Abstract—Requirement change management is very critical and the most important aspect in the software development. Changings unleash themselves during all the stages of software development. Hence we must need to have some effective techniques to tackle these changings. Basically change is a transition from current way of working to another looked-for way and this nature of the change coupled with complexity of services create problems. This paper depicts the main hurdles in the change management like dependability, traceability etc. and efficient tools to cope these changes so that it would not affect the stability. Ultimate aim is to propose a framework to manage this important trait in the process of software development.

Index Terms—Framework, changing requirements, requirements management.

I. INTRODUCTION

It is frequently reported that requirements often change in the software development process. Several factors cause these changes, i.e. inaccuracies in original requirements, evolving & changing customer needs, technological changes, and changes in organizational policy or business environment [1]-[5].

Volatile requirements are observed as an element that causes major complications for most organizations during system development [3]. It is also considered as one of the most significant cost drivers [6]. Moreover, simulation models of software development projects depict that requirements volatility has a major impact on project duration and development effort as well [7], [8].

The Change in requirements is expected during development as well as software maintenance. Because the environment and stakeholders want constant change, similarly as the system, with the intention to continue achieving its projected purpose. Studies depict, the change in requirements is responsible for a huge part of rework in phase of development, even up to 85% for some cases. [4], [9], [10]. Hence, it becomes one of the major reasons for a project’s failure [7]. It is often considered as one of the most prolonged problems in software development [11].

If changes are occurring in a project, i.e. external changes, internal changes, and technical changes learning then those changes surely going to have a severe impact on people, processes and overall project. To help reduce the impact of having unintentional negative outcomes, it is essential to use, change management methodologies in place along with skilled resources for executing these methodologies. This helps to reduce possible negative outcomes and increase probability of positive results.

Current methods of requirements engineering consider anticipation of a requirement change as a detached activity afterward documentation.

II. LITERATURE REVIEW

Bhatti et al. [12] suggested the method of change comprises of 6 elementary steps as presented in Fig. 1. In very 1st step, change is primarily asked. The 2nd step is known as receiving step that includes consideration of particular request. The request of change is acknowledged by a particular request form for req. change. When the request is received, the next step includesevaluation and assessment ofreq. Change request through a control board for change (CCB), to see effect of change on various components of the whole system. The 4th step includes decision concerning approval/rejection of request that is taken mutually by Control board, QA team, and members of project & Request initiators. This decision is constructed on the basis of probability of requirement. Change request. This whole approved req. change request is then forwarded to the phase of implementation, where these req. accepted changes are assimilated into mainsystem. Now final stage is known as configuration stagewhere a log is maintained for entire change requests.

Fig. 1. Bhatti et al. change management process [12].
This process model includes the activities of req. change management in SDLC. However, this model lacks some significant activities, i.e. verification or batch activity. Because if absence of verification activity, it is hard to define that changes that has been done in the system are functioning well.

In the same way with the lack of batch activity, this model can’t noticeably signify thereq. Changes that can’t be fulfilled instantaneously, but in the future can be implemented [13].

A spiral shaped model for the management req. Change is shown in Fig. 2. That comprises 4 cycles or steps. In very 1st round; few alterations are asked as adding new features or fixes of bugs in the existing system.

The model proposed by Ince [14] starts with the stakeholders’ change request as given in Fig. 3. Furthermore the req. Change request is then documented in change request note & afterward sent to the Control Board. The CCB takes decisions on specific req. change request. After viewing this req. change request, the CCB may take 3 kinds of decisions as mentioned below:
1) Rejection of the requested req. change (For any reason, not possible to implement change).
2) Queued the change request for some time (After some period of time, it would be convenient to make this change).
3) Accept particular req. change & immediately implement.

In case if req. changes request is approved for instant implementation, the req. change permission note will also be filled. After performing this step, next step would be the implementation of that req. change. Once implemented, the documents of the system are then updated. After the implementation of req. change is done, it is validated in respective phase. Later on, the test records of problem owners choose if change is compulsory to be completed and if it’s essential, then how it’s going to be incorporated. The 2nd step of this model starts when anticipated change is required to be viewed and evaluate from non-technical perspective. This cycle’s third step includes planning where a plan is built for implementing req. change. Implementation is in fact the final stage of req. change request. If we view this step, in this specific requested req. change is verified and implemented as well in this step. After the req. change is implemented, entire updates regarding that change are maintained in a log. This proposed model [14] is in fact a standard one; its utilized in managing the changing requirements, both in the phases of development as well as maintenance. But, this model is not able to deliver any sort of information about verification of particular req. change or batch activity [13]. Implemented req. changes are created and configured records are altered. Stakeholders are then informed about particular req. change that is implemented. The model proposed by Ince [14] defines the change management procedure in requirements; but this model doesn’t have any verification action [13]. Hence, it’s hard to determine about a particular req. change, completed in the system, is operating accurately.

Like other change management model, a model proposed by Olsen [15] is appropriate for software development phases. The user who requested for the change provides all the sources which require accomplishing change. The changes requested by users are managed by the change manager in change management phase. Once the changes have been accepted, they are sent to the implementation stage. Requested changes are implemented at implementation stage. With the help of testing changes can be verified. Once changes have been verified, change managers are informed to release new changes in the software, shown in Fig. 4. The change is highlighted as a basic element of life cycle of a software development, in change model [15]. On the other hand activities which are required to manage change are not considered in this model.

