a software project in order to provide a complete picture of the quality of the project's documentation. Moreover, our research aims at automating the analysis in order to conserve time required for analysis of documentation and interpretation of results.

Moreno-Len and Robles [30], described an approach to automate evaluation of Scratch[2] projects. Scratch is an introductory and visual programming language aimed to help children learn to code. Moreno-Len and Robles's implementation of automated analysis provided the owner/developer of Scratch projects a quick overview of their Scratch project using automatic source code analysis, the results of which were grouped into indicators that covered the key aspects of a Scratch project. This related work is most relevant to our field of study in terms of automation of assessment of a software project. In contrast to our project, their focus is only on assessing Scratch projects and not documentation or OSS projects.

IV. Research Methodology

The selected research methodology is Design Science, which focuses on creation of innovative artefacts to solve real world problems while providing a high priority towards relevance to the industry which the research is related to [18]. We chose to follow the Design Science Research Process (DSRP) model [10] for the development of UML-Ninja tool. There are six steps in DSRP model:

A. Identify problems and motivate

Problem identification and motivation emphasises on the research problem. In our case, as mentioned in the previous sections, there is lack of tools that automatically assess documentation practices in OSS and facilitates the easy integration of different types of indicators. A solution that automates the process of assessing the quality of documentation can help save time and make the process easier. It also enables the possibility to obtain documentation quality data using a dashboard view that accumulates the required data in an orderly structure.

[2]https://scratch.mit.edu

B. Define Objective of a Solution

Objectives of a solution is about why the solution is needed and what is the aim of the solution for the problem described in step A. In our case, the objective of the solution is to be able to quickly assess documentation practices in OSS that can help in improving documentation quality and to obtain relevant information regarding documentation quality. The solution focuses on assessing documentation in general as well as UML documentation practices, which resulted out of research and discussions carried out with our supervisors. The topic of "indicators" was touched upon during the discussions which were defined as a specific measurable characteristic of the documentation of a software project used to determine the quality of it.

Further discussions with them concluded that documentation can be divided into two parts with sub-parts in order to illustrate the complete picture of documentation:

- Documentation: This part focuses on documentation in general and it is divided into two sub-parts:
  - Documentation Process: Qualitative features regarding documentations such as frequency of documentation updates.
  - Documentation Content: The content of Documentation practices is formed by the information inside the document. Due to not having access to software documentation of each project along with time constraints of this project, the contents of the document were not analysed but indicators that affect its content were researched upon.
- UML: This part focuses on UML models and it is divided into two sub-parts:
  - UML Process: Qualitative features regarding the process of UML documentation in OSS, such as the ratio of UML contributor ratio.
  - UML Content: Content of UML documentation mainly consisted of UML models. A deeper analysis on UML class diagrams are done in the tool using indicators due to them being the most prominent model being used to depict software architecture in open source projects [11].

The indicators for both parts were produced by looking through already done research along with consulting our supervisors who have done multiple researches concerning UML documentation and its use [8], [11], [16].

Table 1 shows the indicators this paper researched about.

It includes details regarding the indicators this paper researched about. Description column describes the indicator in detail. Raw Output contains the range of values that results from calculation of the indicator. Reference column states the source of the indicator. Reason for Implementation Choice explains the reason why the indicator was implemented or not implemented by UML-Ninja.

TABLE I: DETAILED LIST OF INDICATORS USED IN THIS RESEARCH

Indicator ID	Indicator Type	Indicator Name	Description	Raw Output	Reference	Implemented in UML Ninja	Reason for Implementation Choice
1	UML Content	Attributes vs Class	Ratio between total number of attributes and total number of classes in a class diagram	0.0 < 1.0	[20]	Yes	The metrics used to calculate this indicator - Number of classes and number of attributes to a large extent depict the relation between the class diagram and its qualitative features
2	UML Content	Associations vs Class	Ratio between associations and classes in a class diagram	0.0 < 1.0	[20]	Yes	The metrics used to calculate this indicator - Number of classes and number of associations to a large extent depict the relation between the class diagram and its qualitative features
3	UML Content	Methods vs Class	Ratio between number of methods and number of classes in a class diagram	0.0 < 1.0	[20]	Yes	The metrics used to calculate this indicator - Number of classes and number of methods to a large extent depict the relation between the class diagram and its qualitative features
4	UML Content	Average Parameters per Method	Average number of parameters per method	> 0.0	[26],[20]	Yes	This indicator has been empirically analyzed to calculate a threshold value which is viable after verifying this metric with multiple software projects
5	UML Content	Dependencies vs Class	Relation between number of dependencies and number of classes	0.0 < 1.0	[20]	Yes	This indicator bears a positive correlation with the above stated Associations vs Class. Data availability restrictions were also a reason to choose this metric.
6	UML Content	Data Access Metric	Ratio between number of private and protected attirbutes and total number of attributes	0.0 < 1.0	[21]	Yes	The metric has been empirically analyzed to accurately predict cohesiveness of a class
7	UML Content	Correspondence	Similarity of content of UML documentation to source code	%	Supervisor	No	This show both aspects of the abstraction of the model/documentation as well as the 'up-to-date-ness' of the documentation
8	UML Content	Reverse/Forward Engineered	If the model reverse engineered? Or forward engineered?	Boolean	Supervisor	No	Source code analysis was out of scope for this research
9	UML Content	Coupling Between Objects	Count of number of class that are coupled to a particular class	Integer	Supervisor	No	Data not available through SDMetrics. Would have required different analysis technique
10	UML Content	Cohesion	Degree to which elements in a model belong together	%	[29]	No	Data not available through SDMetrics. Would have required different analysis technique
11	UML Process	UML Commit Ratio	Commit ratio between UML commits and all commits.	Ratio	Supervisor	Yes	This shows the amount of UML commits in a open source project and how much they update UML models.
12	UML Process	UML Contributor Ratio	Ratio between number of all contributors to the number of people updating the models.	Ratio	Supervisor	Yes	This shows the amoutn of people who actively work on and update UML models
13	UML Process	Document Evolution	Ratio between UML Document updates and size of the document	Ratio	Supervisor	No	Data not available for the evolution of document
14	Document Process	Document Commit Ratio	Commit ratio between document commits and all commits	Ratio	Supervisor	Yes	This shows the amount of document commits in a open source project and how much they update documentation
15	Document Process	Document Contributor Ratio	Ratio between the number of all project contributor to the number of people updating the documentation	%	Supervisor	Yes	This shows the amount of people who actively work on and update documentation in the project
16	Document Process	Version Controlled	Does the project use versioning (system) (such as GitHub) for documentation	Boolean	Supervisor	Yes	This shows if the document is under a version control system. Project memebrs can review and compare document with other version if the document is under version control.
17	Document Content	Editability	Is the document static (pdf) or editable (.docx, online)	Boolean	Supervisor	No	This is an indicator of project practices towards modeling: en editable format implies that the project considers that models/documents could be updated as the project evolves
18	Document Content	Clone Detection	Clone detection is an analysis done to detect text copies in the technical documentation.	Boolean	[13]	No	Level of content analysis is out of scope for this research
19	Document Content	Structure Analysis	Structure analysis is used to analyze the technical documentation is structured, with respect to chapters, sections, paragraphs, etc.	Layout Structure	[13]	No	Different documents have different structures that they follow which is hard to normalize, especially if they use different documents with different structures.
20	Document Content	Readability	The ease with which a user can read the text written in the documentation	%	Supervisor	No	User input required to assess readability for each documentation which is not feasable for this research. Also, data needs to be obtained regarding the language and grammer in the document.
21	Document Content	Completeness	The extent to which number of software features/functions are documented	%	[9]	No	Out of scope to calculate features/functions of software project along with analysing documentation content

