Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 6; April-June, 2015 pp. 533-537

http://www.krishisanskriti.org/acsit.html

A Framework for Software Requirement Elicitation and Prioritization

Mizbah Fatima[1] and Mohd. Shahid Sagar[2]

[1]M.Tech Scholar Deptt. of CSE Al Falah School of Engg. & Technology Faridabad, Haryana.
[2]Deptt.of CSE Al Falah School of Engg. & Technology Faridabad, Haryana.
E-mail: [1]mizrizvi16@gmail.com, [2]er.shahidsagar@gmail.com

Abstract—*One of the most crucial phases of software development is concerned with the proper gathering of requirements. The success of a software system depends mainly on to what extent it meets the objective for which it was created. Requirement Engineering (RE) is the process of recognizing that purpose. Requirements engineering comprises of activities such as requirements elicitation, analysis, modeling, specification, verification, and management. There are various approaches for performing each of the above activities. In this paper the emphasis is on the extraction of the requirements and then organizing those requirements in a prioritized fashion. For this I have proposed a framework for elicitation of the software requirements using the GORE approach involving goal model of the KAOS methodology and the prioritization of the software requirements is done using TOPSIS method. This paper shall give an overview on the techniques used in the framework.*

Keywords: *Requirements; Requirements Elicitation; Requirements Prioritization; GORE approach; KAOS; TOPSIS;*

1. INTRODUCTION

Requirements' gathering is considered as one of the most important phases of software development and is usually concerned with determining whether or not the project being developed would be successful or not. This success of a software system depends mainly on to what extent it meets the objective for which it was created. Requirement Engineering (RE) is the process of recognizing that purpose [1]. This objective can be achieved by determining the accurate requirements of the software system.

Requirements are statements that specify what the system must do, how it must act, the properties it must exhibit, the traits it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as:

-A condition or capability needed by a user to solve a problem or achieve an objective;

-A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document;

- A documented representation of a condition or capability as in definition 1 or 2 [2].

The Requirements engineering activities can be carried out using the various available approaches. However in this paper we shall focus on the approaches that would lead to the extraction of the requirements and then their prioritization.

2. REQUIREMENT ELICITATION

Software Engineering aims at producing high-quality software that is within budget restrictions and project schedules. However majority of the software projects fail on these major issues such as quality, schedule and cost [4]. The significant reasons behind the software failure pertain to poor requirements. The very first phase of software engineering is the requirement engineering phase and comprises of various activities. Requirement Elicitation, one of the constituent activities of requirement engineering, however forms an extremely useful step in the development of any software system. Wrong elicitation practice is usually the primary reason for failure of most software systems. The process of searching, revealing, obtaining, and elaborating requirements for software systems is often referred to as Requirement Elicitation. The requirements are elicited rather than just captured or collected; that is there are discovery, emergence, and development elements to the elicitation process [5]. Requirements elicitation is further decomposed into activities of fact-finding, information gathering, and integration.

Requirements elicitation is usually carried out using an elicitation methodology or a series of techniques [5]. These techniques are used by the analyst to elicit requirements from stakeholders and other sources. These elicitation techniques can be broadly classified into the following: Traditional, Collaborative, Cognitive, and Contextual. The Traditional requirements elicitation techniques comprise of a large class of general information gathering techniques [1]. The Collaborative techniques involve working in cooperation with different stakeholders to elicit the requirements. The Cognitive techniques usually involve those techniques that are concerned with mental processes such as thinking, understanding, and

learning and were originally developed for knowledge acquisition for knowledge-based systems [7]. The Contextual requirements elicitation techniques involve elicitation of requirements taking place at the workplace of the customer. That is the requirements are gathered with the end user perspective. These techniques are useful in getting detailed knowledge about the work area of the customer that would not be uncovered by the use of other, more conventional methods [8]. The resulting product from the elicitation phase is a subset of the goals [6]. This leads to the concept of Goal Oriented Requirements Engineering (GORE) approach that is useful in defining, eliciting, organizing, analyzing and refining the requirements, so that the system requirements can meet the customer needs [8]. The Goal oriented requirements elicitation helps in identifying the requirements in the form of high level goals that should be incorporated in the software while conforming to the stakeholders needs. It mainly comprises of the following activities: goal elicitation, goal refinement and various types of goal analysis, and the assignment of responsibility for goals to agents. The main GORE approaches discussed in literature include: NFR framework, i*/ TROPOS, KAOS, GBRAM [8]. However this paper shall focus on KAOS in particular.

