Change Requirement Management Issues for a Large Software Development Projects

Hassan Osman Ali1
e-mail: xaseey@gmail.com

Mohd Zaidi Abd Rozan2
e-mail: mdzaidi@utm.my

Abdullahi Mohamud Sharif3
e-mail: saakuut22@kudhan.so

Abstract — There are no major organizations that are free from the challenges of initiating, developing and implementing effective software changes management. Only few project managers got the ability to manage the change efficiently. As a result, software practitioners recognize that strategic change is not temporary issues but it is continued process. Software Change requirement management approach is required for constantly evolving systems to support the comprehension, implementation, and evaluation of changes. A lot of research efforts have been spent on this subject over the last twenty years, and many approaches were published likewise. But, there has not been an extensive attempt made to develop an approach that can support current software change requirement management in practice. Therefore, this paper examines issues of software change requirement management approach and investigates and compares current practice issues of a software change requirement management approach, examining their strengths and weaknesses. It also reviews existing popular tools and models of software change requirement management and how it supports to the current software change managements. Finally, it will identify existing issues of current processes and suggests the need of proper approach that can support to the current practice.

Keywords – current process issues; change requirement management; project management practitioners; change control

1. INTRODUCTION

Performing Software change requirement management is an important step when changing or maintaining software, especially in incremental processes. Usually, changing requirements during the software development is one of the developers’ expectations and cannot be avoided. The hardware and software platform changes, customer needs evolve, defects are found and so on. However, changing requirements is considered to be the most expensive [57] and long-lasting phase in the lifecycle of most software systems, where more than 50% of all maintenance costs arise from changing software [58]. Consequently, it is difficult to have complete management over and experience what is the common theme of change to project otherwise, it might cause project deterioration and other different problems. Software change requirement management approach and change impact analysis are same in this context. Hence, “the process of evaluation, the implications of realizing a change is termed as software change impact analysis” As software project management practitioners recognize, that software development complicity is increasing nowadays, so the need of software change requirement management approach is highly required. Therefore, this paper identifies existing approaches of software change impact analysis and assesses issues of these approaches. It also compares the proposed approaches of software change impact analysis in the literature. Lastly, it reviews how these issues are associated with software change requirement management approach by looking their weakness and strengths of the change management processes.

2. SOFTWARE CHANGE REQUIREMENT MANAGEMENT

Software change happens for different reasons, for example, in order to fix faults, to add new features, or to restructure the software to accommodate future changes [59]. Changing requirement is one of the most important motivations for software changes. Software requirement change from elicitation stage until the project has been rendered obsolete. Additionally, Changes to requirements reflect how the system must change in order to stay useful for its users and remain competitive on the market. At the same time, such changes pose a great risk as they may cause software deterioration. Thus, changes to requirements must be captured, managed, and controlled carefully to ensure the survival of the system from a technical point of view. Factors that can inflict changes to requirements during both initial developments as well as in software evolution are, according to Leffingwell and Widrig [60]: The problem that the system is supposed to solve changes, for example for economic, political or technological reasons. Or the users change their minds about what they want the system to do, as they understand their needs better. This can happen because the users initially were uncertain about what
they wanted, or because new users enter the picture. Problems arise if requirements and changes to requirements are not managed properly by the development organization [59].

3. EXISTING PRACTICES AND ASSESSMENT TOOLS

Performing software change impact analysis an important step when changing software requirement, specifically in incremental processes [53], it permits developers to decide the required work to implement the change request [55]. Several studies have been conducted in this area for the last 20 years. A lot of different processes and approaches have been published. All these processes and approaches are constituted with a set of roles, artifact and activities that are used to manage the software change process [2, 4]. However, most of the large enterprises today used tools and models for managing their software changes and requirement management [1]. These processes are designed to control all change requests and project development Lifecycle. However, some of these tools are software tools others are light weigh tools (pen & paper) [3]; Mostly, these processes focus more on capturing change request, managing requirements, requirement traceability and managing change requests. However, the following sections will be classified and assessed the current processes of change impact analysis.