C. *Design and develop*

Design and development step focuses on the development of a solution for the problem at hand. In our case, the aim is to develop a tool for automatic assessment of software documentation quality. The tool facilitates an easy integration of different types of indicators for assessing documentation practices in OSS.

D. *Demonstrate*

Demonstration refers to showing and using the developed artefact. In our case, we demonstrate the tool by testing it with 10 OSS projects. These projects were chosen based on the availability of their data on the two databases that we had access to. The rest of the details are described in the section V, *Introducing UML-Ninja Tool*.

E. *Evaluation*

Evaluation, emphasises on evaluating the artefact with regards to if and how it solves the identified problem. Throughout the project time frame, we had weekly meetings with the supervisors on updates and progress. During meetings, we also discussed and received continuous feedback on the tool.

To evaluate the final version of UML-Ninja, we conducted 14 interviews. The interview questions consisted of two parts. Part 1 consisted of 8 open-ended questions concerning the usefulness of the tool. The participants were asked if and how the tool could help them in accomplishing their tasks in terms of assessing quality of documentation in a project. They were also asked questions regarding limitations of the tool and possible indicators that could be implemented in UML-Ninja. Part 2 of the questions was focused on evaluating the usability of the tool using System Usability Scale (SUS) [19] standard questions. SUS is one the standard and reliable ways to evaluate usability [22], it consists of 10 questions with five response options for respondents. The choices are based on a 5-point scale, ranging from ”Strongly agree to ”Strongly disagree. An example of SUS question form is shown in appendix C. Evaluating usability is important because we want UML-Ninja to be user friendly and easy to use. The interview participants mainly consisted of academic professionals, and researchers at university of Gothenburg. Moreover, committee members of the Mining Software Repository of the ICSE organization[3] were also suggested by our supervisors to participate in the evaluation of UML-Ninja. Student and developer inputs were also taken into consideration for this evaluation due to them being a prospective user of this tool. Students that had been a teaching assistant to one of the university course pertaining to Software Architecture Documentation were chosen to help with the assessment of our tool. Table 2 shows the number and user profiles of the participants.

The prospective interviewee‘s knowledge and contribution to the topic of UML and documentation in general was considered to filter participants that will be relevant to our research.

[3] https://2018.msrconf.org

TABLE II: NUMBER AND USER PROFILE OF INTERVIEWEES

User Profile	*Number of Participants*
Researcher	6
Developer	4
Student	4

Considering the possible use cases of this tool, the interview was aimed at the categories of people as described in section I, *Introduction*. The interviews are used to evaluate UML-Ninja, which is described in section VI, *Evaluation of UML-Ninja Tool*.

Preference for the evaluation was given to an in-person interview with each session being recorded with the interviewee. Due to several constraints, this was not possible for each interviewee therefore a Skype interview was the alternative used for assessment with the entire call also being recorded for analysis purposes. The structure of the evaluation for in-person interviews or Skype calls was divided as such -

- Introduction (3 mins) - A verbal introduction of the research and the tool was given to the interviewee along with statement of anonymity. The interviewee was informed on the procedure and had the right to discontinue the interview at any time they wished to.
- Hands-on Tutorial (5 mins) - A hands on tutorial was given to the interviewee while explaining various functions and elements of the tool.
- Exploration (10 mins) - The interviewee was requested to freely explore the tool and clarify any issues he/she experienced when operating it.
- Interview (10 mins) - The interview session lasted around 10 minutes where first, the interviewee was introduced to the list of prospective users of this tool along with the tasks related to each user. They were requested to answer as one of the prospective users. The interviewee was then posed with open-ended questions regarding the usefulness of the tool followed by SUS standard questions for usability.

If neither of the preferences were suitable for the interviewee, the interviewee was provided with links to the online hosted website along with a Google Form survey to fill in to complete the assessment. The questions asked during any of these methods of gathering input were the same. The survey was online for 7 days. The list of the interview questions can be found in appendix A.

F. Communication

Communication focuses disseminating the research carried out to academia and the industry through, for example academic research papers like this one.

V. Introducing UML-Ninja Tool

UML-Ninja is an innovative web tool that automatically assesses UML and documentation practices in an OSS project. The main design goals for UML-Ninja are divided into two categories: usefulness and usability. Design goals for usefulness include:

- Functionality: Checking if the tool does what it is supposed to.
- Increased productivity: Checking if the tool helps accomplish tasks more efficiently.

Design goals for Usability include:

- Ease of use: Checking if the tool is easy to use.
- User friendliness: Checking if it is easy to understand the elements of the tools.

The development of UML-Ninja consists of three phases: data collection, data analysis and data presentation.

