

Name: Gaurav Kumar Singh

Registration Number: 19BCE2119

Course: Software Engineering (E2)

Digital Assignment 2

Paper-I

An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles (2018) *Ana M. Fernández-Sáez & Michel R. V. Chaudron & Marcela Genero*

Abstract

UML (Unified Modelling Language) is one of the key modelling languages in the field of software engineering. UML provides a standardized way to visualize the design of the system. UML's origin is co-related to the OOPs (Object Oriented Programming). It consists of several diagramming techniques and methods which make it easier for the developers to visualize various aspects such as scale, actors, timeline etc. of the project. All this directly impacts the software maintenance strategy of the project as well. Software maintenance takes one of the biggest shares from the timeline for development of required software. Perception of purpose of UML may vary from person-to-person. UML can be considered as communication tool, overview, prototype guideline, testing guideline or can be for one's own understanding as well.

Scale of software projects can grow exponentially in very small amount of time and after a certain threshold it becomes very difficult to keep track of all the developed and developing modules of the project, therefore proper maintained documentation of the project is important to be able to maintain the software more easily. UML is one of the most efficient ways to quickly and accurately visualize different aspects of the project/ module. It requires proper experience and training to be able to develop well formulated UML for its referencing to be effective to be used for software maintenance of the project.

Paper-II

Analysing Forty Years of Software Maintenance Models (2017) *Valentina Lenarduzzi, Alberto Sillitti, Davide Taibi*

Abstract

Software maintenance mechanism has naturally evolved in the last forty years to adapt to the changing software development needs, models, programming languages, scale etc. Software maintenance tools are using increasingly advanced prediction models to cater to this rapidly growing industrial sector. In this paper, the author uses ISO/IEC 25010:2011 definition and reports on models for analyzability, modifiability, modularity, reusability, and testability using literature reviews following systematic snowballing.

The following points were concluded by the end of the research:

- 1) Continuous increase in number of papers: This implies software maintenance is still an increasingly popular research topic. More number of software maintenance models are being proposed everyday each with their own set of pros and cons.
- 2) Continuous increase of the complexity of the models: The normalized complexity of every sequentially proposed software maintenance model is increasing due to discovery of improved mathematical techniques, visualization techniques, advanced deep mining and neural network approaches which increase the reliability and adaptability of the software but is still limited in its usage.
- 3) Increase in the size of data analyzed: Scale of software projects substantially increase every decade due to availability of enormous open-source codes which are very well maintained by the community concerned and modified for organizations own needs.
- 4) Limited evolution of metrics adopted: Despite of proposal of innumerable amount of software maintenance models, the analysis metrics have not changed much in the last four decades.

Paper-III

A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts (2015)
Uttamjit Kaur, Gagandeep Singh

Abstract

Software maintenance has many challenges and requires a lot of efforts to be done properly. This paper suggests some issues and problems encountered during the process of software maintenance. Software maintenance is often time consuming and costly.

Software maintenance serves the following purposes to the project:

- 1) Updating the software
- 2) Software upgrades
- 3) Continuity of Service

Maintenance costs are mostly by-products of software improvements rather than corrections. Maintenance of software in general is a tedious task since it requires one to have thorough knowledge of the source-code in question of updating, upgradation or integration.

It opens another entry of risk which is cost estimates of product. Since, the maintenance cost is tough to predict due to its volatile nature in general. Cost estimate is a very common risk to have in software projects. Maintenance issues could be concerned with staff turnover, cost, deadline/timeline, product quality, database size, documentation quality etc.

Maintenance issues can be minimized through:

- 1) Eliminating Dead Code
- 2) Decreasing Turnovers
- 3) Reducing unnecessary and redundant complexity from project
- 4) Re-evaluation of software maintenance model to stay up-to-date
- 5) Quality testing


- 6) Bug Elimination
- 7) Increasing understandability of documentation

REFERENCES

- ❖ *Ana M. Fernández-Sáez & Michel R. V. Chaudron & Marcela Genero (2018) An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles*
- ❖ *Valentina Lenarduzzi, Alberto Sillitti, Davide Taibi (2017) Analyzing Forty Years of Software Maintenance Models*
- ❖ *Uttamjit Kaur, Gagandeep Singh (2015) A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts*

IMAGES

An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles

Ana M. Fernández-Sáez¹  **• Michel R. V. Chaudron² • Marcela Genero¹**

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract UML is a commonly-used graphical language for the modelling of software. Works regarding UML's effectiveness have studied projects that develop software systems from scratch. Yet the maintenance of software consumes a large share of the overall time and effort required to develop software systems. This study, therefore, focuses on the use of UML in software maintenance. We wish to elicit the practices of the software modelling used during maintenance in industry and understand what are perceived as hurdles and benefits when using modelling. In order to achieve a high level of realism, we performed a case study in a multinational company's ICT department. The analysis is based on 31 interviews with employees who work on software maintenance projects. The interviewees played different roles and provided complementary views about the use, hurdles and benefits of software modelling and the use of UML. Our study uncovered a broad range of modelling-related practices, which are presented in a theoretical framework that illustrates how these practices are linked to the specific goals and context of software engineering projects. We present a list of recommended practices that contribute to the increased effectiveness of software modelling. The use of software modelling notations (like UML) is considered beneficial for software maintenance, but needs to be tailored to its context. Various practices that contribute to the effective use of modelling are commonly overlooked, suggesting that a more conscious

Analyzing Forty Years of Software Maintenance Models

Valentina Lenarduzzi
Faculty of Computer Science
Free University of Bolzano/Bozen
Bolzano, Italy
valentina.lenarduzzi@unibz.it

Alberto Sillitti
Institute of Information Systems
Innopolis University
Innopolis, Russian Federation
a.sillitti@innopolis.ru

Davide Taibi
Faculty of Computer Science
Free University of Bolzano/Bozen
Bolzano, Italy
davide.taibi@unibz.it

Abstract— Software maintenance has dramatically evolved in the last four decades, to cope with the continuously changing software development models and programming languages and adopting increasingly advanced prediction models. In this work, we present the initial results of a Systematic Literature Review (SLR), highlighting the evolution of the metrics and models adopted in the last forty years.

