Identify and Manage the Software Requirements Volatility

Proposed Framework and CaseStudy

Khloud Abd Elwahab
MSc student /Information system Department
Faculty of computers and information, Helwan University line 3: Cairo, Egypt

Mahmoud Abd EL Latif
Associate Professor of information system department
Faculty of computers and information, Helwan University
Cairo, Egypt

Sherif Kholeif
Assistant Professor of information system
Faculty of computers and information, Helwan University
Cairo, Egypt

***Abstract*—Management of software requirements volatility through development of life cycle is a very important stage. It helps the team to control significant impact all over the project (cost, time and effort), and also it keeps the project on track, to finally satisfy the user which is the main success criteria for the software project.**

In this research paper, we have analysed the root causes of requirements volatility through a proposed framework presenting the requirements volatility causes and how to manage requirements volatility during the software development life cycle.

Our proposed framework identifies requirement error types, causes of requirements volatility and how to manage these volatilities to know the necessary changes and take the right decision according to volatility measurements (priorities, status and working hours). This framework contains four major phases (Elicitation and Analysis phase, Specification Validation phase, Requirements Volatility Causes phase and Changes Management phase). We will explain each phase in detail.

Keywords—software requirements; requirement errors; requirements volatility; reason for requirement changes and control changes

I. INTRODUCTION

The software engineering industry faces several issues; requirement changes are one of the most significant and critical issues during software development process. The project requirements are almost never stable and fixed as has been explained by [Jones (1996)]. Requirements are defined in [1] as "The information from the user about what will do and what is the main objective of this project and what is the deadline to deliver this project and so on".

Accuracy and focus through gathering requirements does not prevent requirement changes to take place during the software development life cycle. Requirements volatility is defined in [18] as "the emergence of new requirements or modification or removal of existing requirements". Numbers of requirements change during software development process depending on the quality measures of requirements (Correct, Unambiguous, Complete, Consistent, Importance and Stability, Verifiable, Modifiable, Traceable, and Understandable) [5].

Requirements changes have a significant impact on project performance, project schedule and budget. This paper proposes a framework focus on how to manage requirements volatility during the software development life cycle and to limit the implications thereof. The remainder paper is structured as follows: **Section 2** presents the motivation to search for the requirements volatility topic. **Section 3** explains the requirements volatility definition, factors, causes, and the measures, and finally explains the impact of that on software development process. **Section 4** contains proposed framework and presents a case study in explanation of the benefits of this framework**. Section 5** concludes our results and provides some open research directions

II. MOTIVATION

Understanding the requirements volatility and the impact thereof during the software development life cycle is a very interesting area that needs more studies to focus on how to manage these changes. Abeer. Al and Azeddine, in [18] according to previous studies suggest that 86% of the change requests are related to requirements volatility and it is often more than 50% of the requirements are changed before the delivery of a software project, furthermore implementing the requirements volatility in later phases causes 200 times costlier than implementing the requirements volatility in the analysis phase. All these facts are the motivation to start searching and try to find solutions for such critical point.

III. RESEARCH QUESTIONS

In this paper, we are going to discuss the following questions:

1) *What is Requirements volatility?*
2) *What are the causes of Requirements volatility?*
3) *What is the impact of Requirements volatility?*
4) *How to manage requirements volatility in different phases using proposed framework.*

IV. REQUIREMENTS VOLATILITY

Requirements volatility refers to additions, deletions and modifications of requirements during the system development life cycle, as defined in [4] "Stable requirements are the holy grail of software development." RV creates additional work in design and coding, which increases the system development cost and time and compromises the system quality. Ignoring

requests for requirement changes can cause project failure due to user rejection, and failure to manage RV can increase the development time and cost.

A. REQUIREMENTS volatility factors

Environment changes are the main factor for requirements volatility; according to previous studies, there are Development Environment changes, and we also cannot avoid the Business Environment changes [11].

– **Business Environment changes** are, for instance, government regulations, market competition, financing sources, restrictions, management changes, organization policy, legal factors, technological factors and business laws.