A. Tool Data Collection

The data required to analyse the projects and calculate the chosen indicators were obtained by combining data from two databases that our supervisors provided to us. One of the databases [33] contained data regarding the number of UML commits, number of contributor and miscellaneous data about the project that was obtained using GitHub API[4]. The other database, which was the conclusion of a study from Chaudron et al. [8], contained data regarding UML models for the 10 projects that we had chosen to assess. To obtain the data for the UML class diagrams in the projects, the tools Img2UML [27] and SDMetrics[5] were used. Img2UML was used to converts the UML class diagrams to XMI format that was later imported to SDMetrics to obtain properties of the class diagram, such as number of classes, number of attributes etc. XMI is a mark-up language that is intended to provide a standard way to exchange information regarding metadata.

The combined data regarding the 10 projects was then added to an SQLite[6] database that was local to UML-Ninja.

B. Tool Data Analysis

From the 21 indicators that we researched about in this paper, we implemented 12. The analysis of the data was done based on the data available and the chosen indicators. For example, the data for the indicator Document Editable (boolean) is different from the data of the indicator Document Contributor Ratio (Integer). Details about the indicators and the implementation choice is shown in table 2.

[4] https://developer.github.com/v3/
[5] https://www.sdmetrics.com
[6] https://www.sqlite.org

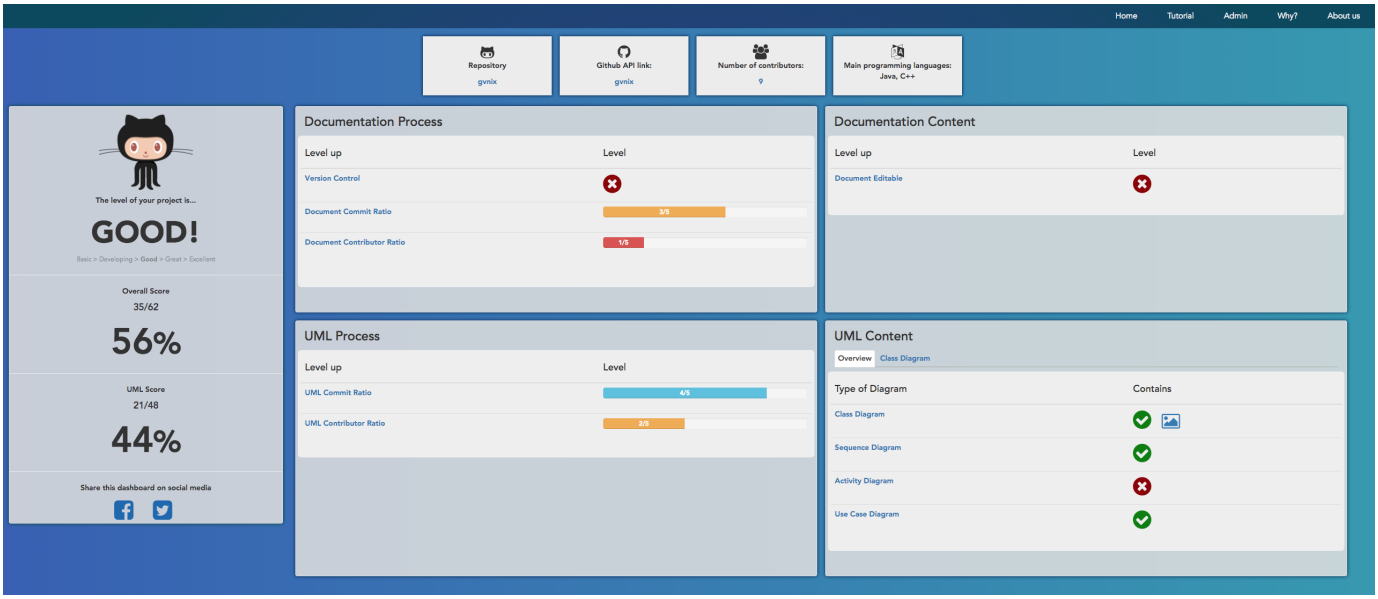

Fig. 1: Screenshot of UML-Ninja dashboard view.

Regarding the implemented indicators for UML process, documentation process and documentation content, we chose to implement them because of several reasons, including the importance of the indicators and value they add to the assessment of documentation, the type of available data about the OSS projects, time constraints and the level of complexity of indicator. Data regarding OSS project is different than a local software project. For example, data about every change in a GitHub repository is stored under GitHub's version control system whereas, this does not apply to a local software project.

We used the data that was accessible to us, to implement the respective indicators. For example, for UML contributor ratio we had access to the data about the number of people who added and updated the UML models as well as the total number of project contributors.

The motivation behind the UML content indicators that the tool currently has implemented is based on the importance of the class diagram metrics that are used to calculate values for those indicators. The metrics that are used for the 6 UML content indicators include: Number of Classes, Number of Methods, Number of Associations, Number of Attributes. These metrics are a part of Genero et al.s metrics [20] and the analysis of these metrics have been done in the following study done by Piattini et al. [34]. Piattini et als study concluded through empirical validation involving multiple experiments that the above stated metrics share a positive relation to qualitative features of the class diagram such as modifiability, understandability and analysability.

Table 3 shows the implementation details regarding the implemented indicator in UML-Ninja. Calculation and Analysis Method column describes the method with which the indicator is calculated. Level Value is the range of values an indicator can be in. Score Points is the range of scores an indicator can achieve based on the level value.

The levels for each indicator shown in table 2 were calculated and converted into a ratio of 0 to 1, and the ratio was converted into a 1 to 5 level or Yes/No value. The reason of converting the values into a 0 to 1 ratio was to have all the values on the same scale.

The following formula was used to convert the raw data of each indicator into a 1 to 5 scale level:

If you have number x in the range $[a, b]$ and you want to transform them to number y in the range $[c, d]$, the following calculation is required:

$$y = (x - a)\frac{d - c}{b - a} + c$$

The use of levels to rate the indicators is inspired by Dr. Scratch[7] [30] which is a web tool to automatically assess Scratch projects. Using levels is a type of gamification, for example, the user can level up if they improve the quality of the indicator. Information on how the levels are calculated can be found in appendix B.

Each indicator can either award maximum 5 points if it is on a 1-5 scale, or 2 points if it is a Yes/No indicator. According to [28], Class Diagram, Sequence Diagram, Activity Diagram and Use Case Diagram are the main UML diagrams. The project that is being assessed receives 2 more points for each diagram if they exist in the project. We chose the number of points to be rewarded as 2 for this because primarily,

[7] http://www.drscratch.org

we wanted the number to be between 1 and 5 for easier calculation and interpretation. Moreover, the indicator of these types award points to a project based on their existence in the project therefore awarding more points than 2 would not have been appropriate in regards to the efforts required.

