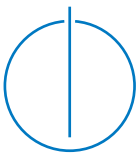
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Improving the Software Architecture Documentation Process for Mediawiki Software

Ankitaa Bhowmick



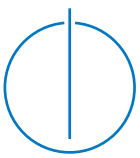
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Improving the Software Architecture Documentation Process for Mediawiki Software

Author:	Ankitaa Bhowmick
Supervisor:	Matthes, Florian; Prof. Dr. rer. nat.
Advisor:	Klym Shumaiev
Submission Date:	15th August, 2015



I assure the single handed composition of this master thesis in informatics only supported by declared resources.

Munich, 15th August, 2015 Ankitaa Bhowmick

Acknowledgments

Abstract

The thesis involves the initial research on the available state-of-the-art Software Architecture documentation processes, tools, etc. that help in maintaining a software architecture documentation that is consistent with the evolving architecture.

Understanding the current software architecture documentation process at Wikimedia, keeping the evaluation goals in mind, is an essential part of this thesis. It also focuses on critical evaluation of the documentation process to derive requirements for its improvement.

Based on analysis, an improved Software Architecture documentation process will be proposed and evaluated.

Contents

Part I.

Introduction

1. Introduction

1.1. Motivation

A good software architecture is the focal point of an evolving software([7]). To make this software maintainable, extendable and sustainable, a robust software architecture and a defined documentation process for this architecture are required ([4]).

Documentation is a factor that determines the quality of a software. A good software architecture documentation helps to understand, evaluate and communicate the various architectural decisions from different stakeholder viewpoints ([2]). Also, as the software evolves and its complexity and dependencies increase, the corresponding architecture documentation needs to updated as well([18]).

Standardized software processes and tools for Application Lifecycle Management provide structural support to a software engineering project's life-cycle. The quality of a software process directly affects the quality of the software ([6]).

Summing up, a standard process for documentation improves the quality of the documents and ultimately, the quality of the software itself.

1.2. About the Topic

Open source softwares have distinguished themselves as the trendsetters in the field of software engineering in this era and have demonstrated advantages which are beyond comparison. But there are a few downsides to this approach of software development ()[12]. In the pretext of software process, open source softwares can be categorized as loosely co-ordinated and less process-oriented ([19]). They believe in "Do-ocracy" where there is more focus of doing (building) the software from small to big, rather than following a process-oriented strict software life-cycle management process. This leads to the basic scope of this thesis : Improving the process in an open source environment

In the recent past, Mediawiki software (WMF Foundation) has grown to become one of the largest open source communities in the world. This prompted the choice for the candidate software for the thesis: Improving the process for Mediawiki software

As discussed above, software architecture documentation is as important in the software project as the software architecture itself. With some background study, it was found that lack of documentation is one of the major downsides of open source

development model ([5]) ([19]). Hence this thesis topic aims to find a proof of concept and a theoretical reasoning that may prove helpful for Open Source community in general and in particular : Improving the software architecture documentation process of Mediawiki software.

1.3. Research scope

The scope of the thesis has been reduced to maintenance of mid-level software architecture documentation of Mediawiki that is available as a part of the source code on mediawiki.org.

Moreover, a process has been defined and demonstrated that can be used as a basis for a process that can aide in maintenance of documents over a period of time. Coupling the existing review process and task management system, this documentation process is well-bound to the practices in the Mediawiki community and aims to win greater acceptance of the defined process. [5]

1.4. Reader's guide

The next chapter will enumerate the questions to which this thesis aims to provide an answer. This will help us understand our initial assumptions, the existing problems and the expected solution.

The following chapter will present literature analysis giving theoretical proofs to explain the important concepts for this research and the reasoning to support the thesis work (chapter 3).

Then, chapter 4 will show the approach followed to find a proper solution by conducting discussions and meetings with the stakeholders. The system design is also covered in this chapter.

The consecutive chapter will present a detailed description of the system implementation, defining all of its features (chapter 5).

With regards to chapter 6, the thesis focuses on evaluating the proposed solution by comparing it with the standard processes in the industry and also by evaluating stakeholder satisfaction

Lastly, chapter 7 will conclude the concepts of this work, its future scope and the answers to the initially proposed research questions.

2. Research Questions

2.1. Initial Hypothesis

A software architecture document is not just a necessary afterthought of architecture design ([2]), but an important contributor to the entire software design and development lifecycle. At an initial phase, for a new project, the software architecture document is produced as an artifact for software architecture views for different stakeholders. During the course of project lifecycle, the software architecture document grows and serves as an artifact to record important architectural decision made by the architects. At the design phase the software architecture document provides developers with a high level view of the software architecture and helps to understand the system interfaces, component interaction and basic functionality of each architectural component.

A software architecture document is not a static artifact. Rather, it is as dynamic as the software requirements itself ([2]). Maintenance of software architecture requires deep understanding of the skeleton system and depends heavily on its documentation. This escalates documentation to the highest position in the software evolution cycle. But usefulness of this document is measured by its relevance and consistency. This requires maintenance of the document itself to keep it as up-to-date as the current system. Thus, software architecture documentation is an integral activity that revolves not only at a software inception phase, during software architecture design, but also during the course of software's development maintenance and evolution. Since documentation is an activity, it needs to be regulated as a software process.