In general, tool is a process that designed to achieve a specific purpose, especially if the item is not consumed in the process”. There are different tools, frameworks and models available in the market and can be classified into a certain number of categories in order to assess and identify their weaknesses and strengths. According to Vanita, he has discussed the most popular of impact analysis and how it supports the current practices. Many project developers used these tools to manage their software change requirements management. However, these tools are consisting of a variety of processes [27]. Some of these tools are commercial off-the-shelf software applications such as CELADON TOOL, SLATE (SDRC) and DOORS AND DOORSNET (TELELOGIC AB). Hence, the following table 1 review different tools of software change requirement management in the market today and assess their strengths and weaknesses to identify current issues of software change impact analysis in the field.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Strengths</th>
<th>Weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELADON TOOL</td>
<td>Celadon tool has implemented using eclipse environment context and used to assess the source code and divides their variation in a set of modification together with their dependence relationships. As a result, the assessed out game is presented as change impact parts and impacted tests [50]. It also consists of four main items: the atomic change generator, atomic change dependence detector, AspectJ call graph builder, and change impact analyzer. <em>With this tool when Atomic Change Generator evaluating two different programs, it always uses dynamic programing algorithm to assess the differences between two ASTs from there, it generates the corresponding atomic change sets [48].</em> <em>The tool supports more on source code change identifications.</em> <em>Normally, “change dependence detector uses the semantic dependence rules between atomic changes to determine the dependence relationships” [52].</em> <em>This tool takes change impact analyzer as an outputs of the change dependence detector and AspectJ call graph builder as inputs</em> <em>It can get to all impacts tests simply with it impacted changes and affected code fragments in the edited program version [52].</em></td>
<td>Celadon tool support only identification of source code change. It does not support the identification of the rest of the project artifacts [50]. It is difficult to evaluate the change impact dynamic point cuts. There is no process of change configuration management strategy. <em>It has less support to identify the impact of a change in estimation [48].</em> <em>The tool does not support change prioritization activities to identify the value of the change.</em> <em>The tool treats source code change identification in complex process.</em> <em>There are no requirement links to trace the change requests [49].</em> <em>The tool was not designed to support change impact analysis process for the whole software project.</em> <em>There is no option for determining the type of change.</em></td>
</tr>
</tbody>
</table>
| **SLATE (SDRC)** | SLATE (System Level Automation Tool for Engineers) is an application dedicated to systems engineering, combined software change and requirements [52]. This tool supports arranging folders and other objects of SLATE in hierarchal way. In this case it supports change impact analysis [50].  
• This tool supports tracing requirements through document-to-document.  
• Tracing requirement-to-requirement allows following up the source of the requirement to the complying item and creating output of the process instead using the main document which may be the heart of the process.  
• It supports change impact analysis using traceability [52].  
• Links are objects and are bi-directional.  
• Changes can be detected using MS-Word or Adobe Frame maker publishing system or in an ASCII text file.  
• The tool is built on the commercially available object-oriented client-server database  
• Attributes can be defined for all objects in the database when they are parsed/created  
• This allows requirements to be grouped into logical groups for easy categorizing, parsing, identifying, and analyzing the pool of requirements.  

| **DOORS AND DOORSNET (TELELOGIC AB)** | DOORS, is a software change requirement management tool that designed to use bidirectional traceability. It also allows changing impact analysis [51]. It comprises a complete change request process that allows baseline of the modules [52]. This tool supports: (1) detecting the artifacts by directly inserting them into DOORS, (2) controlling requirements in a traceable way [48].  
• The tool can be classified/categorize requirements during change identification  
• This tool supports change impact analysis. i.e., user can trace the change consequences to any piece of data from the rest of the system [49].  
• This tool support to create a relationship between modules in several project requirements or generating traceability reports that indicate how one project impacts another.  
• It supports specifying changes in traceable approach.  
• DOORS supports multiple users and groups, following a system of access levels and permissions similar to Unix users and groups  
• The DOORS interface is much like common file browsers such as Windows Explorer. DOORS allow users to create folders to store reports.  