All level points are summed up to produce an overall score. Based on the implemented indicators in the current version of UML-Ninja, the maximum overall score is 62 points. The score of each indicator is summed up to calculate the overall score of the project in term of documentation. The sum of indicator scores is divided by the maximum overall score, which is 62, and converted into a percentage.

$$DocumentationPercentageLevel = \frac{Sum of All Indicator Scores}{Maximum Overall Score} * 100$$

The level of documentation in the project is assigned based on the following percentage ranges:

- 0 - =< 20% = Basic
- 20% - =<50% = Developing
- 50% - =<60% = Good
- 60% - =<80% = Great
- 80% - =<100% = Excellent

Having levels is an attempt to gamify the process. For example, to level up, the user is expected to improve the quality of their project's documentation. Another reason is that it is a quick way to show the quality of documentation in the project. If the user wishes to explore the details they can check the details of each individual indicator and its score.

C. Data Presentation

As seen in Fig. 1, UML-Ninja has a dashboard to presents the information regarding the indicators and the scores. The dashboard view is divided into 3 parts, excluding the navigation bar at the top of the page that is used to navigate to other pages in the web tool:

a) Info Bar: The Info bar contains information regarding the current project the use is viewing:

- Repository: Name of the repository being assessed.
- GitHub API Link: Link to the repository's GitHub API
- Number of Contributors: Number of developers of the project.
- Main programming languages: The main programming languages used in the project.

b) Indicator Boxes: As shown in Fig 1, there are 4 indicator boxes on the dashboard view, each of which consist of indicators that fall in that particular category. The properties of the bars:

- *Indicator bar level*: This is used to show the level of the indicator. The levels are colour coded: if the indicator is at Level 1, it is shown in red, level 2 and 3 are shown in orange, level 4 is shown in blue, level 5 is shown in green. The colours represent the status of the quality level.
- Each indicator name is clickable which will open an information modal containing information regarding that particular indicator, as shown in Fig. 2.
- Each level bar of the indicator is also clickable which opens a graph modal that displays the variables that were used for the calculation of that indicator. The raw value of each indicator is also shown in the graph modal, as shown in Fig. 3.

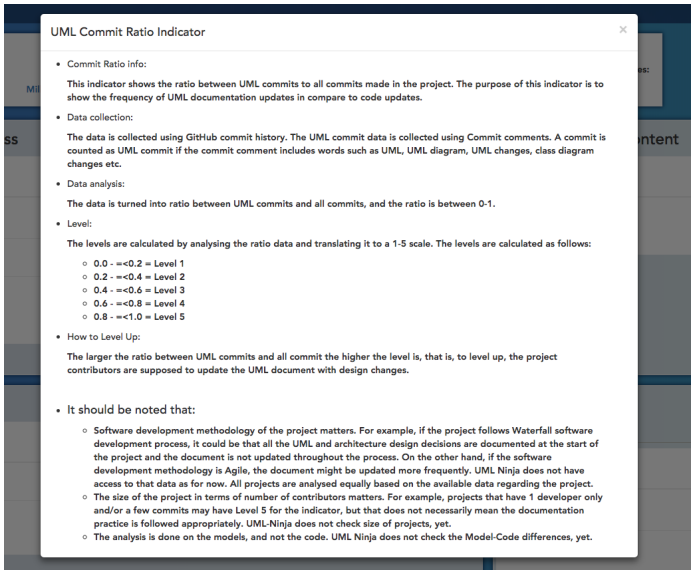

Fig. 2: Modal Information

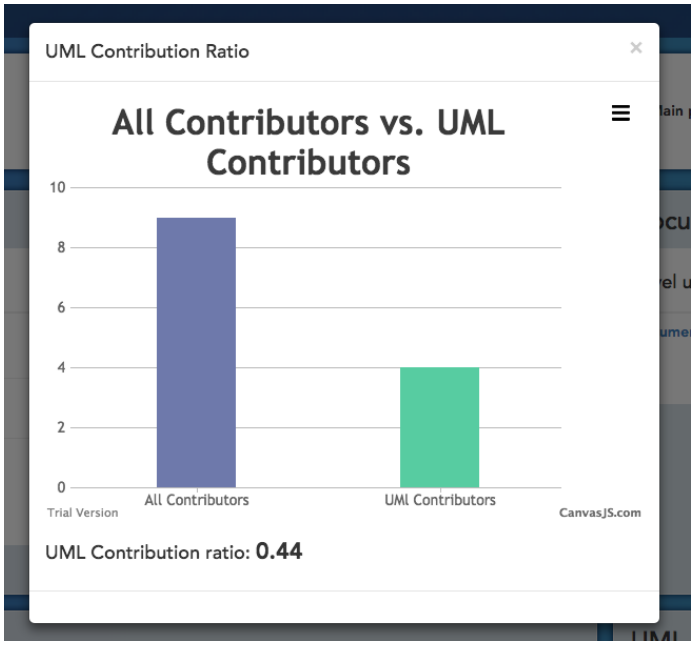

Fig. 3: Detail and Graph Modal of Indicator

c) Overview Box: The overview box is on the left side of the dashboard which contains the overall score of the project based on the indicators, UML score and the level of the project based on the overall score percentage. It also contains

