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. To make this software maintainable, extendable and sustainable, a robust software architecture and a defined documentation process for this architecture are required.

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. Also, as the software evolves and its complexity and dependencies increase, the corresponding architecture documentation needs to updated as well.

Standardized software processes provide structural support a software development project's life-cycle. The quality of a software process directly affects the quality of the software.

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 trend of the trade in this era and have advantages which are beyond comparison. But there are a few downsides to this approach of software development. In the pretext of software process, open source softwares can be categorized as loosely co-ordinated and less process-oriented. 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 [**OssDoc**]. 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.

1.4. Reader's guide

The next chapter (chapter 2) 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, **??** will conclude the concepts of this work, its future scope and the answers to the initially proposed research questions.

2. Research Questions

2.0.1. Initial Hypothesis

A software architecture document is not just a necessary afterthought of architecture design [DSAView], 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[DSAView]. Maintanace 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 [SoftwareProcessRoadmap]. 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[OSSDoc].

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 has grown to become one of the top most open source communities in the world, powered by 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 at a high level, some documentation is available on "mediawiki.org". But to cater to new developers and first time users of mediawiki, the mid-level architecture details and technicalities of architectural components is scarcely available on "mediawiki.org". Although some 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. Hence, all the research and conceptualization of improved documentation process was based on these initial ideas.

2.0.2. Research Qestions

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 S/W ?
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 SAD ?
5. RQ 5 : What process can be followed to automate the quality assurance of SA documentation in OSS

The following sub-sections will cover the research questions :

2.1. Current state-of-art

2.1.1. Problems

2.1.2. Maintainability

2.2. Requirement Analysis

2.2.1. Stakeholders

2.2.2. Meetings

3. Literature Survey

Answer research Questions from literature Derive ideas from existing examples, come up with process ideas when no existing example is available

Part II.

Thesis Contribution

4. Conceptualization

4.1. Idea Generation and Evolution

4.1.1. Initial Ideas

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

4.1.2. A proof of concept

In Figure 4.1 we can see the

4.2. Improved Process

4.2.1. Process Details

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

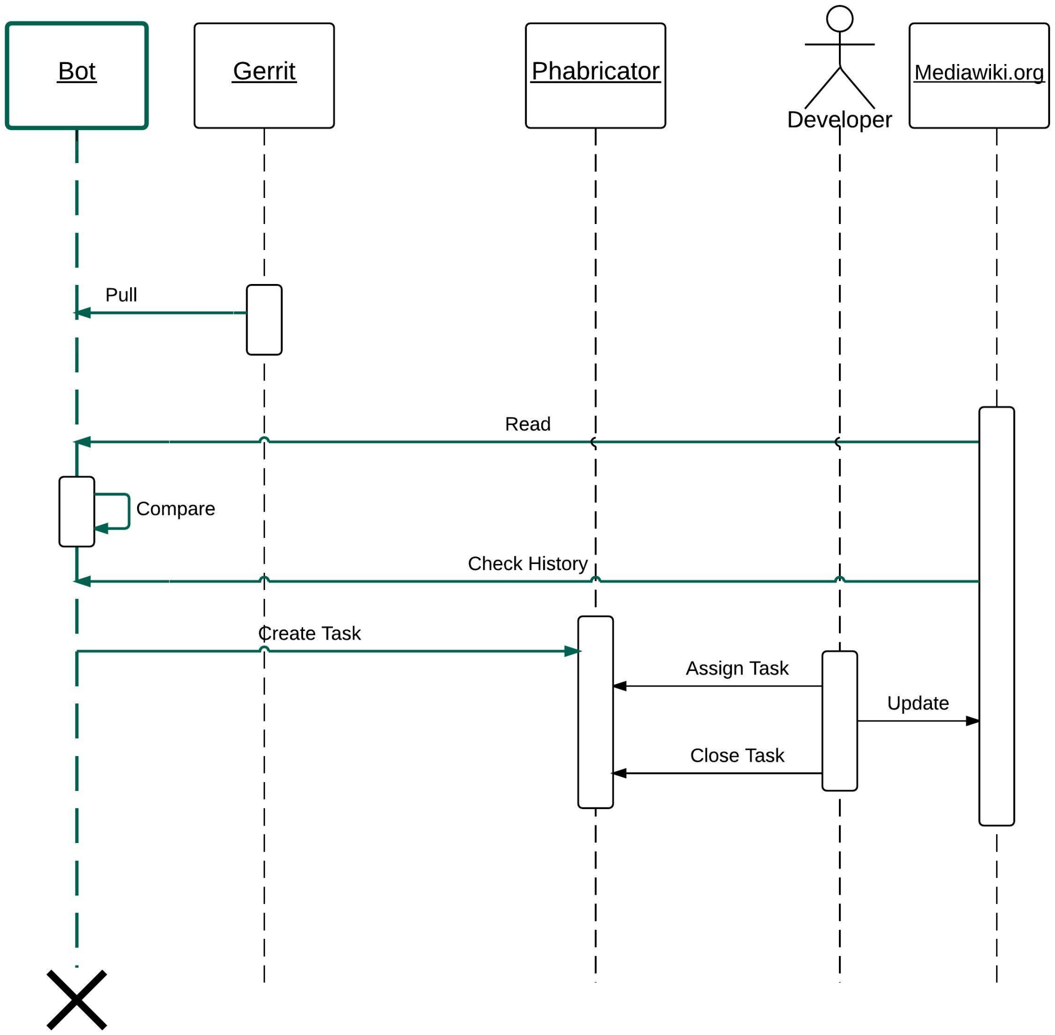

Figure 4.1.: Maintenace Bot Sequence diagram.

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