– **Development Environment changes** are, for instance, requirement errors, evolving user and technological needs, new technology, missing members of team project, incorporation of cost upgrades; resolve requirement conflicts, missing requirements).

Fig.1 shows the various factors causing requirements volatility [11].

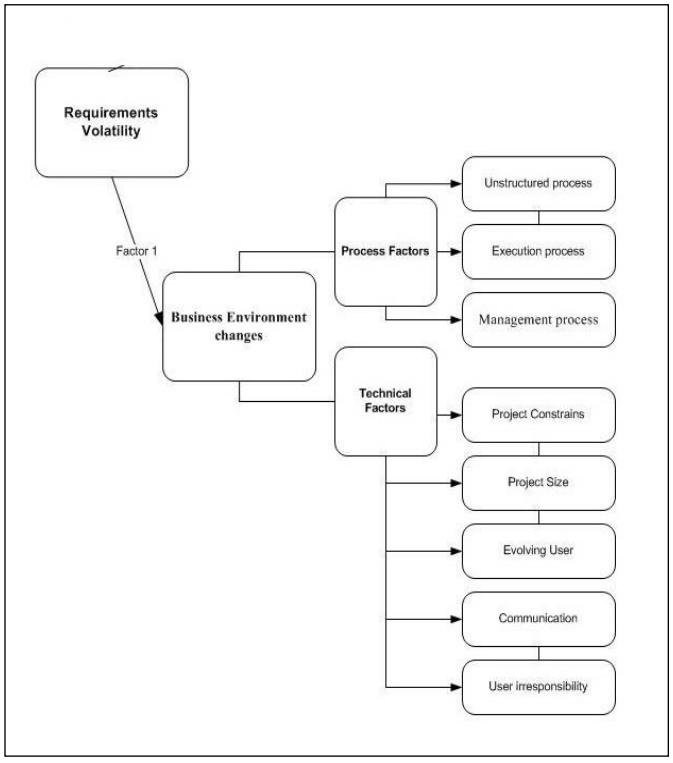

Fig. 1. (a)Requirements volatility factors [11]

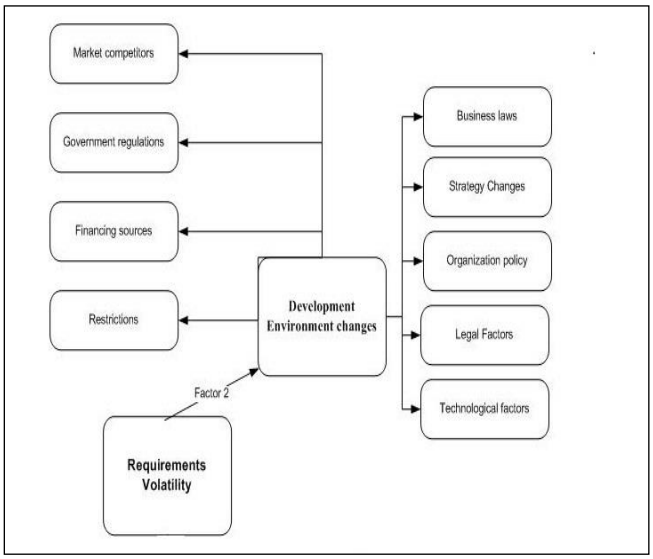

Fig. 1. (b)Requirements volatility factors [11]

B. Cause of REQUIREMENTS volatility

After considering the environment changes, we concluded that there are several causes for requirements volatility, Requirements errors are one of the main causes. For more accuracy, we classify requirement errors into three main groups: people errors, process errors and documentation errors [7], Fig. 2.

- **People Requirement Errors:**

– The communication gaps between stakeholders.

– Poor participation between the development team and management team.

– Less understanding domain knowledge, unstructured process execution.

– Users have unreasonable timelines and do not really know what they want

- **Process Requirement Errors:**

– Bad management process.

– Rush and bad analysis requirements, using old process and methodology.

– Inadequate method of achieving objectives and requirements change during the project

- **Documentation Requirement Errors:**

Team may do not understand the policy of the user's organization or no use of standards.