| **IT's not compatible with Object-oriented database and give independent data such as deductive object-oriented databases. Hence, change configuration management strategy will be needed.** |  
• It is difficult to specify features and functions being added  
• The tool supports the only traceability approach and misses other approaches of impact analysis[52].  
• The tool does not provide Baseline requirements for the change  
• It also misses the option of storing the activities and observations  
• There is no process of System release and integration  

This tool Provides only a single-database repository [51]. As DOORS tool user views, the tool configuration management does not support with high project requirements churn i.e. If for example, 70% of the project requirements in a database change in a short period of time [52]. The tool needs to train a proper setup of the DB with respect to the attributes and traceability is good for suitable saving of project requirement.  
• It performs very limit in traceability directions (backward and forward)  
• This tool can only identify changes with goal attribute called No. of modifications.  
• If the object has been modified without change request process, this script will not recognize as a change to the object [52].  
• The tool can only calculate elements that in the proposed change which have already applied  
• The tool does not consider the change until it's approved or implement.  
• The tool does not support multi change requests and does not follow a change impact analysis process to estimation accurately.
However, software Change Requirements is the basis for which the tool needs to pre-plan. The high rate of software failure shows that there is a lack of proper approach to change requirement management. This may cause fixing a lot of change requests. In order to manage software change requests, it's important to have proper requirement management process from the beginning and pre-plan the strategy of software changes. Further, evaluating the requirement metrics in a project development is behaviour of a good project requirement management. The various requirements management processes and change management tools which available in the market causes the work to be complicated or increase the number of projects to be a failure. Consequently, these tools are not essentially providing the metrics for change and requirements management process. On the other hand, the following sections will be assessed and specified the strengths and weaknesses of models and frameworks of the current process.

4. ASSESSMENT OF EXISTING MODELS

In this context, process model, defines what are we going to be done (activities), who will be accountable (role), and what it should be the input and output [27, 32]. These are known as the items or elements of the model [25, 27]. These elements are the constructs of all models, which we have integrated a set of framework [39]. Mostly, different researchers have provided several items and there is no consensus for their items [25, 39] but when we look such as activities, roles and artifacts which are mostly discussed in the literature. Based on that, we are synthesizing from the literature define role as a set of responsibilities to be assigned to an actor. It provides a framework for actors to carry out tasks [25, 26, 27, 29, 30, and 31]. Activity is defined as action performed during the process and has clearly defined objective, entry, and exit conditions e.g. writing code for a module or writing a test case etc. [25, 26, 29, 30, and 30]. Artefacts /Deliverable is defined as product created, used, or modified during a process [25, 26, 29, 30, 30]. However, in these sections we have reviewed some concepts presented in the software change requirement management models in the literature [33, 34, 32, 28, 36, 37]. In the following table 2 shows activities, artifacts and roles/actors.

In Leffingwell and Widrig model [30] ignored the main part of the process which is change implementation. More specially, there are no verification activities shown in this model, thus, it is difficult to understand the proper work of change implementation. It also difficult to know the completion of the process and all activities were not documented in any form, e.g, it is difficult to refer, if someone wants to use the process of change decision next time, there is no detail information in this situation. Likewise, there is no process of change request and who has requested the change and when. Besides that, CHAM [35], is only used to estimate impacts on the resource and assesses whether change detail is enough or its impact on the effort. Besides that, cost benefit analysis can also help in this context, but no testing activity e.g. Acceptance testing or regression testing is not mentioned. Usually, when implementing change, regression test is important to assess whether the change is properly accommodated or not.