Software process is affected by organizational behavior of a community ([6]). Different organizations work on a culture specific to the standards and processes followed by the within their scope of control. In this context, Open Source software communities are noteworthy due to their relaxed process control and organizational structure. With regards to any form of artifact, especially documentation, this community is loosely coordinated where developers or contributors tend to code solutions without producing adequate documentation ([5]).

This brings us to an initial hypothesis that forms the basis for this research work on Software architecture documentation process: Open source software community lacks a process for maintenance of software architecture documentation. For a concrete example, Mediawiki was chosen as the ideal candidate. In the last few years the wiki

community (WMF - Wikimedia Foundaion) become one of the largest open source communities in the world. The software that runs the these wikis is the Mediawiki engine. The robust architecture of the mediawiki software is a complex system that has evolved over the years and its architecture complexity has grown manifolds. To explain its architecture, some documentation is available on "mediawiki.org". But to cater to new developers and first time users of mediawiki, architecture details and technicalities of architectural components is scarcely available on "mediawiki.org". Although some architectural component documentation is available as a part of the source code, this documentation is not well structured or available in wiki format. This deficit was realized as a part of the initial study and discussions with the stakeholders at mediawiki which will be elaborated in section 2.3. Hence, all the research and conceptualization of improved documentation process is based on these initial ideas. The following section lists the research questions that are intended to be answered by this thesis work.

2.2. Research Questions

1. RQ1 : How software architecture documentation process can be improved for Wikimedia Software?
2. RQ2 : What state-of-the-art architecture documentation process (methodology, tools) are available in the industry that meet domain-specific requirements – e.g. Open Source Software ?
3. RQ3 : What are the quality characteristics and metrics for evaluation of the software architecture documentation process?
4. RQ 4 : Which specific requirements of Wikimedia stakeholders should be met by documentation process for Mediawiki Software Architecture Documentation ?
5. RQ 5 : What process can be followed to automate the quality assurance of Software Architecture documentation in Open Source Software

The following sections will explain the reasons and requirements that lead to the formulation of the above-mentioned research questions :

2.3. Current state-of-art

The following sub-sections explain the current state of mediawiki software architecture documents and the current software and documentation process in the organization.

2.3.1. Software Architecture Documents

Mediawiki currently has all its software architecture documentation available on "mediawiki.org". The wiki pages belong to different namespaces such as "Manual:", "Help:" etc. to segregate them according to the intended information. Yet, these documents are scattered as a forest of links like any typical wiki, which makes it hard to follow for new users and harder to maintain for the existing users.

The available documents are useful for understanding some architecture components and help new mediawiki users to understand their installation, usage and operational details. But, these documents are not detailed enough for new developers to acquire a thorough understanding of the architectural component. Documentation of the Mediawiki core source code is auto-generated via "Doxygen" and is available at *https://doc.wikimedia.org/mediawiki-core/master/php* . This auto-generated code level documentation is always updated as a part of cron-jobs during deployment cycles and hence they are auto-maintained. The overview of Mediawiki's architecture was captured and written as a part of the book "The Architecture of Open Source Applications" by *Sumana Harihareshwara* and *Guillaume Paumier*. This documentation, availbale on "Mediawiki.org", is an excellent explanation of the various architectural decisions and corresponding rationale that were adopted over the years leading to the current architectural state of Mediawiki software. It is available under the "Manual:" namespace on "Mediawiki.org" and can be viewed for an abstract high level understanding of the system.

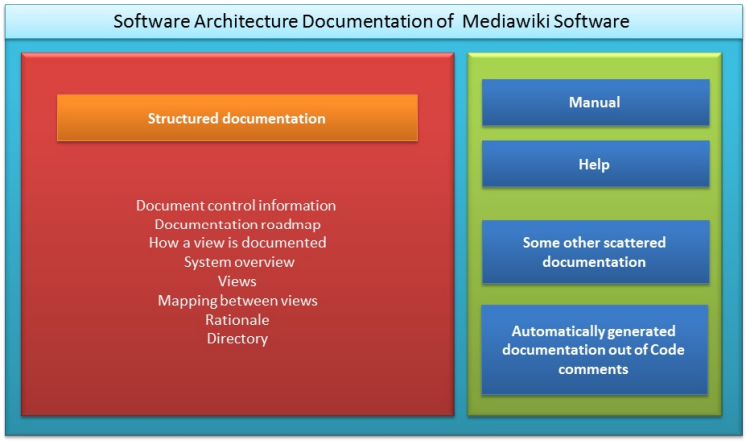


Figure 2.1.: Current state of documentation for different software architecture levels.

In Figure 2.1 we can see the current documentation structure that is available for different levels of detail of the software architecture. The green area indicated the

existing documentation and the red area in the indicates the lack of availability and maintenance of complete architecture documentation in accordance with the standard software architecture documentation structure [2] which covers the different views, rationale, etc.

2.3.2. Software Process

Mediawiki software community follows a process for maintaining its software (code base) that involves the interaction of the multiple systems for its review, versioning, tracking and task management. In this regard, before a piece of code is deployed into production environment, it is important to understand the role of the following entities as a part of the software process.

1. Developers : The software developers are the the most important functional entity of the software process in any software project or organisation. Similarly in the mediawiki community, the process is driven, managed and used by the developers of the software. Although other roles like software architect may exist as a subset of the stakeholders within the community, they all belong to the larger set of "Developers". Developers have the ultimate responsibility to implement and manitain the software process.