TABLE III: DETAILS ABOUT THE INDICATORS IMPLEMENTED IN UML-NINJA

ID	Name	Type	Calculation and Analysis Method	Level Value	Score Points
1	Version Control	Document Process	Is the document under a version controlling system, such as GitHub?	{Yes, No}	{0, 2}
2	Document Commit Ratio	Document Process	$Commit\ Ratio = \frac{\#\ documnt\ Commits}{\#\ all\ commits}$	[1-5]	[1-5]
3	Document Contributor Ratio	Document Process	$Contributor\ Ratio = \frac{\#\ document\ contributors}{\#\ all\ contributers}$	[1-5]	[1-5]
4	Document Editable	Document Content	File format and extension, e.g., docx, pdf, txt etc.	{Yes, No}	{0 , 2}
5	UML Commit Ratio	UML Process	$UML\ Commit\ Ratio = \frac{\#\ UML\ Commits}{\#\ all\ Commits}$	[1-5]	[1-5]
6	UML Contributor Ratio	UML Process	$Contributor\ Ratio = \frac{\#\ UML\ contributors}{\#\ all\ contributers}$	[1-5]	[1-5]
7	Attributes vs Class	UML Content	$AvsC = \frac{\#\ attributes}{(\#\ attributes + \#\ number\ of\ classes)\hat{}2}$	[1-5]	[1-5]
8	Associations vs Class	UML Content	$AvsC = \frac{\#\ associations}{(\#\ associations + \#\ number\ of\ classes)\hat{}2}$	[1-5]	[1-5]
9	Methods vs Class	UML Content	$MEvsC = \frac{\#\ methods}{(\#\ methods + \#\ number\ of\ classes)\hat{}2}$	[1-5]	[1-5]
10	Average Parameters Per Method	UML Content	$APPM = \frac{\#\ parameters}{\#\ methods}$	[1-5]	[1-5]
11	Dependencies vs Class	UML Content	$MEvsC = \frac{\#\ dependencies}{(\#\ dependencies + \#\ number\ of\ classe)\hat{}2}$	[1-5]	[1-5]
12	Data Access Metrics	UML Content	$APPM = \frac{\#\ private\ attributes + \#\ protected\ attribute}{\#\ attributes}$	[1-5]	[1-5]

Facebook and Twitter icons, which upon clicking will allow the user to share the current status of the project to the social media channels.

The information is structured and presented from high to low abstraction. The user is presented with a high level of abstraction regarding the documentation quality of their project. If they wish to know about the implemented indicators or the reasoning behind their score or how to level up, they can get the relevant information by clicking on the indicator, upon which an information modal is displayed. Fig. 2 shows an example information modal for one of the indicators. At the bottom of the information modal, we have mentioned a few points that need to be considered when using UML-Ninja. They can be seen in Fig 2.

UML-Ninja can be accessed at http://204.48.31.89/.

VI. Evaluation of UML-Ninja Tool

In this section, we describe how UML-Ninja tool was evaluated. For the evaluation of our solution, we contacted prospective participants belonging to the user profiles stated in the introduction. Contacts to most of them were made through email with a few being done verbally when applicable. Our preference was to have an in-person or online interview to better understand the thoughts and opinions of the interviewees.

A. Approach of UML-Ninja

The approach taken by UML-Ninja was to automate the process of assessing documentation in OSS and show the information in a dashboard view in an orderly way. In the interviews we asked interviewees about their opinions of the approach. The main themes of the feedback regarding the approach included that the approach is a ”good idea”, ”useful”, ”quick”, ”elegant” and ”efficient”.

There was also critical feedback. The main critical feedback was regarding the use of the indicators and how they were selected and assessed. An interviewee was ”not sure how accurate the finding were”, another stated that the tool ”should also be able to analyse the current state of the project and could be able to tell about the timeline of the documentation and how it was evolved”. Another interviewee said that the approach is a ”quicker way to process result for a large project, where there are lot of stuff stored in the repos. However, the update frequencies of UML models are not taken into consideration, I believe this aspect is also important to be included”.

B. Usefulness of UML-Ninja

To understand the respondent‘s perspective on the usefulness of the tool, open ended questions were asked during the first phase of the interview. More specifically, there were 3 questions that were set-up to receive direct feedback regarding

usefulness of this tool. The feedback to first question - *"Do you think this tool will help you accomplish your tasks?"* were mostly positive with 12 out of 14 respondents answering "Yes". A following 10 of those 12 respondents detailed their feedback. 4 of the respondents stated how "Easy" it is to accomplish their tasks and also stated how this tool will help them save time. One of those 4 respondents was the developer of Dr. Scratch tool who stated that "the tool is on the right track for usefulness for researchers and developers but might need more testing for developers." The other 3 respondents were divided respectively into 2 researchers and 1 student. Moreover, the summarization aspect of the tool was appreciated and brought up by 5 respondents. A researcher from University of Gothenburg who is researching on "Threats that prevent developers to adapting to research approaches" was one of the respondents who supported that feature and stated that it would save time to assess the quality of documentation.

The next question that helped produce direct feedback for the usefulness of the tool was directed towards motivation - *"As a developer, do you think this tool will motivate you to improve quality of documentation of your project?"*. 11 out of 14 respondents answered Yes to this, including researchers who had recently or still play the role of a developer. One of the themes that was brought up by the respondents was gamification of documentation analysis to which, 4 respondents reacted positively stating it as the reason behind motivation to improve documentation quality. During the interview, the theme Visibility was also bought up by 4 respondents who stated that it would not be the UML-Ninja tool itself but the method of its deployment that will motivate them to improve documentation quality. Enforcement by product owner, collective use as a team, deployment on easily visible interface are the aspects of deployment stated by the respondents that would further motivate the user to use this tool to improve documentation quality. "Quick Feedback" was another qualitative feature of the tool that can serve as a motivation as stated by 3 developers, 1 student and 1 researcher.

On the other hand, there was criticism on the purpose of the tool that was provided by a researcher who is an associate professor of Software Evolution at the Model-Driven Software Engineering group in a technical university in Netherlands. It was stated that this tool could promote "superficial improvement of documentation quality" instead of actual improvement of documentation quality as developers could be focussed on improving the scores of indicators present in the tool which would not necessarily improve the overall documentation quality. In addition to this, feedback from the creator of the Dr. Scratch tool also made it clear that this tool could instead be perceived to be used as merely a status check rather than a motivation to improve documentation quality.

During the interview, an interesting observation was bought up by the developer of Dr.Scratch who stated that another possible user profile could be lecturers in Software courses. He further explained that lecturers can adopt this tool as a part of their teaching by enforcing his/her students to attain a certain score/level before submission of their work. The final question that provided direct feedback on the usefulness of this tool was aimed at gaining knowledge regarding whether or not the respondents would want to use the tool for their purposes. 4 respondents stated that they would not use the tool for their purposes as they do not work with software projects concerning UML, 3 of which were researchers with the remaining being a developer.