Furthermore, documenting activity is also missing. Therefore, this model is not supported current process of software change impact analysis. On the other hand, S.A Ajila model [36] is used to identify the consequence of the change on the functionality, but how can that change be finished without assessing the effect of change on requirement, cost and effort. Discussion being another key activity as whenever requested a change and its decision cannot be taken in isolation, collaboration among the concern of the project team will be required. No artifacts and actors discussed in this model which adds value to the completeness of the model [7]. Simon lock model [37] is missing change initial stage; there is no process of understanding the change request. Based on that, the change request will be stopped at the initial stage of the process that helps in saving the software cost [10].

5. INTEGRATION OF MODELS AND TOOLS’ ISSUES

This classification was created after assessing the existing work which aimed to identify the issues of a current process of software change impact analysis. We have reviewed some existing models and tools of software change impact analysis to compare in order to distinguish their problems to understand what is the main cause of the problem? On the other hand, during our specification and identification of existing processes issues, we identified the main issues of the process which
consist of two important processes, such as, Models and tools issues and practitioner’s perspective. These three issues were classified and addressed in separate.

In addition to that, the study recognizes that, most of the existing processes are not supporting current software change requirement management. For example, most of the tools used manual to specify impact of the requested change while other processes same like software requirement management do systematically. There is no help for indirect correlation in all tools excluding Top Team Analyst, and in RequisitePro and Caliber only links which are in a same path are used, so a manual check for effected requirement is still needed. The biggest problem with this tool is for example, the DOORS Change request process, permits a dedicated group to do the analysis and keeping the overview. It does not use to support link during analysis yet, but using the script language and it can be made more useful. In this tool, software change request process is an exceptional feature that adds value for large software projects. However, the identified issues consist of three several important perspectives such, Model, Tools and Practitioners perspectives. Each and every issue has its own load in the process, thus, the summary of the issues are:

• Currently, Impact analysis checklist process is not perfectly supported current software projects
• Most of the Solutions are specified with too many details by high-level analysts to perform accurate decision.
• Usually, Analysis are performed by the wrong persons
• In the tools, there is no option for determining the type of change.
• In the current practices, Change request decisions are based on interest
• Different change request have different levels of complexity, and there is no strategy for appropriate decision.
• Responsibilities and project balance are difficult to handle for analyses that span several systems.

On the other hand, the second reviewed approach of a current process are models, these processes were designed to help software change management. In this study, we have conducted a review of software change impact analysis models; this review reveals that there are a lot of distinctions among their descriptions. We didn’t find any similar pattern within these models; Most of the models which we have analyzed above are lack of deep detail of the approaches. There was no clear or defined relationship between these processes (roles, artifacts and activities). Therefore current change impact analysis processes are not supported to the project stakeholder's. This situation lets researchers by area to verify in detailed and improve like Software Development process models.

6. CONCLUSION

This paper summarizes our review. It shows that software change requirements management tools are more famous and used instead of purely traceability tools and limited process. Each tool has some strong and weak points. Others have limited support. But none of the processes are supporting change impact analysis process effective, so, there is a need to do more research in this direction, we have also presented current practice issues of Impact Analysis from two important sources in the literature and practitioners. This review confirms issues of current change impact analysis process and reveals that there are a lot of weaknesses and differences between the descriptions. All processes are lack of detail in their approaches. There is no clear correlation between activities, artifacts, and roles/actors. However, current processes of change impact analysis have less support to the software change requirement management. This situation calls for a change impact analysis process improvement in a lightweight manner and developing a detailed approach that can support practitioners to use for their software change processes.

REFERENCES


[34] M. Makarainen, Software change management process in the development of embedded software, Dissertation, VTT Technical Research Center of Finland, ESPOO, 2000

[35] B.J.M. Abma, Evaluation of requirements management tools with support for traceability-based change impact analysis, Software Engineering Faculty of Electrical Engineering, Mathematics and Computer Science University of Twente, 2009


[43] Moore, Gary C., and Izak Benbasat. "Development of an instrument to measure the perceptions of adopting an information technology innovation." Information systems research, 192-222. 1991