2. Maintainers : As the name suggests, Maintainers are developers with the acquired competence to take up the responsibility and become maintainers of different modules in the mediawiki code base *https://www.mediawiki.org/wiki/Developers/Maintainers*. They are instrumental in reviewing and following the software process and help to track and complete required functionality

3. Mediawiki BOTs : Besides human maintainers, Bots assume the role of semi-automated process to carry out maintenance activities that may be time-consuming or impossible to perform manually *https://www.mediawiki.org/wiki/Project:Bots*.

The above-mentioned entities need supporting systems to perform their daily activities as a part of the software process. In mediawiki, the software process activities are supported by following systems that simplify the process management activities.

1. Gerrit : Gerrit (*https://code.google.com/p/gerrit*) is a web-based code collaboration tool that has been adopted by the mediawiki community for managing the code base. This tool allows the review and maintenance of the master and forked branches of the mediawiki code repository and allows the developers to manage their contributions. The tools allows code management as a part of the software process of mediawiki which helps in easy maintenance of the software.

2. Phabricator : Phabricator (*http://phabricator.org/*) is an open-source task management and project communication platform that helps to manage different projects and their stakeholders within the organization. The mediawiki community has adopted the Phabricator to manage their daily tasks related to software development. The tasks can be managed according to projects, build versions, tags, etc by human maintainers. It provides features to discuss on issues related to the task and to also fork new related tasks.

Figure 2.2 shows the sequence diagram that explains a simple use case scenario : A software development task and the process followed for task management.

A developer may create a task on Phabricator to add/update a software functionality. He assigns the task either to himself or to another developer. The developed piece of code is pushed to an intermediate repository in Gerrit and awaits review. Once the code is reviewed and approves by senior developers, it is pushed to the authoritative repository which is ready for deployment. Once this is completed the task is finally closed.

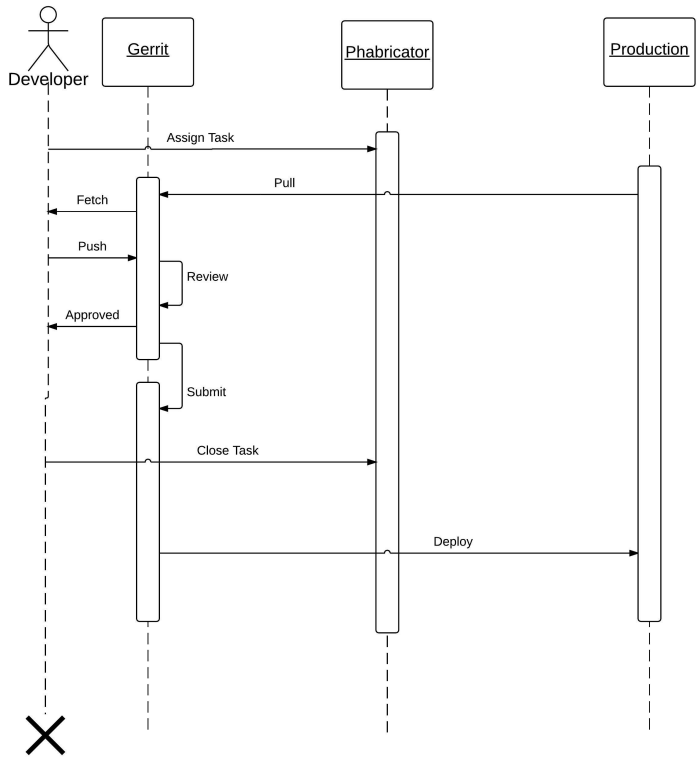


Figure 2.2.: Mediawiki Software Process Sequence diagram.

2.3.3. Documentation Process

Similar to their software process, the mediawiki community has a standard software architecture documentation process which involves the interaction of human maintainers and use of Phabricator for task management. Tasks for documentation activity are created manually, based on the need realized by developers. The management of the task is manual and its tracking, organization and management is supported by Phabricator

Figure 2.3 sequence diagram explains the use-case scenario : Manage a documentation task to update a document on "mediawiki.org"

In this case a developer himself may create/ assign a task on Phabricator for document update on "mediawiki.org". Once the update has been completed, a the task maintainer comments and closes the task on Phabricator.

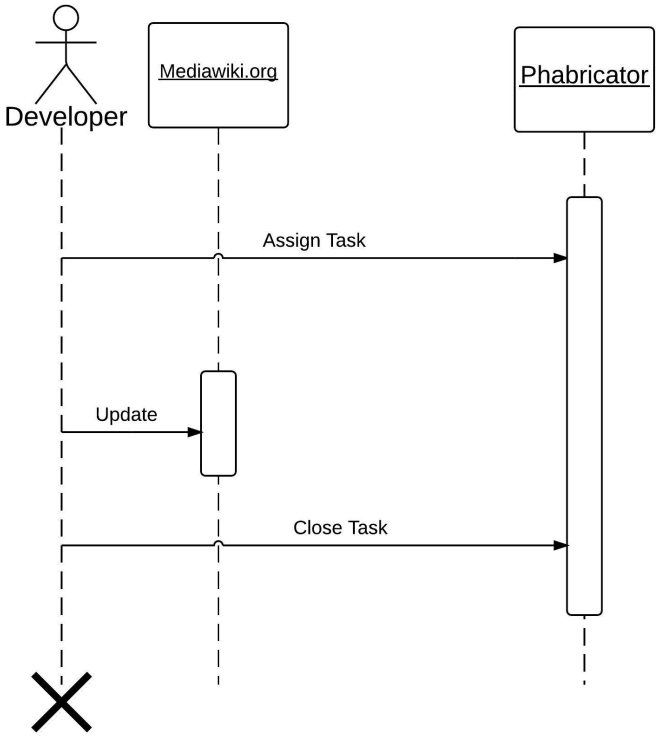


Figure 2.3.: Mediawiki Documentation Process Sequence diagram.