2.1 KAOS - Knowledge Acquisition in automated Specification

KAOS or Knowledge Acquisition in autOmated Specification is a goal oriented requirements engineering approach, developed by University of Oregon and University of Louvain [9]. This methodology is concerned with constructing requirements models and obtaining the requirements documents from KAOS models [10]. The KAOS uses the four models: goal model, responsibility model, object model, operation model [10].

KAOS methodology involves development of the requirements model and comprises of the following steps:

-Build a Goal Model depicting the requirements in the form of goals in AND/OR graph.

-Build a Responsibility Model in order to achieve those goals through the help of agents.

-Build an Object model along with building all the consistent and complete glossary of the problem-related terms that are used to write the requirements.

-Build an Operational model describing the behavior of the agents responsible for achieving the goals they are responsible for.

-Build the requirements document based on the requirements model.

-Validate your requirements by first reviewing the model.

The goal model is considered the base and starting point of the whole method. It declares the goals of the composite system and thus forms the basis for obtaining all the other models through these goals. The goal model represents a set of interrelated goal diagrams that are used to deal with a problem. The main idea behind this approach is to represent system requirements as business goals and objectives and hence focus on realizing these business goals. Goals are typically all the functional and non-functional requirements that should be incorporated in the system that is being developed, often through the assistance of some agents. After the preliminary analysis of the system and identification of the goals by the requirements engineer, these goals are refined into progressively simpler goals until they can be easily implemented. Thus sub goals are derived from the high level goals and are further refined into more concrete sub goals. This acyclic directed graph called the goal graph or the AND/OR graph has nodes which represent the goals to be achieved by the system and its edges express logical dependency relationships between the connected goals. The goal graph has two types of goal decomposition: the AND decomposition and the OR decomposition. The AND decomposition signifies that if all of the sub goals are achieved, their parent goal can be achieved or satisfied while in OR decomposition, the achievement of at least one sub goal leads to the achievement of its parent goal.

The KAOS Responsibility model is a compilation of derived responsibility diagrams. It involves entities called agents which may be humans or automated components that are concerned with achieving the goals/requirements. The assignment of the agents to fulfill the particular goal is done according to the goal model. The goals are always assigned to a number of agents. However, whenever there is a single agent response for the goal, it indicates that there is no room for any further goal refinement and this difference gives the analyst a criterion to stop refining goals into sub goals.

The Object model is basically concerned with linking the application domain and establishing constraints on the operational system. The objects could be categorized as entities, agents and associations where “entities” describe and translate the state of the object but do not carry out operations; the “agents” are concerned with performing the operations whereas “associations” are entities that are dependent on the object and do not have the ability to carry out the operations.

The Operation model represents all the behaviors that agents must have to accomplish their needs. Behaviors are basically operations performed by agents. These operations are used to manipulate the objects described in the object model: they can create objects, provoke object state transitions or trigger other operations through sent and received events [10]. An operation diagram thus describes how the agents need to cooperate in order to make the system work.

In KAOS, there is a template document which contains all the information extracted from the four models to specify the requirements document. The glossary part is derived from the object model; the requirements are specified according to the

goal model from top (the business/strategic goals) to bottom (the requirements). Requirements on the system architecture are obtained from the responsibility model and requirements for the system behavior from the operation model. This entire process leads to extracting the system requirements and ultimately resulting in a complete, consistent and unambiguous document. Thus the output of the four models is a complete requirements document. Lastly the requirements gathered through the KAOS methodology can be validated by reviewing them so as to build a high quality product by organizing collective reviews of the KAOS model.

KAOS finds its application in various industries such as mechanics, telecommunication, and health care and hence can be used for any type of information system. It is considered an efficient goal oriented requirement elicitation method that uses the concept of building requirements models. KAOS provides a systematic and sound way to organize requirements and uses graphical way to tackle a problem [10]. It allows defining concepts relevant to the problem description, clarifying the responsibilities of all stakeholders and also provides a clear hierarchy for stakeholders enabling easy and efficient communication [10].

3. REQUIREMENTS PRIORITIZATION

In large and complex projects one of the major concerns is prioritizing the requirements. It is often needed to prioritize the requirements so that the highest priority requirements can be implemented first. There are various stakeholders involved in the project and may have different requirements with each of them wanting each and every one of its requirements to be implemented in the system. Due to time, resource and cost constraints it is usually impossible to implement all of the requirements. The stakeholder views may clash during the requirements elicitation phase and hence the prioritization of requirements may become quite a daunting task in itself. By considering high-priority requirements before low-priority ones, one can notably reduce project costs and duration [11]. Requirement prioritization process identifies the most important candidate requirements of a software project that should be included in a certain release, and for this purpose different techniques are used.