Keywords—component; Software Maintenance, Systematic Literature Review

I. INTRODUCTION

Software maintenance has been under investigation for more than forty years, with numerous kinds of analysis performed by numerous authors, including those of this paper, and dealing with a variety of process and product aspects [1, 2, 3, 4, 5, 6, 15, 16, 17, 18, 19]. In this work, we introduce the results of a Systematic Literature Review (SLR) depicting the evolution software maintenance models and analyzing the how they have evolved over the past four decades.

The goal of the SLR were to:

- Analyze a chronological overview of the research on maintainability models
- Analyze how maintainability models have evolved
- Understand how the metrics used in maintainability models evolved

We analyzed 1,044 papers written from 1970 to 2015.

Other reviews on software maintenance have been published in recent years [7, 8, 9, 10, 11, 12, 13]. However, our work differs from them in terms of the timeframe, as we consider papers published over 40 years, and comprehensiveness, as we focus on a broad ranges of maintenance models.

As for software maintainability, we consider the ISO/IEC 25010:2011 definition and therefore report on models for

Library, Science Direct, Scopus, Google Scholar, Citeeaser library, Inspec, Springer link, as suggested by Kitchenham [14] [15].

TABLE I Search terms

P	software maintainability	P terms	<i>software maintain, software maintainability, software maintaining, software maintains</i>
I	model	I terms	<i>prediction model, estimation, characterization model, tool model, approach model</i>

Based on the keywords identified in Table I, we adapted the following query for each bibliographic source:

("prediction" OR "estimation" OR "characterization" OR "tool" OR "approach") AND "model" AND ("mainten*" OR "maintain*") AND "software".

We included only papers published in international journals and conferences in the field of Computer Science. We excluded papers that were not peer-reviewed and those not written in English. Moreover, we excluded papers that do not contain any references to maintainability fields in the title and abstract. We also excluded pure maintenance papers in which only a maintenance process was presented without proposing models for assessing and/or predicting maintenance.

We identified 1,044 papers published between 1978 and 2015. We rejected 789 papers after the application of inclusion and exclusion criteria for both title and abstract, and 94 (66 papers for Quality Assessment) after the full read, and we added 10 additional papers from the references, thanks to the systematic snowballing process. Finally, we applied the assessment criteria (see Table II) and ended up with 78 papers that passed the assessment criteria process.

TABLE II Assessment criteria

Assessment	
-------------------	--

A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts

Uttamjit Kaur
Department of Computer Science
GIMET
Amritsar

Gagandeep Singh
Department of Computer Science
GIMET
Amritsar

ABSTRACT

Software Maintenance and evolution are identifying by their huge cost and slow speed of implementation. Survey showed that around 60% of the maintenance effort was on the total cost of software. But after delivering the software to the client, the maintenance work begins. This paper suggests some issues and problems faced by software maintenance process. There are some issues of software maintenance i.e.: database size, system age, maintenance budget, system size, staff size or restructuring for change. This paper presents several ways to reduce cost and efforts involved in software maintenance. Software maintenance costs can be reduced significantly if the software architecture is well defined, clearly documented, and creates an environment that promotes design consistency through the use of guidelines and testing quality.

General Terms:

Software Maintenance, maintenance cost reduction.

Keywords

Software maintenance, issues and problems in software maintenance, cost and challenges in maintenance, maintenance cost reduction.

1. INTRODUCTION

Software does not get tired, it need to meet some new requirements that enhanced the software system. The changes made in the software system can be performed by software maintenance. Software Maintenance comes under process when a software team delivers a successful project to its client within a fixed time. Software Maintenance is the modification of a software product after delivery. The Modification can be done to improve performance, correct faults and to adapt the product to a modified environment [1]. According to ISO

1.3 Providing Continuity of Service

Maintenance focus on recovering from failures such as hardware or software and changes in the operating system.

Three type of software maintenance considered: Corrective Maintenance used for emergency program fixes and routine debugging; Adaptive Maintenance used to making changes in response to technology changes; Perfective Maintenance used to improve documented and requested enhancement [3]. Survey showed that around 75% of the maintenance effort was on the adaptive and perfective maintenance. Or 21% consumed by error correction [4]. Maintenance costs depend on the Number of changes and the costs of change depend on the maintainability. Important issues of software maintenance are limited understanding. Limited understanding refers to how quickly a software engineering can understand where to make a change or correction. In software 40-60% of the maintenance effort is devoted to this task. Limited understanding is to produce documentation. Documentation is lacking or incomplete and the people who know the software leave or retire without being replaced. Other issues i.e. customer priorities, staffing, and cost estimation with some technical issues i.e. testing, maintainability. If we want to reduce the overall cost of software, the goal of development should be reduce the maintenance effort [1].

2. MAINTENANCE PROBLEMS

Maintenance costs are due to software improvement rather than corrections. Most problems of software maintenance are associated with the software development process [3]. In software process, software engineer develop the part of the software, which may maintaining the software, has to get to get with the detailed design and functioning of the source code. Maintenance cost and effort is very important part of a