C. Usability of UML-Ninja

As mentioned in the previous sections, we are using SUS standard questions [19] to evaluate usability of UML-Ninja. SUS calculations produce a single number that represents a composition measure of the overall usability of the tool being evaluated. Calculating SUS score includes first summing up the score contributions from each question item.

Each of the individual score on the question, ranges from 0 to 4. For question items 1, 3, 5, 7 and 9, the score contribution is the position of the choice minus 1. For example, for question 1, if the position of the choice is 3, the score is 2.

For question items 2, 4, 6, 8 and 10, the score contribution is 5 minus the position of the choice. For example, for question 2, if the position of the choice is 3, the score is 2. After summing up the individual item scores, the sum is multiplied by 2.5 to obtain the overall SUS score.

SUS score values have a range of 0 to 100; it should be noted that it is not a percentage value.

In case of evaluating UML-Ninja, we calculated SUS score for each respondent then calculated the average SUS score by summing all the SUS scores and dividing it by 14; the number of respondents. The SUS score we achieved for UML-Ninja's usability was:

$$\frac{1082.5}{14} = 77.32$$

Fig. 4 shows the grade ranking of SUS scores. According to [23], score 77 is a solid C grade, meaning that usability of UML-Ninja is good but not excellent.

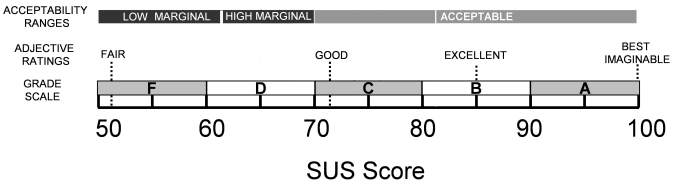

Fig. 4: Grade rankings of SUS scores from [23]

D. Limitations of UML-Ninja

One of the open-ended interview questions was aimed at gathering feedback on limitations of the tool. The main limitation as stated by 4 respondents was customizability. The tool does not currently allow for users to control the visibility

of indicators as well as re-arranging UI elements. Another aspect of customizability is to customize the threshold values of indicators.

Improper visibility of UI elements was the second main limitation as stated by respondents. They stated that though this tool provides information in order to improve documentation, the placement of this information is not appropriate as it needs to be in the forefront of all other data. All other data as stated by the respondents mostly constituted of the indicators and their levels. This limitation therefore became the foundation for another limitation that was brought up by 3 respondents that the system is too assessment driven rather than a motivator to improve documentation quality as the majority of the dashboard view of the tool displays the current levels of indicators depicting the quality of documentation of the project and not suggestions to improve the quality which is the aim of the tool.

Another critical feedback stated by a researcher was in regards to using static ranges to convert raw data from indicators to a 1-5 scale. It was brought to our attention that static ranges for conversion in general are becoming lesser adopted in tools and more accurate ways of converting ranges to a scale such as percentile are beginning to become a standard.

The data required to set a percentile by analysing multiple repositories with documentation was out of the scope for this research but can be applicable as an extension for future work for the tool to be able to more accurately analyse documentation.

Lastly, there was feedback provided regarding the choice of assessment criteria where some more important indicators such as difference between model to code, analysis of other UML models could be a more reliable reference to depict software documentation quality than the ones currently in use.

Furthermore, the participants of the evaluation were posed by an open-ended question on improvements to this tool. The majority of the respondents, 6, brought up the topic of adding more indicators pertaining to mostly UML and documentation content. Additional UML indicators to analyse more UML models was the most common feedback in regards to addition of indicators. The second-most important feedback we received was regarding analysing software process, more specifically whether or not the chosen software process for a project is followed. A researcher also provided feedback regarding analysing content of GitHub ReadMe file to check whether it follows a documented structure or not. Addition of these indicators will make it more difficult for users to easily navigate our website, and therefore customization which was stated as a limitation, can be a part of the improvement plan.

Project comparison was also a request for a feature that was mentioned by several researchers as it would really help them in their researches to compare the documentation quality data of 2 or more projects. This improvement suggestion goes hand in hand with the improvement suggestion - Evolution. This suggestion would allow a user to see data progression and would be valuable for researchers to compare several projects or for developers to map the growth of the software documentation quality.

E. Threats to validity

In this section we discuss the relevant threats to validity and how we tried to mitigate them. We consider the four aspects of validity threats as presented by Wohlin et. al [24].

a) Construct Validity: To avoid misunderstanding, we presented a brief introduction and the purpose of the study to the interviewees as well as what we expect from them. We also included that information in the interview invitation emails. However, this threat exists because the way the interviews were constructed and carries out were different. There is a validity threat that the interviewees who answered the questions online, without our verbal introduction, misunderstood the questions or the purpose of the study.

b) Conclusion Validity: The degree of conclusion validity is limited in this study as we did not look into any relationships in our data. However, the number of interviews (14) we carried out to evaluate our tool and our approach was fairly small. Nevertheless, majority of the interviewees we interviewed are researchers who are equipped with profound knowledge regarding UML and documentation.

c) Internal Validity: The data UML-Ninja uses to assess quality of documentation in a project is obtained using other tools and services. Incorrect data or miscalculations from other tools affect the result UML-Ninja produces regarding the quality of documentation. For example, if the tool Img2UML is unable to identify a property of a class diagram or if the tool SDMetrics misinterprets the XMI structure produced by Img2UML, the results UML-Ninja show are directly affected. Furthermore, the data used regarding UML and documentation commits is based on the commit message. For example, if a commit comment is ”Update class diagram”, it is counted as a UML commit. This is an internal validity threat because if there is a mistake in counting UML or documentation commits, the result of those indicators that use this data for assessment is affected. However, as mentioned in section V, Introducing UML-Ninja Tool, the data that was provided to us by our supervisors had been validated. We do not have much control over these threats, they are potential threats to validity and cannot be ruled out in the current approach taken by UML-Ninja. However, to mitigate these threats, we use a local SQLite database and save the data about the projects there.

Another internal threat to validity is that the interest, experience and background of the interviewees may influence their feedback. For example, students or developers who may not think documentation is important may find UML-Ninja not useful or necessary. To mitigate this threat and obtain valuable feedback, we made sure the prospective

interviewees had some degree of competence regarding UML and documentation.

d) External Validity: The approach of UML-Ninja is focused on automatic assessment of documentation in OSS. To improve external validity, we selected interviewees who had different professions and roles, e.g., students, developers and researchers. Furthermore, the approach of UML-Ninja is tailored to automate assessment of UML-Ninja, the approach might not be applicable for other purposes or in different domains.