Some of the common existing techniques for requirement prioritization are already provided in literature [13, 14]. Any of these techniques can be relied upon usually in case of a small project. However for large scale projects having thousands of requirements and multiple number of stakeholders there is need for a requirements prioritization method that should accommodate a number of issues such as size of the project, negotiation of requirements, feasibility measure, fuzzy concerns of stakeholders and multiple criteria viz. cost, performance, risk etc.[15].

According to observation the prioritization methods provided in literature [13, 14] may not be perfectly suited to meet simultaneously all the requirements of an application. A most suitable prioritization method for one application may not be an ideal fit for another application. A wrong selection of a requirements prioritization method may result in wastage of resources causing customers' dissatisfaction.

3.1. Multi-Criteria Decision making method for Requirement Prioritization

The different stakeholders involved in the project may have dissimilar needs with each of them wanting each and every one of its requirements to be incorporated in the system. In addition to this there may be disagreement among the stakeholders over different views during the requirements elicitation phase. Thus the process of requirements prioritization comes into play for allowing stakeholders to decide on the core requirements for the system and to deal with conflicting requirements, and resolve disagreements between stakeholders. Thus decision making is done by choosing the most appropriate available alternative. It is at this point that the concept of Multi-criteria decision making needs to be introduced here.

Decision-making involves finding the best option from all of the possible alternatives. Multiple criteria decision making (MCDM) refers to making decisions in the presence of multiple, usually conflicting criteria [16]. MCDM usually yields the most excellent choice out of significant alternatives with respect to a number of criteria. A number of Multi-Criteria Decision Making methods have been reported in literature, namely AHP, fuzzy AHP, TOPSIS etc. However we shall keep our emphasis on TOPSIS in particular.

TOPSIS is a MCDM method suitable for small sized projects where no hierarchical decisions are required [17].

3.2. TOPSIS- Technique for Order of Preference by similarity to Ideal Solution

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi-criteria decision analysis method, developed by Hwang and Yoon [18]. The main idea behind TOPSIS is that the chosen alternative should have the shortest geometric distance from the positive ideal solution and the longest geometric distance from the negative ideal solution [19]. In this method the alternatives are scored against a set of criteria. It is a method that compares a set of alternatives by identifying weights for each criterion, normalizing scores for each criterion and calculating the geometric distance between each alternative and the ideal alternative, which is the best score in each criterion.

In this method it is assumed that the criteria are monotonically increasing or decreasing.

Normalization is usually vital as the parameters or criteria are often of incompatible dimensions in multi-criteria problems [20] [21].

The basic terminology used in TOPSIS method is as follows: The alternatives here are the options/ requirements which are

to be evaluated for selection of the best; the criteria or attributes are those entities that will impact the selection of alternatives/ requirements; the weights estimate the relative importance of criteria. (Each attribute is given rating in the scale of 0-10 or 0-100 by a team of experts or decision makers.); decision matrix is a table used for making selection from a range of options.

In TOPSIS method there are assumed to be two different types of alternatives: The Ideal alternative is one which has the best attribute values(i.e. maximum benefit attributes and minimum cost attributes) where as the Negative ideal alternative is one which has worst attributes (i.e. minimum benefit attribute and maximum cost attributes).

For the detailed algorithm of TOPSIS the reader may refer to [22, 23, 24]. However the mathematical concept may be described as follows. The scoring may be against an absolute scale (e.g. dollars, effort-hours etc.) or a relative scale (e.g. 1-9 Likert scale or 1-3-9 etc.) Each criterion has a particular direction of preference (i.e. the more or less of that criterion is preferred for the prioritization. For example, effort may be less effort the better depicted with "–" symbol; or the other way round if higher effort implies higher priority, depicted with a "+" symbol) . The prioritization algorithm is based on the Vector Space Model of computation. Each alternative is assumed to be a multi-dimensional vector in vector space. The ideal alternative S' is the one which has the best value for each of the criteria. Consequently, S* is the non-ideal alternative – the one with the worst values for each of the criteria. (Depending on the direction of preference of the criteria (+/-) the best/worst is the maximum or minimum score in the set of alternatives for that criterion.) The algorithm rank orders the alternatives so that the distance from the ideal solution (S') is minimized and that from the non-ideal solution (S*) is maximized – hence the name, Technique for Ordered Preference by Similarity to Ideal Solution (TOPSIS).

4. CONCLUSION AND FUTURE WORK

The use of the proposed framework simplifies the elicitation and prioritization of requirements. The AND/OR Graph or Goal graph from the KAOS methodology provides a list of elicited requirements in a graphical manner whereas requirements prioritization focuses primarily on determining right requirements to meet stakeholders' apprehensions.