Any information about the requested change like who made the request for change, when the request has been made and how, is not provided in this model. Also, this model completely ignores the activities which are part of the change request. Because of completely ignored batch activities this model does not provide any idea about the changes which are
deferred to subsequent releases [13].

![Olsen’s change management model](image)

III. OBJECTIVE OF STUDY

The main purpose of the research is to propose a framework for the change which comes with requirements and to manage that change. This framework will show how to handle different changing requirements which come in different phases of software development. This framework will help in reducing risk of failure by managing changing requirements.

IV. METHODOLOGY

Different literature on RCMM (requirement change management models) [3], [14], [15], books on requirement engineering [15] and case studies [8] has been studied and all information presented here is based on these studies. Majority of the studied literature is based on actual life experiences of implementing the requirement change management.

Data which is related to the requirement change management has been extracted from the studied literature at data extraction phase. Systematic literature review (SLR) as recommended by an evidence-based software engineering paradigm [16] has been carried out during the review process. SLR can be divided into three main phases which are described by kitchenham et al. [16] as follows:

1) Review planning
2) Carry out the review
3) Review reports

In planning review phase a include and exclude criteria is defined for the literature to be reviewed. Where in carrying out the review, related data is extracted from the literature and in review reporting the results obtained from literature review are described.

V. PROPOSED SOLUTION

In this paper, a framework has been proposed which has seven stages. This is a specific set of activities which is performed at each stage while the requirement is changed. The first stage will deal with the change request. It is same as discussed in Ince’s Model [14]. The change request may come from different stakeholders, this stage includes all the information about the change which includes what kind of change is required, why the change has been made, and who made the change request.

The second stage deals with the validation of change request. This stage is divided into two parts, one part is related to the understanding of the change and validating that change via numbers of checks. And what can be the effects of that change on the other system components, schedule and resources. After understanding the requirement, it is decided either to implement the change or not. This stage is adopted from Ince’s Model [14] and spiral resembling model for change management.

The third stage is rejecting stage if a requirement conflicts with any other requirement or cannot be implemented due to some constrains or not valid then it is rejected and move to the rejection stage. And if a change request is deferred to later releases to software then it is moved into batched requests. This phase is also discussed in Ince’s Model [14].

Fifth stage is implementation in which all accepted changes are implemented. And after implementation of the changes the verification of the changes is performed. The verification stage is similar to change management model by Olsen [15]. After the verification of all requirements that the system is working properly next move to validate stage. In which again validation is performed and at the end all the changes are updated. And the new updated version will be released.

VI. SOME COMMON MISTAKES

During the life cycle of the software development requirements change from the beginning till the end of the system development. Change in requirements is inevitable. Even though most of changes in requirements are handled in the beginning and they tend to continue until final implementation. The changes in software requirements affect the system to be developed as well as the enterprise, stakeholders and the project resources and schedule. Whenever there, the changes in requirements occur, it might also affect other system components, it is important to trace and analyzed the change and it's effects on other components. It is necessary to ensure that change in one component doesn’t affect functionality of other components.

The main objective of change requirement management is:

- To identify the main reason of change in requirement.
- To identify the major stakeholders which are involved in change.
- To trace each requirement, i.e. from the original requirement of the changed.
- To identify the impact of changing in requirement on the overall system.
- To agree all the team members on a change.
- To record the change in requirements.

Requirements are managed to provide the reliability in terms of consistency and to avoid uncertainty within the organization. There should be a specific predefined process for controlling the costs and to prioritizing the requirement
and also a standard method to analyze each requirement. These standards and predefined processes are part of project management activities and necessary for effective requirement engineering.

There are some activities which should be part of requirement change process:

- Provide sufficient resources.
- Perform analysis when any requirement changes.
- First document the requirements and then document the change.
- The requirement should be documented in such manners so it is possible to trace each requirement from its origin till the end of the project.
- Avoid communication gap.

Document the baseline project requirements.

VII. FUTURE WORK

As the Requirement change is of most importance. Most of the project fails due to the inefficient change management. Although the studies have been conducted, but still a lot has to be done to handle these changes effectively, the area of change management lacks research. There is a still lack of some efficient enough and effective framework, to handle these consecutive changing requirements during all phases of SDLC. Therefore, we strongly recommend to study further effective techniques to handle changes and identifying and evaluating the causes of requirements change and their relationship with each other to propose an efficient enough framework to handle requirement changes, as they are considered to be an impact factor for the success or failure of software projects.

REFERENCES


Syed Mohtashim Abbas Bukhari completed his BS in computer science from FAST University, Pakistan in 2012 and he is now doing M.Sc in software engineering from National University of Science and Technology (NUST) Pakistan. Since 2012, he has been working at En Medical Systems as a software engineer. His research interests include software requirement engineering, embedded software engineering and software process models.

Muhammad Abbas is a professor at the Department of Computer Engineering, National University of Science and Technology (NUST) Pakistan, Specialization in ERP systems. He obtained his M.Sc and Ph.D. degrees in information systems from University of Manchester, UK. His research interests include ERP systems, project management, information systems.