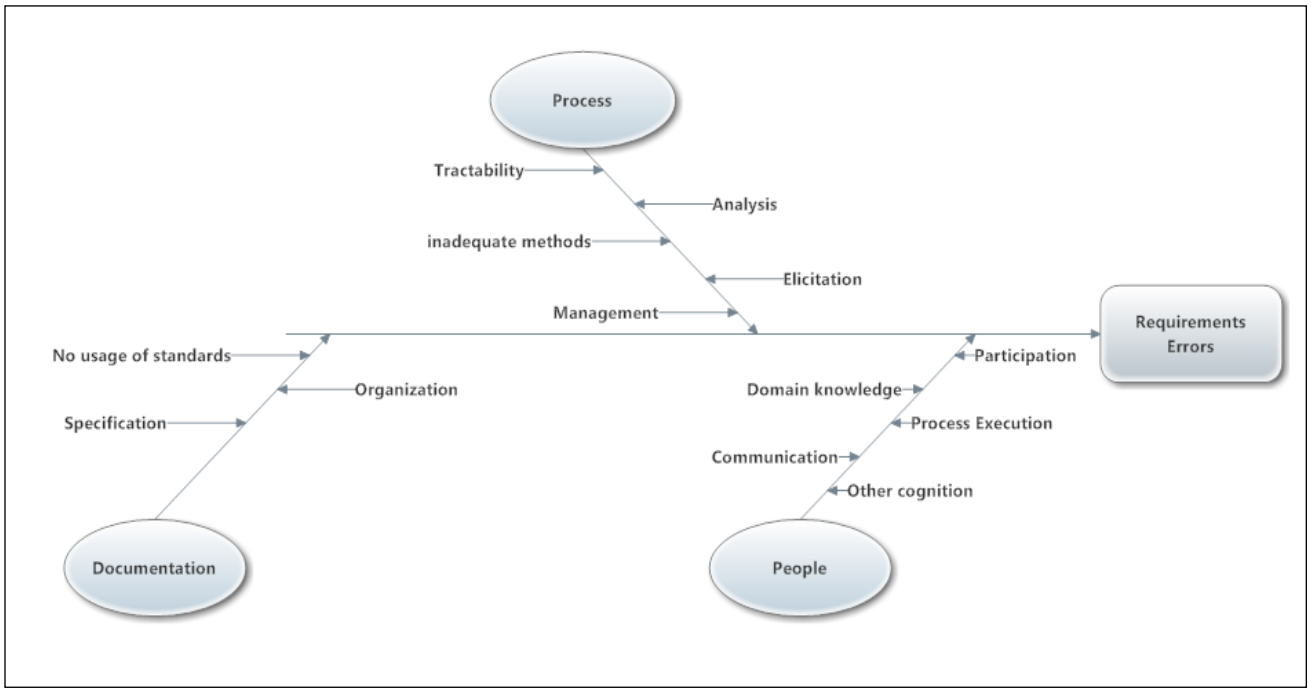

Fig. 2. Types of Requirements Errors [7]

C. Mesurements of REQUIREMENTS volatility

To avoid implementing unnecessary volatility changes, we should analyse the changes request depending on some measures; for instance, priority of changes (low, high, and medium) and the severity of changes (Critical, major, minor….), working hours needed, change types (added, deleted or modified), phase in SDLC and impact of each change. Figure-3

D. Impact of REQUIREMENTS volatility

Several research studies have found that requirements volatility is positively correlated with the increase in the size of the project, effort and cost (often >20%) and schedule duration [19].

Requirements volatility inevitably result in additional work and increased defect density, furthermore, it increased development working efforts that need a rework in code, design, and also increase team working hours and cost. Sometimes we need to reschedule the whole project, as the volatile requirements quality decreases. We need to re-design the test cases according to new requirements.

Fig.-4 presents the requirements volatilely over all (measurements, cause, and impact).