This work captures the requirements from the various stakeholders and employs Multi-Criteria Decision-Making approach as prioritization method thereby assisting the developer to address stakeholders' expectations without wasting resources and hence developing a system of high quality.

TOPSIS allows us to prioritize all the candidate requirements enabling easy decision making using both negative and positive criteria. It allows that a number of criteria can be applied during the decision process, and is advantageous over other techniques as it is simple and faster than AHP, FDAHP, SAW.

As a future work we may conclude that this framework may be applied to any project during the requirement engineering phase.

5. ACKNOWLEDGEMENTS

I have great pleasure to express my deep sense of gratitude and gratefulness to Prof. (Dr.) Anil Kumar, Vice Chancellor, Al-Falah University, Prof. Saoud Sarwar, Head, Department of Computer Science and Engineering, and my guide Mr. Mohd Shahid Sagar, Assistant Professor, Department of Computer Science and Engineering, Al-Falah School of Engineering and Technology for providing me with all necessary facilities in carrying out my work and directing me to move in the right path of research.

I express my sincere thanks to all my teachers and friends, as well as my parents and family for their generous help, moral support and involvement in my betterment.

REFERENCES

[1] Nuseibeh, B., & Easterbrook, S. (2000, June 4 - 11). Requirements Engineering: A Roadmap (2000). Paper presented at the the 22nd International conference on Software Engineering (ICSE 2000), Limerick, Ireland.

[2] IEEE (1990a). IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology

[3] Sommerville (1997a) Sommerville, I. & Sawyer, P. Requirements Engineering: A Good Practice Guide. New York, NY: John Wiley & Sons, 1997.

[4] Hofmann, Hubert F. and Franz Lehner, "Requirements Engineering as a Success Factor in Software Projects", IEEE Software, July/August 2001, pp. 58-66

[5] Didar Zowghi, Chad Coulin. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools: Engineering and Managing Software Requirements 2005, pp 19-46

[6] Rzepka, William E.[89] A Requirements Engineering Testbed: Concept, Status, and First Results. In Bruce D. Shriver (editor), Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, 339-347. IEEE Computer Society, 1989.

[7] Shaw, M. & Gaines, B. (1996). Requirements Acquisition. Software Engineering Journal, 11(3): 149-165

[8] Alexei Lapouchnian, "Goal-Oriented Requirements Engineering: An Overview of the Current Research" Depth Report, University of Toronto, 2005

[9] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal Directed Requirements Acquisition," Science of Computer Programming, Vol. 20, No. 1-2, April 1993, pp. 3-50

[10] Respect IT, A KAOS Tutorial, Objectiver, 2007.

[11] Hofmann, Hubert F. and Franz Lehner, "Requirements Engineering as a Success Factor in Software Projects", IEEE Software, July/August 2001, pp. 58-66

[12] J. Karlsson, "Software requirements prioritizing," in: Proceeding of 2nd IEEE International Conference on Requirements Engineering, pp. 110–116, 1996.

[13] J. Karlsson, C. Wohlin and B. Regnell, "An evaluation of methods for prioritizing software requirements," Information and Software Technology, pp. 939-947, 1998.
[14] Firesmith D (2004) Prioritizing requirements. J. Object Technol 3(8):35–47
[15] Karl E.Wiegers, Software Requirements, 2nd edition, Woodpecker publishers, ISBN: 81-7853-071-6
[16] George J. Klir, Fuzzy Sets and Fuzzy Logic, PHI publications, 1995,ISBN:81-203-1136-1
[17] Serkan, B.: Operating System Selection using Fuzzy AHP and TOPSIS Methods. Mathematical and Computational Applications 14(2), 119–130 (2009)
[18] Hwang, C.L.; Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. New York: Springer-Verlag.
[19] Triantaphyllou, E., 2000. Multi-criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers, Dordrecht.
[20] Yoon, K.P.; Hwang, C. (1995). Multiple Attribute Decision Making: An Introduction. California: SAGE publications.
[21] Zavadskas, E.K.; Zakarevicius, A.; Antucheviciene, J. (2006). "Evaluation of Ranking Accuracy in Multi-Criteria Decisions". Informatica 17 (4): 601–618.
[22] K. Yoon, ―Systems selection by multiple attribute decision making‖, Ph.D. Dissertation, Kansas State University, 1980.
[23] C.L. Hwang and K. Yoon, ―Multiple Attribute Decision Making: Methods and Applications‖, Springer-Verlag, New York, 1981.
[24] R. Wang, ―Performance evaluation method - Technique for order preference by similarity to ideal solution (TOPSIS), researcher.nsc.gov.tw/public/caroljoe/Data/02182133671.ppt.