International Journal of Advance Research in Science and Engineering
Vol. No.4, Special Issue (01), September 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

SURVEY ON REQUIREMENTS COLLECTION PHASE TO ENHANCE QUALITY SOFTWARE FIRMS WITH RESEARCH METHODOLOGY

Mrs.Zahiba Khan[1], Dr.A.N.Nandakumar[2]

[1]*Assistant Professor, Department of Computer Science & Engineering,*
Ghousia College of Engineering, Ramnagara
[2]*Research Supervisor, VTU Belgaum*

I. INTRODUCTION

An effective requirements gathering process is perhaps the most critical driver of software project success. Getting the requirements right – and getting the right requirements – can mean the difference between a successful project – one that satisfies the needs of its users and is delivered on-time and on-budget – and one that fails.

It should come as no surprise that effective requirements gathering involves much more than asking business users what they want and need. It is a complex process that involves users and system designers in a collaborative effort that explores both functional requirements and the new possibilities that technology offers.

The great challenge of the requirements process is finding a way to uncover and capture the needs of the business and communicate those needs to a software development team in a language and style that facilitates the software design process, producing a result that precisely solves the business problem. This is much easier said than done. All too often the requirements process begins with a few key questions about the business need, and then quickly moves to discussions about parts of the technology solution. Traditional requirements documents are a confusing combination of business and technology language. As a result they are time consuming for the careful reader, misunderstood by most, and generally ineffective as communications devices.

Requirement elicitation is a critical activity in the requirement development process and it explores the requirements of stakeholders. The common challenges that analysts face during elicitation process are to ensure effective communication between analyst and the users. Mostly errors in the systems are due to poor communication between user and analyst.

Gathering requirements is the cornerstone to any successful project. Requirements are critical for scoping, defining, estimating, and managing the project. In projects employing Agile, achieving desired results requires a measured approach that may be unique to each organization. For organizations with little or no experience employing Agile methods, the risk is high for delays and overruns. In this comprehensive overview of requirements gathering and project delivery, we explore an approach and framework that can help drive success through open communication with stakeholders around an active project management framework designed to best fit the organization.

In a global economy, organizations are forced to adapt quickly to evolving technology and a fluid corporate landscape. Few of these companies have shown the capability to quickly pivot their software production or manufacturing units to consistently deliver end-user products that are of high quality and on schedule. The capabilities required of a high-performing organization to provide increasing development velocity with frequent and effective communication cannot be implemented overnight. It requires an organizational commitment to change and cultural adaptability for progress.

The trend across industries is toward the Agile methodology of development; it provides a greater flexibility within a development unit to confront a constantly changing market demand and competitive landscape. A company needs to maintain a high standard of work, focus on cost reduction, and promote alignment between business and information technology. The evidence demonstrates how this shift has continued to develop and, as The Economic Times wrote at the end of 2012:

"While less than one percent of IT departments had even heard about Agile in 2000, Gartner and Forrester statistics indicate that 60-80 percent of software development teams now use Agile as their primary method for creating software. This is higher than in 2009, when Forrester research said 45 percent of development teams used Agile methods to write code or create products."

While a greater number of corporations are aware of Agile and are employing its concepts to develop product, Agile will not provide the anticipated "turn-key" solution without proper planning and communication.

One of the key differentiators for the Agile methodology is the requirements-gathering process and the iterative approach that fosters alignment through increased communication between IT and business. In a more traditional approach, requirements-gathering and analysis can become a lengthy process that can last a year or more. A project team spends the time trying to define every detail of how a system should work prior to designing the product.

On the other hand, Agile provides a high level requirements process to define what the system should provide, not how the system should work. The "what" is required to start development early and show progress and, as the project continues, drives alignment between the success criteria and the end result. The iterative process promotes communication between and within teams to establish how the system should work. Iteration fosters communication between IT and business, software developers and business analysts, and creators and end users. The development teams gain greater insight into what is required and a better understanding of the desired outcome. It is acceptable for developers to contribute to the requirements gathering process. The goal of an Agile requirements gathering and delivery process is to promote communication, alignment, and early customer involvement to meet the goals and needs of the organization.