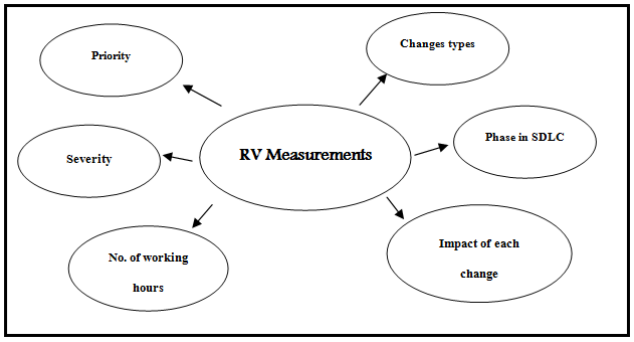

Fig. 3. Measurements of Requirements volatility

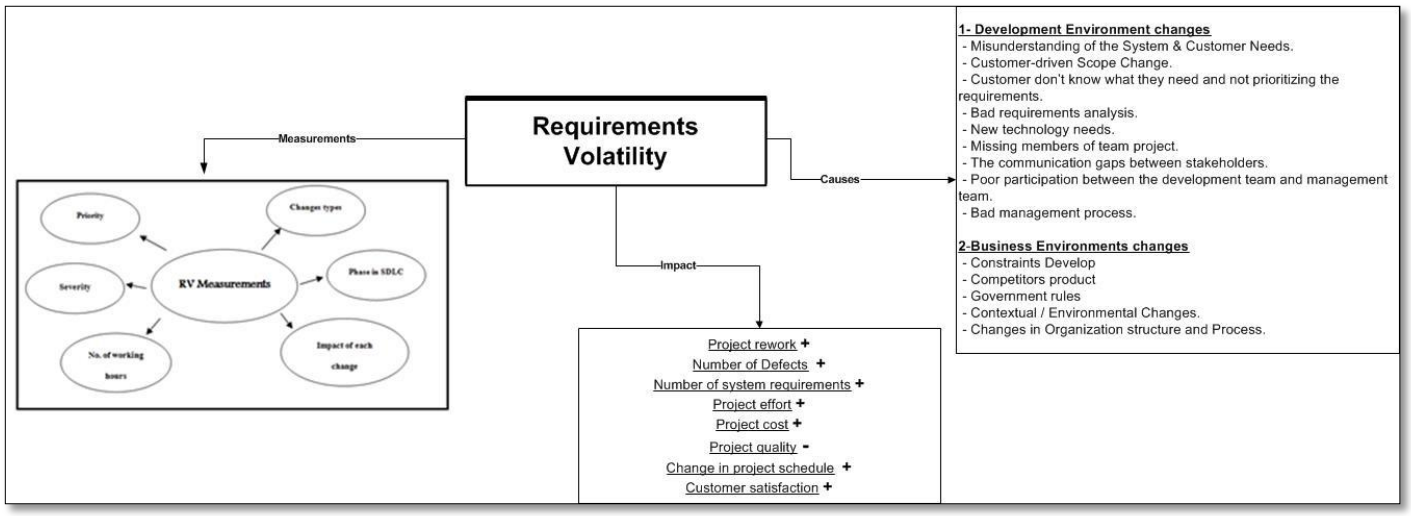

Fig. 4. Requirements volatility Causes, Impact and Measurements

V. SOFTWARE REQUIREMENTS VOLATILITY PORPOSED FRAMEWORK

After studying the causes and impact of volatility in requirements, we have proposed a framework using UML concepts for model-driven software development, which is given in Fig. 5. This framework consists of four major stages.

1) *Elicitation and analysis of business*
2) *Specification validation*
3) *Requirements volatility causes*
4) *Manage Changes phase*

The proposed framework identifies requirement error types, causes of requirements volatility and how to manage these volatilities to know the necessary changes and take the right decision according to volatility measurements (priorities, status and working hours).

The proposed framework is used to help the team to know the root cause of requirements volatility that will reduce number of changes happened in next release and other project.