VII. Discussion

In this section, we discuss the answers to our research questions and relate the findings to the related works. Regarding the main RQ, *How can we automatically assess the quality of UML and documentation practices in open source projects?* in this paper we introduced an approach to automate the process of assessing documentation in OSS projects. As a result, we developed a tool, UML-Ninja, to do so. The results indicate that it is technically possible and applicable. UML-Ninja requires other existing tools and services to function and make the process automatic. The evaluation of the approach indicates that the approach is good and useful. However, in the evaluation it was also discussed that not everything can be done quantitatively, and assessing qualitative data is difficult, as it can be subjective. This is similar to the finding from Dr. Scratch [30]. Automation can help to some extent, e.g. it is quick, but it cannot guarantee that the output is correct or complete.

Regarding the two sub-questions, *How can the assessment of documentation be automated?* and *How can the assessment of UML documentation be automated?* the results show that the approach of dividing documentation and UML into two sub-parts; process and content, was reasonable. Our evaluation indicates that implemented indicators were relevant. In addition, there were suggestions regarding new indicators, such as checking coupling, source code to model ratio and checking more document types e.g., Readme files. Checking different type of files is similar to the related work of Carvalho [32] which was focused on assessing content of documentation only. UML-Ninja assesses the process of documentation as well, to provide a better overview on the quality of documentation and how to improve it. For example, the indicator UML content contributor ratio is related to the process of UML and checks the percentage of all members that actively contribute to designing and updating the UML models.

In regards to usefulness of the tool, 12 out of 14 respondents agree that the tool is in fact useful to achieve the final vision of improving software quality through improving documentation quality. On the other hand, the results also indicate that 3 of the researchers that we interviewed would not use this tool in their respective fields. The reasoning as stated to support this was mostly due to documentation not being an essential part of their field of work. In comparison to this, the remaining 3 researchers stated that they would use this tool as it provided an easy overview combined with indicators covering the vital aspects of documentation. This type of automatic analysis was not an element of Aversano et al.s study [9]. The above stated reasoning by the 3 researchers, was supported by a number of participants from the other 2 user profiles with 3 out of 4 developers willing to use this tool. 2 developers further stated better integration with other systems as mentioned in the evaluation, will serve as a motivation to use the tool. With this feature being requested by a further 2 students and 1 developer it becomes apparent that though the participants are willing to use the tool, a better integration with other systems such as GitHub and other dashboards will be vital to use this tool efficiently.

In addition to the respondent‘s feedback on the usefulness of this tool, benefits of using this tool over other existing methods were also identified by the participants with 4 researchers stating that the tool seems like a step up from manual analysis of documentation. The remaining two researchers did not agree due to not being convinced with the indicators used to assess documentation quality. 1 developer also had similar remarks which could be countered by making the system customizable which can allow users to add, remove or hide indicators based on preference or research. The collected feedback on indicator suggestions were mostly aimed at implementation of more UML Indicators as well as indicators regarding the synchronisation between code and documentation commits. In order to establish these indicators as part of the tool, Chaudron et al.s study [11] can be essential in prioritizing the importance of the UML indicator based on which UML model it pertains to. Their study could also be used to understand the current relation between code changes to document changes and its effects on software quality which can help us understand the severity of the issue on a larger scale.

Regarding the usability of UML-Ninja, our evaluation suggests that it is user friendly and easy to use. It is difficult to compare this with the related works because they have not taken usability into consideration when developing a tool. Usability is important for UML-Ninja because it is an online tool that is accessible by anyone. The aim is to make it easy to use by anyone who wants to assess the quality their project‘s documentation. In addition, usability is important to UML-Ninja because of the extensive amount of information on the dashboard view. It is important to for the user to find relevant information in an easy and quick manner. Overall, a 77 SUS score is a good grade for usability of UML-Ninja in the first-attempt. However, UML-Ninja was evaluated by 14 people only. If the usability is evaluated using a larger number of participants or users with different user profiles than we chose, the SUS number might be different.

VIII. Conclusion

In this paper, in order to automate the assessment of documentation and UML in open source projects, we present UML-Ninja, an online web tool with a dashboard. The goal of UML-Ninja is to be able to quickly assess documentation

practices in software projects that can help in improving documentation quality and to obtain relevant information regarding documentation quality. In order to cover the essential parts of documentation, the approach of UML-Ninja is to divide documentation into two parts and two sub-parts. In total there are 4 parts: Documentation Process, Documentation Content, UML process and UML Content. Each part as stated, includes specific indicators in order to assess the respective aspect of documentation. UML-Ninja grades the overall quality of documentation for a project. The indicators were selected based on research on the topic and discussions with our supervisors. To evaluate the approach, usefulness and usability of UML- Ninja, 14 interviews were carried out with different user profiles, including students, developers and researchers. Finally, a number of limitations of UML-Ninja were presented and explained.

UML-Ninja is the first attempt to automate the assessment of documentation quality in OSS projects. Based on the evaluation points, we realized that the tool can be improved in many ways, and they can be future works of this study. The future research work includes implementing more indicators, implementing a feature to make the indicators customizable in order to assess documentation for different stakeholders. Another possible future work can be implementing a feature to compare the same project from different time period or check the evolution of documentation in a software project lifecycle. Finally, an interesting future work can include integrating UML-Ninja with GitHub to give badges to projects based on the quality of documentation.

REFERENCES

[1] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*, 3rd ed. Addison-Wesley Professional, 2003.

[2] M. Shahin, P. Liang, and M. R. Khayyambashi, "Architectural design decision: Existing models and tools," in *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*. IEEE, 2009, pp. 293–296.

[3] P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford, "Documenting software architectures: views and beyond," in *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society, 2010.

[4] N. Moha, Y.-G. Gueheneuc, L. Duchien, and A.-F. Le Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 20–36, 2010.

[5] K. W. Miller, J. Voas, and T. Costello, "Free and open source software," *It Professional*, vol. 12, no. 6, pp. 14–16, 2010.

[6] S. A. Ajila and D. Wu, "Empirical study of the effects of open source adoption on software development economics," *Journal of Systems and Software*, vol. 80, no. 9, pp. 1517–1529, 2007.