Understanding the current software documentation process leads to the following inherent problems and required improvements that need to be catered by answering

the research questions

2.4. Problems

This section elaborates on the problems that have been identified in the software architecture documentation process of Mediawiki that call for an improvement in the documentation process (RQ1).

2.4.1. Maintainability

As seen in the scenario covered in the previous section, it is evident that the documentation has shortcomings in terms of its maintainability with the rapid evolution of the software architecture itself. The process followed by the community is not strictly structured to ensure that the documents are maintained up-to-date. Phabricator may help to organize the task of documentation but does not guarantee the availability of precise documentation itself. Also only a manual check on document maintenance, without a strict process, is highly dependent on the motivation of the task owner to create, assign and complete the task. With the existing documentation process, a key requirement of document maintainability is not completely satisfied. Hence there is a need for an improvement to incorporate the requirement of up-to-date documents as a part of the documentation process.

2.4.2. Roles and Responsibilities

Mediawiki is an open source software community and hence it is not structured in its organization of well-defined roles and responsibilities. This poses a problem in defining, maintaining and following a strict process-based approach for software development and documentation. As compared to code maintainers mentioned in the previous section, there is no defined responsibility in the mediawiki community specially focused on documentation. The role of a developer for a certain architectural component implicitly assigns him the responsibility of corresponding documentation maintenance. But the lack of explicitly defined responsibility for the same creates a relaxed documentation process.

2.4.3. Availability and Management

An issue with the current documentation process is that software architecture documentation is not available under a single namespace or category and rather scattered in the wiki-forest. This makes it harder to manage the documentation and guarantee its availability on “mediawiki.org”.

2.5. Requirement Analysis

The above listed problems were identified to understand the requirements to be met by the improved process (RQ1).

2.5.1. Stakeholders

To understand a system and its architecture, it is important to understand the stakeholder perspective (RQ4). Mediawiki's software architecture documentation is available for developers, architects and system administrators on "mediawiki.org". Out of this the developers are the largest stakeholder group that access and use the architecture documents to the maximum. To cater to new developers various channels and features offer help in the form of mailing lists, IRC (Internet Relay Channel), Feedback dashboard, etc.

But a more concrete documentation needs to be prepared and maintained by the architecture component developers themselves. These detailed documents will help future developers to understand the software architecture in a more comprehensive way and on a more readable medium (mediawiki.org). This requirement was also realized during the "Mediawiki Developer meetup – 2009" which suggested the need for improved documentation and hinted on the usage of Bots for maintenance purpose.

Stakeholders play an important role in the implementation and maintenance of a process. Likewise in the case of documentation process, the developers are the key stakeholders who as both provider and user of the documents. As the developers understand their respective development in the best possible way, they themselves should prepare the documentation for the corresponding component/ feature/ module. This will help to capture the architecture decisions and rationale that can be utilized for future reference.

2.5.2. Meetings / Interactive Sessions

To understand the requirements from the perspective of stakeholders, sessions were held remotely and on-site with the members of the mediawiki organization at Berlin. These meetings and conversations gave a chance to understand the existing process and requirements for process improvement in a more detailed and focused manner (RQ2). The mediawiki representatives explained that although a compressed user guide could be copied along with a fresh wiki installation that includes basic information/ details in a concise yet understandable form, there is dire lack of a structured, detailed and complete architecture documentation within the community (RQ4).

Some documentation is available as a part of the source code for some architectural components. But the community prefers to have all documentation available on

"mediawiki.org"(RQ4).

The problem that documentation is often not updated / maintained due to lack of a strict process was realized within the community who wanted quality documents that were mostly up-to-date (RQ3). A process that streamlines this maintenance activity was put up as an important requirement during these meetings (RQ4).

The availability of guidelines to support the preparation of software architecture documents and assigning responsibility of its maintenance to developers or bots will assure the quality of the resulting documents (RQ5). The documentation is best understood and evaluated by the developers using them and thus, quality of documentation was indicated as an important requirement.

3. Literature Survey

This chapter aims to answer the previously formulated research questions by surveying already available literature and the related work in this direction This literature survey forms a basis in this thesis to derive ideas from existing examples and to come up with ideas to comceptualize the implementation work ahead. Also the related work helps to start with the initial idea and build upon it to derive a novel solution to solve the existing problems.

3.1. Points from Literature

This section elaborates on some facts and answers to research questions derived from existing literature. These points of reference help to build on ideas for finding solutions to the research questions formulated in this thesis.

3.1.1. Improved Documentation Process

How software architecture documentation process can be improved for Wikimedia Software and why is it required?