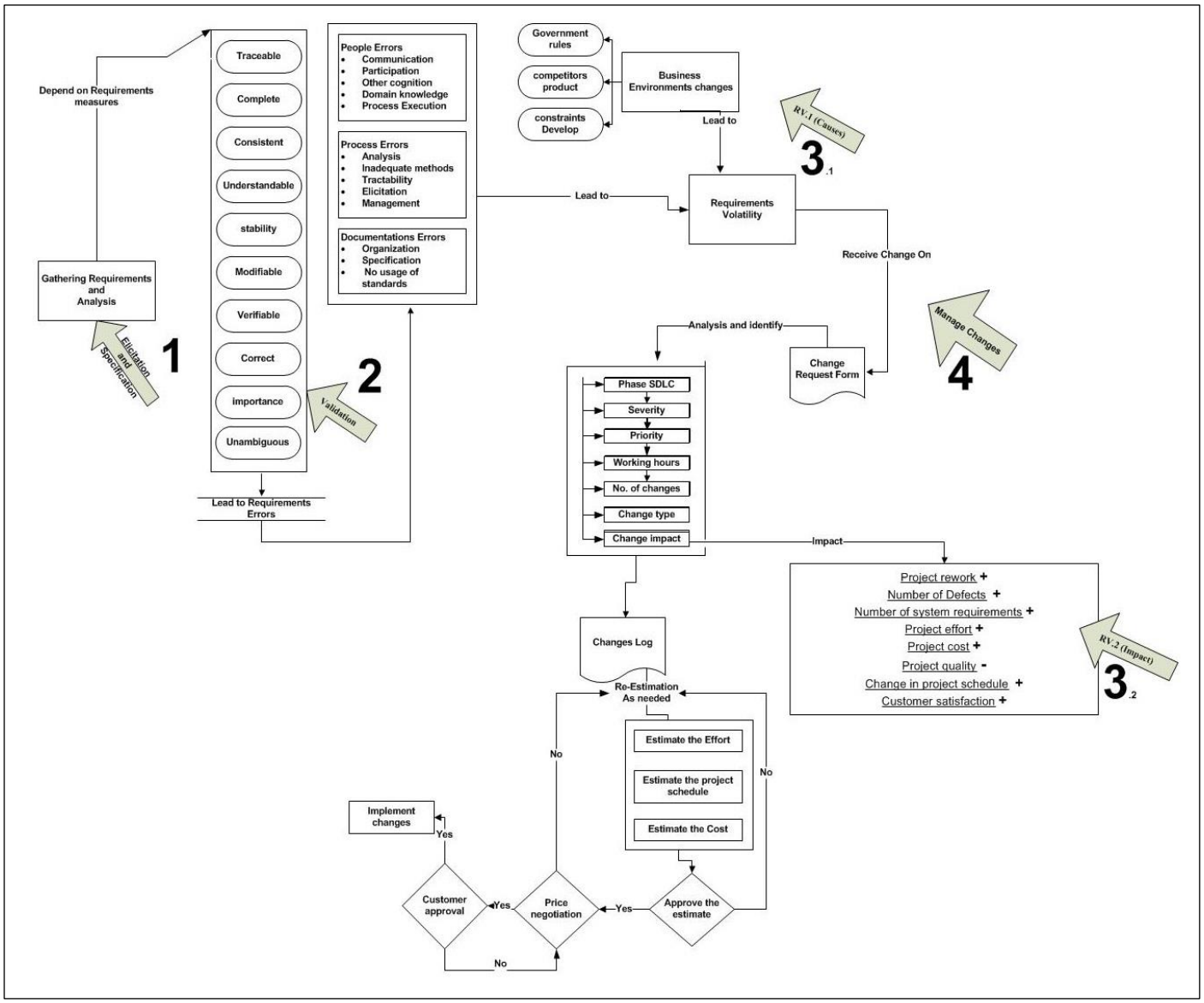

Fig. 5. Proposed framework to identify and manage the software requirements volatility

B. *Elicitation and Analysis of business REQUIREMENTS:*

Elicitation and analysis requirements is a critical phase in the software development life cycle, in this phase, requirements should discussed with user to make clear and then enter agreement therewith for all requirement needs. This process is to ensure that requirements are visible to and understood by all stakeholders.