[7] D. L. Parnas, "Precise documentation: The key to better software," in *The Future of Software Engineering*. Springer, 2011, pp. 125–148.

[8] R. Hebig, T. H. Quang, M. R. Chaudron, G. Robles, and M. A. Fernandez, "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*. ACM, 2016, pp. 173–183.

[9] L. Aversano, D. Guardabascio, and M. Tortorella, "Evaluating the quality of the documentation of open source software," in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*, 2017.

[10] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[11] M. H. Osman and M. R. Chaudron, "Uml usage in open source software development: A field study." in *EESSMOD@ MoDELS*, 2013, pp. 23–32.

[12] A. Tsiolakis, "Consistency analysis of uml class and sequence diagrams based on attributed typed graphs and their transformation," in *ETAPS 2000 workshop on graph transformation systems*. Citeseer, 2000.

[13] A. Wingkvist, M. Ericsson, R. Lincke, and W. Lowe, "A metrics-based approach to technical documentation quality," in *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*. IEEE, 2010, pp. 476–481.

[14] C. Larman, *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Pearson Education India, 2012.

[15] W. J. Dzidek, E. Arisholm, and L. C. Briand, "A realistic empirical evaluation of the costs and benefits of uml in software maintenance," *IEEE Transactions on software engineering*, vol. 34, no. 3, pp. 407–432, 2008.

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

[17] J. Wst. (2018) Sd metrics. [Online]. Available: http://sdmetrics.com/

[18] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[19] J. Brooke, "Sus: a retrospective," *Journal of usability studies*, vol. 8, no. 2, pp. 29–40, 2013.

[20] M. Genero, M. Piattini, and C. Calero, "Early measures for uml class diagrams," *Lobjet*, vol. 6, no. 4, pp. 489–505, 2000.

[21] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on software engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[22] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[23] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[24] G. H. Travassos, S. Pfleeger, and V. Basili, "Experimental software engineering: an introduction," in *1st Experimental Software Engineering Latin American Workshop-ESELAW*, 2004, rECHECK REFERENCE.

[25] R. Hilliard, "Using the uml for architectural description," in *International Conference on the Unified Modeling Language*. Springer, 1999, pp. 32–48, rECHECK REFERENCE.

[26] M. Lorentz and J. Kidd, "Object-oriented software metrics, a practical guide," 1994.

[27] B. Karasneh and M. R. Chaudron, "Img2uml: A system for extracting uml models from images," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*. IEEE, 2013, pp. 134–137.

[28] P. Langer, T. Mayerhofer, M. Wimmer, and G. Kappel, "On the usage of uml: Initial results of analyzing open uml models." in *Modellierung*, vol. 19, 2014, p. 21.

[29] M. Hitz and B. Montazeri, "Measuring coupling and cohesion in object-oriented systems," 1995.

[30] J. Moreno-León and G. Robles, "Dr. scratch: A web tool to automatically evaluate scratch projects," in *Proceedings of the workshop in primary and secondary computing education*. ACM, 2015, pp. 132–133.

[31] N. J. Kipyegen, W. P. Korir, and K. Njoro, "Importance of software documentation," *International Journal of Computer Science*, vol. 10, no. 5, pp. 1694–0784, 2013.

[32] N. R. Carvalho, A. Simoes, and J. J. Almeida, "Dmoss: Open source software documentation assessment," *Computer Science and Information Systems*, vol. 11, no. 4, pp. 1197–1207, 2014.

[33] H. Linero, *The Relation Between Code Documentation and Internal Quality of Software*, Master Thesis, 2018.

[34] "Journal of object technology," pp. 59–92, 2005.

APPENDIX

Appendix A: Interview Questions

Based on the Possible User Profiles stated above, how would you describe yourself as? *

- ○ Software Project Manager
- ○ Developer/Student
- ○ Researcher
- ○ Other: _____

1. Do you think this tool will help you accomplish your tasks? *
Refer to 'Possible User Profiles' to get an idea of the tasks related to this tool.

Your answer

2. If yes to Question 1, how do you think this framework will compare to other traditional ways of checking documentation quality in terms of ease of use?
More traditional ways would include manually checking documentation against its specifications.

Your answer

3. As a developer, do you think this tool will motivate you to improve quality of documentation of your project? Please motivate your answer *
If this does not apply to you, please write in 'N.A' or something similar in the answer.

Your answer

4. Do you see any limitations of this tool? If so, what are they? *
Limitations of the tool could vary from not having enough indicators to the UI not looking very professional.

Your answer

5. What are your thoughts on the approach used in this tool to analyse software documentation in terms of efficiency? *
The approach taken by this tool is to automatically assess documentation practices by analysing raw data of the project. The data is then used to calculate indicators of the project.

Your answer

6. Do you think you will use this tool to assess documentation practices in your project? Please motivate your answer. *

Your answer

7. Do you have any thoughts on how to improve this tool? (Including feature requests) *

Your answer

8. Do you have any ideas for indicators that can be implemented in this tool? *
Indicators in the website were for eg. Document Commit Ratio, Version Control, Attributes vs Class.

Your answer

Appendix B: Indicator Level Calculation

The levels for the UML content indicators are calculated as follows:

- 0.0 - =<0.2 = Level 1
- 0.2 - =<0.4 = Level 2
- 0.4 - =<0.6 = Level 3
- 0.6 - =<0.8 = Level 4
- 0.8 - =<1.0 = Level 5

The levels for the UML content indicators are calculated as follows:

- 1.0> - 0.8 = Level 1
- 0.8> - 0.6 = Level 2
- 0.6> - 0.4 = Level 3
- 0.4> - 0.2 = Level 4
- 0.2> - 0.0 = Level 5

Appendix C: SUS standard questions

1. I think that I would like to use this system frequently.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
2. I found the system unnecessarily complex.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
3. I thought the system was easy to use.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
4. I think that I would need the support of a technical person to be able to use this system.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
5. I found the various functions in this system were well integrated.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
6. I thought there was too much inconsistency in this system.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
7. I would imagine that most people would learn to use this system very quickly.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
8. I found the system very cumbersome to use.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
9. I felt very confident using the system.
   1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree
10. I needed to learn a lot of things before I could get going with this system.
    1. Strongly Disagree | 2. | 3. | 4. | 5. Strongly Agree