Software Architecture Documentation : A very extensive research and usage of software architecture documents, documentation process and evaluation has been covered in the book "Documenting Software Architecture- Views and beyond" [2]. The book suggests implementation of a "Package Module" for documentation that aims at collecting all relevant architecture documentation as a package i.e. all in one place. The architecture documentation is regarded complete when it captures the following aspects :

- Document control information
- Documentation roadmap
- How a view is documented
- System overview

- Views
- Mapping between views
- Rationale
- Directory

This makes the documents more available and improves ease of access. The book elaborates on capturing various views of the software architecture from the stakeholder's perspective and explains the structuring of documentation based on these "stakeholder views" (e.g) in this thesis the target stakeholders are the developers- hence a software architecture documentation that explains the architectural component overview and inter-component interaction specification is required for the better training of new developers and serve as reference for experienced developers.

The book also answers the question : "Why choose wiki ?". It suggests that documenting software architecture on a wiki platform ha several advantages :

- wiki-links are easy to navigate
- they provide easy formatting options
- wiki is easy to learn and more or less, intuitive
- it delivers nice readable web pages which provide editing and revision feature for version tracking and maintenance
- the wiki-pages are available/ accessible by all

Process and Community : Literature supports the idea the understanding the social environment of software communities has helped to understand their functional model and process-orientation. This understanding has lead way for guidelines and best practices to improve their current processes, as explained in [9]. The Open source culture of Mediawiki community poses limitations brought about by the relaxed process management and control [5]. Hence an improvement requires a process to be built upon the existing, available resources that can be easily be adopted or integrated into the environment. The "Eclipse Development Process" [Eclipse dev process] suggest that guidelines can be provided for new members such that they follow processes in a more self-regulated manner. The eclipse development process sets an example for open source communities by providing such guidelines for user groups like "committers" and "contributors".

Software architecture documentation is an inherent part of the software architecture

itself and is an integrated part of the architecture design process [16]. It is very important to document the software architecture as it helps to identify and record important decisions taken during the course of architecture design and also forms the basis for future architectural re-factoring.

Open Source Software Architecture documentation : The article "Empirical study of the effects of open source adoption on software development economics" [1] quotes that "When adopting an Open Source Software, software architecture documentation has a positive impact on the degree and cost of the software adoption". Thus, it is important for open source communities to offer concrete documentation to expand and enrich their contributing community. Some research has indicated that a lack of software architecture documentation maintenance in open source projects may hinder the use and further development of the software [11].

Research and empirical studies [5] have revealed that there is no dedicated role in open source communities to take responsibility of collecting, archiving , alighning and maintaining the software architecture documentation. This mandated the need for an advanced documentation process to handle the responsibility gap.

Documentation Level and Extent : The scope of detail in software architecture documentation "how much is enough and appropriate" and its need within open source developer community is largely dependent on "the contextual factors of software development, such as development method, rate of change, size of project, and architecture stability" ([8] [3]).

3.1.2. Current Industrial State-of-the-Art

What standard architecture documentation processes are available in the industry and practices of open source software?

Many industry standards have been defined in literature and practiced in the field of software engineering that support software process management and evaluation. Also, there is a possibility of adopting these standards in the open source community. Some research on the current practices in open source software development helps to understand the community processes and methodologies

Application Lifecycle Management tools :

Capability Maturity Model : For process oriented software engineering, CMM standards [14] have been set for process maturity evaluation. But open source communities are more focused on development maturity rather than process maturity.

IEEE1471-2000 standard : Documentation in standard software engineering projects follow the IEEE1471-2000 standard [2] that defines the outline for documentation of software architecture views and viewpoints.

Software process in Open Source community : The journal for "Systems and Software" [19] suveyed that over 61% of the open source projects also employ bug tracking tools, and a majority of projects use bug tracking tools provided by the host web sites.

Natural Language and visualizations : It was surveyed and studied that 70.4 % of the OSS projects use natural language with HTML as the main format for documenting software architecture [5].

XWiki :

3.1.3. Evaluating Documentation Process

What are the quality characteristics and metrics for evaluation of the software architecture documentation process?

Socio-Technical factors : Software quality is influenced by the way the community interacts [10]. The socio-technical environment within a community of developers who are geaographically separated and not bound by strict process control tends to introduce risk factors in terms of software quality. QualOSS assessment model [17] suggest that the organization of open source communities is loosely-bound and statistical research [19] proves that only about 20% of the open source projects have planned release dates . Some process oriented co-ordination approaches have been developed and adopted by open sorce software communities to manage their software processes .The "STIN" (Socio-Technical Interaction Networks)in Free/Open Source Software Development Processes [13] describes the well-established STIN ("Socio-Technical Interaction Networks")relationship for process enforcement by combining the socio-technical aspects that effect open source organizations.

Process Quality metrics : Maintainability and evolvability of the system should be supported by the software process [Documenting SA -Views].There is a direct correlation between the quality of the process and the quality of the developed software [6]. The article on "Software Process Roadmap" [6] suggests that the degree of maturity of the process is a main dimension of process assessment. Open source communities have can be evaluated on the basis of a few exceptional metrics in this regard [19] :

- Responsibility : level of user participation in open source projects is extremely high
- Organizational process : simpler feature-request process and easier transition from detection to debugging
- Efficiency : larger motivation of developer to propagate personal need to community need
- Collaboration : open source processes and tools for change management include cutting edge, large-scale collaborative software development

3.1.4. Stakeholder Requirement Satisfaction

Which specific requirements of Wikimedia stakeholders should be met by documentation process for Mediawiki SAD ? At mediawiki, the most active and important stakeholders are the coders or developers of the software.