There are some techniques that help in getting more accurate requirements such as: Brainstorming, Document Analysis, Focus Groups, Requirement Workshops and Interface. The results will be documented in Software Requirements Specification SRS, The SRS is full analysis and formalizing the requirements definition, what the software will do and how it will be expected to implement [16] [17]

C. *SPECIFICATION VALIDATION:*

Specification validation works with the final requirements document where a group of work team read and validate the requirements according to nine requirement measures "Correct, Unambiguous, Complete, Consistent, Importance and Stability, Verifiable, Modifiable, Traceable, and Understandable"[5] and user needs, and then look for errors to discuss and agree to actions to address such errors before beginning implementation to avoid volatility during software development life cycle (SDLC).

Fig. 6 shows the nine requirement measures and the classifications of requirements errors that will appear due to poor requirements gathered

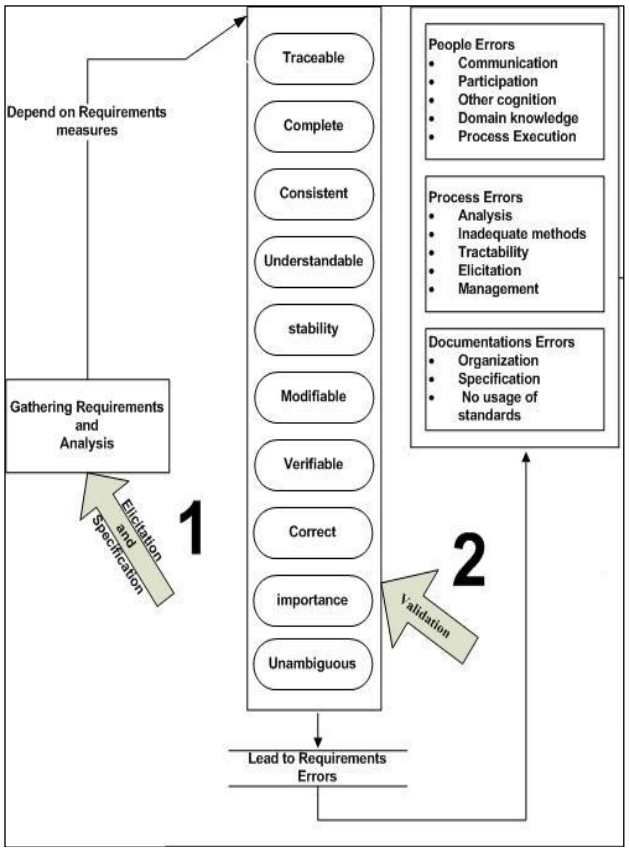

Fig. 6. Elicitation and analysis & Validation phase

D. Requirements VOLATILITY:

It is impossible to find software project without requirements volatility. We can define requirements volatility as missing requirements or misunderstanding and also gathering requirements without consideration of the nine requirement measurements to be added, deleted or modified, to keep project on track.

The development team needs to deal with changes and handle them. If the changes are not handled effectively, problems can occur needing extra efforts to manage the impact thereof on cost, quality, and schedule, as can be seen in the impact of requirements volatilely framework, both of **Project rework + Number of defects + Number of system requirements + Project effort + Project cost + and need to Change in project schedule.** At the end, all these changes may conclude into Customer satisfaction.

The causes of volatility can be classified into two main categories: development environment changes and business environment changes. These have already been discussed before.

Fig. 7 present causes of volatility and we are going to describe how to control the changes in next phase.

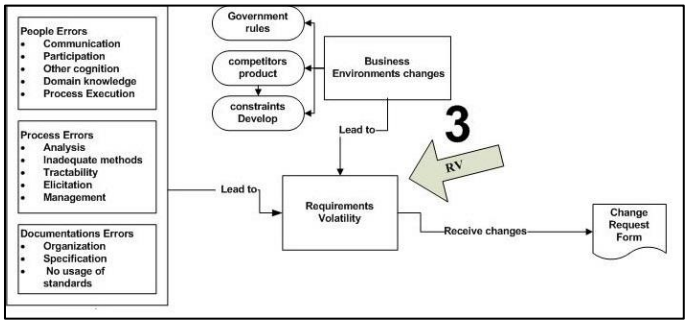

Fig. 7. RV Causes

E. Changes Management phase

It is almost certain that projects face changes during project life cycle development as explained above. These changes may help projects to cope with business needs. Change management is an important part of the project management process, thus each change should be considered carefully before approval, in order to deliver a project successfully.