II. LITERATURE SURVEY

Requirement Engineering (RE) is the process of collecting, analyzing and modelling software requirements in a systematic manner [1, 2, 3]. Requirement modelling is the major challenge of automotive software development [4]. One of the main problems of RE is to describe the requirements in terms of concise and manageable formal models and to integrate models to form a consistent and complete understanding of the software to be developed. Requirements modelling and analysis are the most important and difficult activities in the software development. Software development is becoming more mature by advancing development processes, methods,

and tools. The famous Christ Honour and Other Served (CHAOS) has reported the statistics published by Standish Group show that still only about one third of software projects can be called successful, i.e. they reach their goals within planned budget and time [5]. Research on postmortem projects' analysis shows that the major problems comes when the requirements elicitation, analysis, specification, and management is not performed regularly. Deploying successful requirements process in a concrete organization is an important issue for software practitioners [6, 7]. While companies continue to use text based documents as major means for specifying and analyzing requirements, the graphical requirements modelling are getting increasingly more attention in industry. This trend has increased after Object Management Group (OMG) standardized Unified Modelling Language (UML) [8]. As we know, that a picture is worth a thousand words. It is also applies in requirements analysis, where business people have to communicate with software developers, who do not know their domain and speak a different technical language. Additionally, UML tools support refining requirements models with design and implementation details for enabling traceability, validation, prototyping, code generation and other benefits. In large software development projects, these features are very important for evolving and managing requirement models. There are some practical problems with UML complexity and lack of unified method or framework for requirements engineering [9]. Practitioners and scientists propose different approaches for eliciting and analyzing software requirements. The most popular tools that are used in modern requirements analysis is use cases. It was adopted by numerous companies, and described in requirements engineering textbooks [10, 11]. UML provides Use Case diagram for visualizing use case analysis artifacts. However, requirements analysis is not limited to use cases. In fact, they capture only end user-level functional requirements. A lot of research is also made in specifying business goals and processes, performing domain analysis. Although it was shown that UML might be extended and used for business modelling, the business modelers' community was not satisfied by UML, and created a new Business Process Modelling Notation (BPMN), which has become OMG standard as well. In many cases, they also apply Integration Definition for Function Modeling (IDEF) notations [12, 13]. In domain analysis, analysts continue to apply old-style Entity Relationship (ER) notation, which was popular in database design since 70s [141]. A significant attention is paid to business goals, business rules, business object lifecycles, business roles and processes in organization, which also can be done using UML [15, 16]. Real-time and embedded system developers have also come up with a different flavour of UML – System Modelling Language (SysML). It defines requirements diagram and enables capturing various non-functional and detailed functional requirements [17]. Also, it establishes specific links between requirements and other elements. Most popular requirements text books introduce various diagrams based on both UML and other informal notations, e.g. system context diagram, and hand-drawn user interface prototypes [11, 18]. The mentioned Requirements artefacts can be modelled using UML. Since UML is a general purpose modelling language with more Than 100 modelling elements (UML Meta classes) and without standardized method, practitioners apply it only Fragmentally, and at the same time, they do not make use of its powerful capabilities to define consistent, integrated, and reusable requirements models. Various researches have already been performed to produce framework for Creating UML models for MDD (Model-Driven Development) [12, 15]. This paper extends it with more Focus on the details of a specific part of the framework by applying UML concepts for requirements modelling.

III. PROBLEM DEFINTION

Errors in requirements elicitation are, overall, most serious in software development, and the hardest to repair. 70% of the systems errors are due to inadequate system specification

Classification of elicitation problems

- Problems of scope. The boundary of the system is ill-defined, so that unnecessary design information may be given, or necessary design information left out.
- Problems of understanding. Users have incomplete understanding of their needs; analysts have poor knowledge of the problem domain; user and analyst speak different languages (literally or figuratively); "obvious" information may be omitted; different users may have conflicting needs or perceptions of their needs; requirements are often vaguely expressed, e.g., "user friendly" or "robust".

Problems of volatility. Requirements evolve over time, either because of changing needs or because of changing perceptions by the stakeholders

IV. RESEARCH METHODOLOGY

Task 1 (Data Collection)

- For Data collection, we will be using questionnaires and visit many software industries.
- Primary sources of data like statistics given small & medium level software firms
- Secondary data like published articles by small & medium level software firms.

Task 2 (Data analysis)

- Value stream mapping tool will be used to identify non value added activity
- Failure mode effective analysis tool will be used to identify possible failure modes
- 7 quality control tools will be used to analyze quality after implementing the model.

Task 3(Data Behavior)

- Regression testing mathematical model to find out effect of variables.
- Chi square test to track behavior of variables.