Documentation within source code : The open source developers need to collaborate at a larger extent than traditional software systems. The daily work of developers within the community involves version control systems to manage and maintain the software repository. Thus, a requirement arises for collaboration in terms of code commit and source code consistency. Also, the documentation should be consistent with software version. It is always preferable and desirable that documentation confirms to the master branch source code.

Community acceptance : Any existing or newly incorporated software process should fit into the socio-technical environment of the open source community [Analysing the evolution of social aspects of open source software ecosystems].Community interaction helpa to understand the roles and responsibilities of community members as a part of the software process.

Architectural views : [2] stresses the a good software architecture document should provide the different architectural views for all the different stakeholders of the software. In Figure 3.1 the standard views in accordance with the software architecture has been mapped to stakeholder concern. With the specific case of mediawiki software where the Developers(Programmers) are the prime stakeholders, all documentation needs to be created and maintained for the developers and by the developers.

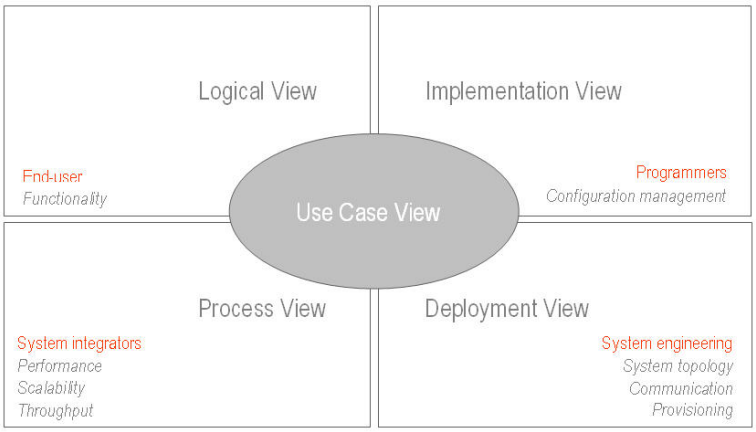


Figure 3.1.: "4+1" Unified View of the Software Architecture.

The document itself and the process to create/ update and maintain this documentation should assist the stakeholders and not add to cost of the software project ([15]).

Roles and Responsibility : The user management system is usually well-established in any software development organization which follow group management where user rights are group specific. Also, guidelines for user management with regard to the responsibility within the documentation process can be issued and followed [17].

3.1.5. Process Quality Assurance

How can the quality of SA documentation process be assured?

3.2. Idea Generation

This section covers the ideas that were derived from the literature to improve the documentation process for Mediawiki.

The book "Documenting Software Architecture – Views and Beyond" [2] suggests to define a page for architecture document in the exiting wiki. *Idea* : Use category feature of wiki to segregate "Mid-level Software Architecture Documentation" pages. Also, add templates on wiki page such that source-code consistent documentation belongs to non-editable parts and cannot be modified by sources other than Mediawiki developers.

Wiki is not able to track and edit past changes and only provides a discussion page. *Idea* : Add the software architecture documentation to version control system (along with the source code). Also, migrate document related discussion from wiki

page to task management system where documentation process can be tracked as a task.

Many options are available to capture documentation in wiki format. The book "Documenting Software Architecture – Views and Beyond" [2] suggests to use word2wiki to migrate word documents into wiki. Also XWiki (*http://www.xwiki.org/xwiki/bin/view/Main/WebHome*), a free wiki software platform includes WYSIWYG editing, OpenDocument based document import/export options, semantic annotations and tagging, and advanced permissions management.

"Software Process" [6] suggests to pay attention to the complex interrelation of a number of organizational, cultural, technological and economical factors.

Idea - It is wise to interact personally with members of the community to understand their specific requirements, already existing practices/ processes and try to improve it, rather than bringing in something completely new. This increases the acceptance of the process within the community

"Documenting Software Architecture from Knowledge Management perspective" [9] suggests to provide rationale for final architectural solution.Within a software organisation, the extent of design that constitutes its software architecture is based on its "context, domain, culture, assets, staff expertise, etc.". And this "thin line in the sand" must be made visible to all stakeholders. Also, it is important to "revisit, redefine and adjust" the architecture design decisions as the software and organization evolves. It is the software architect's responsibility to "make design choices, validate them, and capture them in various architecture related artifacts" [**Kruchten2008**]. The Mediawiki coding convention suggests having a text (.txt) file for the corresponding component in the source code *mediawiki.org/wiki /Coding conventions Documentation*

Idea : The documentation should be written by the architect/ developer as they understand the architectural components in the best possible way. While an architectural component is added/ updated, the corresponding text file documenting the component shpuld also be update. There can be inter-references between the code and document to find relevant parts easily.

The architectural viewpoint needs to capture details that are more abstract than source code functional details and less abstract than high-level architectural component interaction [9].

Idea : Capture the architectural component details and their functional details from a developer's view of the system to add relevant details as understood by current developer and as would be required for the future developer.

"Mediawiki.org" follows a standard user based rights system to grant permissions *(Manual:User Rights)* to user groups with the special case of BOTs that have the rights to access and modify "mediawiki.org" pages for huge volume of maintenance activities.