The Proposed framework illustrates change control process, that each change requires a form properly defined by the user, that includes details of the change, and the business case, then the development team analysis may be considered and the changes approved prior to implementation to avoid unnecessary changes, and not to disrupt resources and delay the project delivery depending on seven measures (change impact, severity, priority, working hours, phase SDLC, change type and the number of changes), using all collected data to record all changes requested and decisions made in change log, the task group would re-estimate the project plan (effort, project schedule and project cost), developers are responsible for estimating the effort required to implement the new requirement changes which they will work on , project managers notify user and negotiate the cost of changes going to be implemented, after the change is done correctly, the case is closed in the change log.

Fig. 8 present the controlling of requirements volatilely process

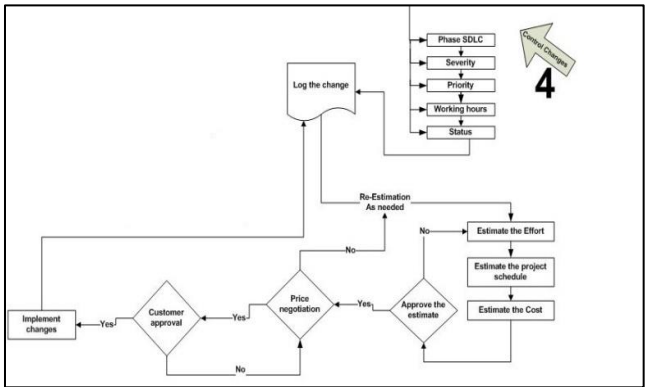

Fig. 8. Ccontrolling process of requirements volatilely

VI. Case Study

In this section, we will apply the proposed framework on ADJD system as a case study and discuss the impact of software requirement changes during the life cycle development in different project aspects (cost, schedule, and quality).

ADJD system refers to the Abu Dhabi Judiciary Department-Treasury System; the treasury system developed for the Abu Dhabi Judicial Department integrates with the existing case management system to facilitate the management of financial deposits and withdrawals associated with legal cases. By recording details of the beneficiaries and building an accounting structure, the Judicial Department is able to organize the financial processes involving receivables, check issuing and bank account management.

Basically clients are not technical people. They need software for their business but the requirements are unclear. Nobody has done an in-depth analysis of all the implications so during implantation phase client request some new features. The new features will probably break some assumptions development team made in their code and they start thinking immediately of all the things you might have to refractor, no matter how conscious you are of all these new features, you give shorter times than you originally suppose it might take. Special when you feel the pressure of deadlines and management expectations.

Measuring the Requirements volatility percent: Stark [20], derived a formula based on the statistics on different projects that:

Requirements volatility = (added + deleted + changed (modified))/ (# requirements in VCN)*100 [20].

Where VCN (version content notice) = set of requirements agreed by both the developer and customer.

In VCN 1.2 release has 55 requirements initially, later 2 new requirement are added, and 5 requirements are deleted from initial requirements, modified 11 requirements and at the end add 2 new features as business needs.

RV = (2+5+11+2)*100/55= 36.4 % of project has changes

TABLE I. Log of Changes

RV No.	Priority	Severity	Change Type	Effort *6 H (man days)
2	2	Critical	Additions	3
11	3	Major	Modifications	2
5	5	Minor	Delete	2
2	1	Critical	Add new feature	5
Total= 20				Total=12 man day * 6

- First when the proposed framework is used, it helped the team to know the root cause of requirements volatility that will reduce number of changes happened in next release and other projects.
- As is explained in case study, 36% of project requirements changed due to requirement errors that have not been discovered before development team start implementation. Proposed framework manage changes appeared during life cycle development by analysis of changes requested depending on some criteria (priority, severity, type of change, change type, impact of change and no of working hour's needs) to avoid implement unnecessary volatility change.

VII. Conclusion

In this paper we have described several aspects of requirements volatility, such as factors, causes, measurements and impact of requirement volatility on software life cycle development, and also proposed framework to manage requirements volatility.