Task 4(Mathematical Models)

We will be using a typical mathematical program which consists of a single objective function, representing either a profit to be maximized or a cost to be minimized, and a set of constraints that circumscribe the decision variables. In the case of a linear program (LP) the objective function and constraints are all linear functions of the decision variables. We will identify which activities to be targeted to get maximum value in terms of time.

Then network flow program its class of network flow programs includes problems as the transportation problem, the assignment problem, the shortest path problem, the maximum flow problem, the pure minimum cost flow problem, and the generalized minimum cost flow problem. By using this class because many aspects of actual situations are readily recognized as networks and the representation of the model is much more compact than the general linear program.

Integer programming will be used to solve optimization problems in which some of the variables are required to take on discrete values. Rather than allow a variable to assume all real values in a given range, only predetermined discrete values within the range are permitted.

We now investigate a finite-state stochastic process in which the defining random variables are observed at discrete points in time. When the future probabilistic behavior of the process depends only on the present state regardless of when the present state is measured, the resultant model is called a Markov chain.

REFERENCES

[1] D. Pandey, U. Suman, A. K. Ramani, Social-Organizational Participation difficulties in Requirement Engineering Process- A Study, National Conference on ETSE & IT, Gwalior Engineering College, Gwalior,2009.

[2] Dhirendra Pandey, U. Suman, A.K. Ramani, Design and Development of Requirements Specification Documents for Making Quality Software Products, National Conference on ICIS, D.P. Vipra College, Bilaspur, 2009.

[3] Dhirendra Pandey, U. Suman, A.K. Ramani , An Effective Requirement Engineering Process Model for Software Development and Requirements Management, IEEE Xplore, 2010,Pp 287-291

[4] M. Broy, I. Kruger, A. Pretschner and C. Salzmann. Engineering Automotive Software. Proceedings of THE IEEE. 95(2): 356-373, Febrary 2007.

[5] D. Rubinstein Standish Group Report: There's Less Development Chaos Today. SD Times, March 1, 2007.

[6] J. Aranda , S. Easterbrook , G. Wilson Requirements in the wild: How small companies do it. 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 39-48.

[7] M. Panis, B. Pokrzywa, Deploying a System-wide Requirements Process within a Commercial Engineering Organization. 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 295- 300.

[8] Object Management Group. Unified Modelling Language: Superstructure. Formal Specification, 15th IEEE International Requirements Engineering Conference (RE 2007), 2007.

[9] G. Engels., R. Heckel, and S. Sauer, UML – A Universal Modelling Language? In M. Nielsen, D. Simpson (Eds.): ICATPN2000, LNCS 1825, pp. 24-38, 2000.

[10] I. Jacobson, Object-Oriented Software Engineering. Addison Wesley Professional, 1992.

[11] K. Wiegers. Software Requirements. 2nd edition, Microsoft Press, 2005.

[12] Object Management Group. Business Process Modelling Notation Specification. Final Adopted Specification, version 1.0, 2006.

[13] O. Noran. UML vs. IDEF: An Ontology-oriented Comparative Study in View of Business Modelling. Proceedings of International Conference on Enterprise Information Systems, ICEIS 2004, Porto, 2004.

[14] P. Chen, P.-S. The entity-relationship model – toward a unified view of data. ACM Transactions on Database Systems (TODS), vol. 1 (1), 1976.

[15] Van Lamsweerde, A. Goal-Oriented Requirements Engineering: A Guided Tour. RE'01 – International Joint Conference on Requirements Engineering, Toronto, 2001, pp.249-263.

[16] M. Penker , H. Eriksson, E. Business Modelling With UML: Business Patterns at Work. Wiley, 2000.

[17] Object Management Group. Systems Modelling Language. Formal Specification, version 1.0, 2007.

[18] E. Gottesdiener, The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements. GOAL/QPC, 2005.

[19] M. Glinz, Problems and Deficiencies of UML as a Requirements Specification Language. 10th International Workshop on Software Specification and Design, 2000, p.11 - 22

[20] S. Konrad, H. Goldsby, K. Lopez, Visualizing Requirements in UML Models. International Workshop REV'06: Requirements Engineering Visualization, 2006.

[21] H. Behrens, Requirements Analysis and Prototyping using Scenarios and State charts. Proceedings of ICSE 2002 Workshop: Scenarios and State Machines: Models, Algorithms, and Tools, 2002.

[22] Da Pinheiro, P. Silva, The Unified Modelling Language for Interactive Applications. Evans A.; Kent S.; Selic B. (Eds.): UML 2000 – The Unified Modelling Language. Advancing the Standard, pp. 117-132, Springer Verlag, 2000.