Idea : Apart from manual creation of documents by the Mediawiki developers, the responsibility for their maintenance can be partially automated by the usage of BOTs.

3.3. Important Terms, Concepts and Definitions

Software Architecture : SAD provides a blueprint of a software-intensive system for the communication between stakeholders about the high-level design of the system [5]

Complete – "good enough to meet our expectations for this system within the contect in which we are developing it" [2]

Software Process – Software processes are processes too ! [6]

Architecture Documentation of SA from knowledge perspective] – "If it is not written, it does not exist"

System : A collection of components organized to accomplish a specific function or set of functions (IEEE, IEEE Std 1471-2000 International Standard, Recommended Practice for Architectural Description of Software Intensive Systems, 2000).

Environment : Environment determines the setting and circumstances of developmental, operational, political, and other influences upon that system (IEEE, IEEE Std 1471-2000 International Standard, Recommended Practice for Architectural Description of Software Intensive Systems, 2000.).

Stakeholder : An individual, team, organization who has an interest in a system (IEEE, IEEE Std 1471-2000 International Standard, Recommended Practice for Architectural Description of Software Intensive Systems, 2000.).

Architectural View : A representation of a whole system from the perspective of a related set of concerns (IEEE, IEEE Std 1471-2000 International Standard, Recommended Practice for Architectural Description of Software Intensive Systems, 2000.).

Part II.

Thesis Contribution

4. Conceptualization

4.1. Idea Generation and Evolution

4.1.1. Initial Ideas

Analyse software architecture documentation Doxygen – tool for documenting S.A Who, what how ? Methodology for documentation

Different versions - arguments and decision-making, user scenarios discussions

4.1.2. A proof of concept

In Figure 4.1 we can see the

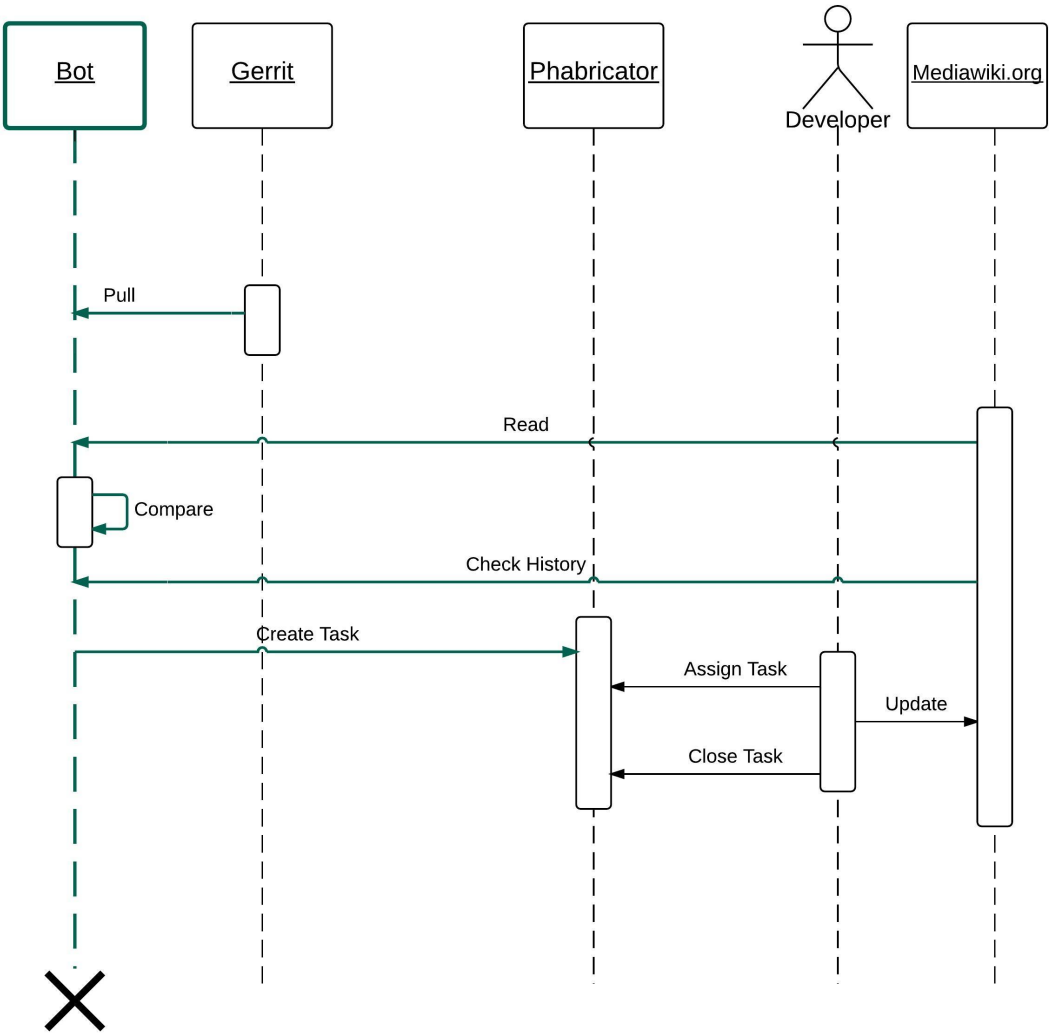


Figure 4.1.: Maintenace Bot Sequence diagram.