The causes of requirements volatility cannot be overcome fully but we described some causes like (poor communication between stake holders and developers, technical aspects and bad management process, etc.)

Requirements volatility has an impact on the whole software life cycle development. It has impact on the project schedule, cost, and quality. It cannot be avoided, but we can manage it to reduce the effect of requirements volatility by following some methods in analysis, design, and coding. Due to the impact of requirements volatility many projects have failed.

The proposed framework will manage the requirement changes that help to reduce the effect of changes made all over the project. This framework contains four major phases (Elicitation and Analysis phase, specifications Validation phase, Requirements Volatility phase and Change management phase) that help project development team to know the necessary changes and take the right decision according to volatility measurements (priorities, status and working hours).

Future work: more research is needed to develop the flexible architecture which is suitable for requirements volatility. We need to define new methods to manage requirements volatility, and also more work on management of the requirements volatility, modified framework is needed by adding different software process models

References

[1] Amira A. Alshazly a, *. A. (2014). Detecting defects in software requirements specification. Alexandria Engineering Journal, 15

[2] Aybüke Aurum, C. W. (2005). Engineering and Managing Software Requirement. Germany: © Springer-Verlag Berlin Heidelberg.

[3] CS2 Software Engineering note 2. (2004, autumn 1). Software Requirements1. Software Requirements1, p. 9.

[4] Daniel D. Galorath, Galorath Incorporated. (2006). the 10 Step Software Estimation Process for Successful Software Planning, Measurement and Control. Galorath Incorporated, 13.

[5] Davis, A., Colorado Univ., C. S., Overmyer, S., Jordan, K., & Caruso, J. (1993). Identifying and measuring quality in a software requirements specification. Software Metrics Symposium, IEEE. Proceedings. First International, 12.

[6] Dhirendra Pandey1, U. S. (2011, May). A Framework for Modelling Software Requirements. IJCSI International Journal of Computer Science Issues, 8.

[7] Gursimran Singh Walia a, J. C. (2009). A systematic literature review to identify and classify software requirement errors. Information and Software Technology, 23.

[8] Kelly, A. (2008, Feb). Changing Software Development: Learning to Become Agile. Wiley.

[9] King, A. F. (2005, December). How to Detect Requirements Errors-A Guide to Slashing Software Costs and Shortening Development Time. Retrieved from ravenflow: www.ravenflow.com

[10] Loconsole, A. (2007). Definition and validation of requirements management measures. SE-90187 Umeå, Sweden, Umeå University, Thesis, 106.

[11] M.P.Singh, R. V. (2012, 9). Requirements Volatility in Software Development Process. International Journal of Soft Computing and Engineering (IJSCE), 6.

[12] N Nurmuliani, D. Z. (2007). Analysis of Requirements Volatility during Software Development Life Cycle. Australian Software Engineering Conference (ASWEC'04), 13.

[13] Sakthivel, S. (2010). Manage Requirements Volatility to Manage Risks in IS Development Projects. ISACA JOURNAL, 4.

[14] Sudhakar, M. (2005). Managing the Impact of Requirements Volatility, Master Thesis, Department of Computing Science,Umeå University,SE-90187 Umeå, Sweden, 42

[15] Awasthi, R. (2012). Development of a Structured Framework to Minimize Impact of Requirement Volatility. International Journal of Computer Applications (0975 – 8887), 7

[16] International Institute of Business Analysis (IIBA), "Business Analysis Body of Knowledge 2.0", 2009

[17] Wiegers, Karl E., "Software Requirements, 3nd Edition", Microsoft Press, 2009

[18] Abeer AlSanad and Azeddine Chikh., (2014). " The Impact of Software Requirement Change – A Review "

[19] G. Stark, P. Oman, A. Skillicorn, and R. Ameele, "An Examination of the Effects of Requirements Changes on Software Maintenance Releases" in Journal of Software Maintenance Research and Practice, Vol. 11, 1999, pp:293-309

[20] Mohd.Haleem, Mohd.Rizwan Beg, Sheikh Fahad Ahmadlnternational,Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, No 5, May 2013, pp: 1811-1815.