4.2. Improved Process

4.2.1. Process Details

Solves issues identified previously maintainable/ visible/ streamline more people into a process

5. Implementation

5.1. Architecture and Technical outline

5.2. Details of the implemented parts

5.3. Future implementation

Part III.

Evaluation and Conclusion

6. Evaluation

6.1. Meetings and Discuassions

6.2. Survey

6.3. Evaluation

7. Conclusion

7.1. Challenges

Acceptance within community Socio-behaviorial aspects of OSS community technical challenges

7.2. Benefits of implemented solution

7.3. arguments to support the idea

7.4. Concluding Remanks

List of Figures

List of Tables

Bibliography

[1] S. A. Ajila and D. Wu. "Empirical study of the effects of open source adoption on software development economics." In: *Journal of Systems and Software* 80.9 (Sept. 2007), pp. 1517–1529. ISSN: 01641212. DOI: 10.1016/j.jss.2007.01.011.

[2] F. Bachmann, L. Bass, P. Clements, D. Garlan, J. Ivers, M. Little, P. Merson, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Second. Addison-Wesley Professional, 2010.

[3] L. Briand. "Software documentation: how much is enough?" In: *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.* (2003). ISSN: 1534-5351. DOI: 10.1109/CSMR.2003.1192406.

[4] e. a. Crouch Stephen. *The Software Sustainability Institute*. Computing in Science & Engineering , vol.15, no.6, pp.74,80, . Dec. 2013. URL: http://www.software.ac.uk/.

[5] W. Ding, P. Liang, A. Tang, H. V. Vliet, and M. Shahin. "How Do Open Source Communities Document Software Architecture: An Exploratory Survey." In: *2014 19th International Conference on Engineering of Complex Computer Systems*. Aug. 2014, pp. 136–145. ISBN: 978-1-4799-5482-7. DOI: 10.1109/ICECCS.2014.26.

[6] A. Fuggeffa, A. Fuggetta, and P. Milano. "Software Process : A Roadmap Software Process : A Roadmap." In: 97 (1988).

[7] D. Garlan and M. Shaw. "Software Architecture: Reflections on an Evolving Discipline." In: *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (2011), p. 2. DOI: 10.1145/2025113.2025116.

[8] P. Kruchten. "Contextualizing agile software development." In: *Journal of Software: Evolution and Process* 25.4 (2013), pp. 351–361. ISSN: 2047-7481. DOI: 10.1002/smr.572.

[9] P. Kruchten. "Documentation of Software Architecture from a Knowledge Management Perspective – Design Representation." In: *Software Architecture Knowledge Management*. Ed. by M. Ali Babar, T. Dingsøyr, P. Lago, and H. van Vliet. Springer Berlin Heidelberg, 2009. Chap. 3, pp. 39–57. ISBN: 978-3-642-02373-6. DOI: 10.1007/978-3-642-02374-3_3.

[10] T. Mens and M. Goeminne. "Analysing the evolution of social aspects of open source software ecosystems." In: *Proc. 3rd Int. Workshop on Software Ecosystems* (2011), pp. 1–14.

[11] M. Michlmayr, F. Hunt, and D. Probert. "Quality Practices and Problems in Free Software Projects." In: *Proceedings of the First International Conference on Open Source Systems*. Ed. by M. Scotto and G. Succi. Genova, Italy, 2005, pp. 24–28.

[12] W. Scacchi. *Architectural Issues*. Vol. 69. Advances in Computers. Elsevier, 2007, pp. 243–295. ISBN: 9780123737458. DOI: 10.1016/S0065-2458(06)69005-0.

[13] W. Scacchi. "Socio-Technical Interaction Networks in Free/Open Source Software Development Processes." English. In: *Software Process Modeling*. Ed. by S. Acuña and N. Juristo. Vol. 10. International Series in Software Engineering. Springer US, 2005, pp. 1–27. ISBN: 978-0-387-24261-3. DOI: 10.1007/0-387-24262-7_1.

[14] SCAMPI Team. "Standard CMMI Appraisal Method for Process Improvement (SCAMPI) Version 1.3a: Method Definition Document for SCAMPI A, B, and C." In: March (2013).

[15] M. Shahin, P. Liang, and M. A. Babar. "A systematic review of software architecture visualization techniques." In: *Journal of Systems and Software* 94 (2014), pp. 161–185. ISSN: 01641212. DOI: 10.1016/j.jss.2014.03.071.

[16] M. Shahin, P. Liang, and M. R. Khayyambashi. "Architectural design decision: Existing models and tools." In: *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture* (2009), pp. 293–296. DOI: 10.1109/WICSA.2009.5290823.

[17] M. Soto and M. Ciolkowski. "The QualOSS open source assessment model measuring the performance of open source communities." In: *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*. Oct. 2009, pp. 498–501. DOI: 10.1109/ESEM.2009.5314237.

[18] S. Yeates. *OSSWatch*. June 2008. URL: http://oss-watch.ac.uk/resources/archived/documentation.

[19] L. Zhao and S. Elbaum. "Quality assurance under the open source development model." In: *Journal of Systems and Software* 66.1 (Apr. 2003), pp. 65–75. ISSN: 01641212. DOI: 10.1016/S0164-1212(02)00064-X.