A STRUCTURED APPROACH FOR EXTRACTING FUNCTIONAL REQUIREMENTS FROM UNCLEAR CUSTOMERS

Mohammed A. Hagal, Omar M. Sallabi,

Faculty of Information Technology, Garyounis Univ. Benghazi, Libya
mohdhg@yahoo.com, osallabi@garyounis.edu

ABSTRACT

Many challenges are facing the developers during specification of the requirements for new systems. The errors in the requirements that detected in last stages of the system (such as implementation stage) will be very expensive to correct because it may require rework effort. Such errors sometime occur when customers do not have ability to articulate their requirements or developers make implementation compromises in order to get working prototype rapidly which might cause inappropriate design decisions and inefficient algorithms. Thus, effective management of extracting requirements is essential. In this paper we propose a guideline approach consisting of some managed stages aim to help in extracting precise software requirements.

***Keywords*:** *Software engineering, Requirement engineering, UML, Prototyping.*

1. INTRODUCTION

"Requirement is a condition or capability needed by the user to solve a problem or achieve an objective" [1]. The marketing culture indicates that the developer should work on gaining customer confidences and keeping them in touch through giving services that are compatible with their requirements. The marketing thought focuses on viewing the services through the customer point of view and then manipulated through customer's acceptance of the requirements and justify customer needs.

System developing is an interacting and complementary operation, characterized by close connection and direct relation between the developer and the customer and it focuses on the quality, precisely, completing the information that interacted between them which is considered as the important element for identifying the customer needs.

Often, the customer and the developer may face some difficulties in identifying and clarifying the customer requirements precisely, as a result from the miscommunication between them, which affects generating clear requirements. To establish good communication between the developer and the customer, the understanding and clarification of the customer requirements precisely is required.

Therefore, the need for a guideline approach that organizes and manages user requirements in the form that simplifies the customer requirements by narrowing and bridging the gap that may result between the customer and the developer.

In this paper we propose an approach consisting of some managed stages aimed to generate precise software requirements. These stages are named as preparation stage, initial extracting stage, and the enhancement of the initial extraction stage which is the refinement of the initial extracting stage.

2. BACKGROUND

Requirements elicitation is often one of the challenges that lead to system failures, because it is usually incomplete, unambiguous, too many requesters and different views of different users, etc. In addition to these, many authors focus on analysis quickly without focusing deeply on elicitation [2]. Furthermore, most elicitation problems are caused by problems of scope, problems of understanding or problems of volatility [3]. Some research has been carried out on software risks which may trend to software failure because the shortage of defining the precise dependability and traceability of software requirements [4, 5]. To overcome the above mentioned reasons, many researchers attempt to find/define techniques to solve these problems.

Many researchers have proposed that prototyping can be considered as useful technique; especially in elicitation to avoid the risk that is caused from misunderstanding of both (developers and customers).

Prototyping is useful for risk assessment and as a means for validation of customer requirements. There are many well known and commonly used approaches to prototyping such as throwaway versus evolutionary [1], horizontal versus vertical [6], textual versus visual, and executable versus non-executable prototypes.

Most of the available literatures on prototyping are conceptual and there exists a lack of empirical studies to provide a comprehensive evaluation of prototyping approach based on the field experience. Prototyping approach, however, is not free from weaknesses[7].

Oshiro, Watahiki and Saeki [8] proposed a method for Requirements Elicitation. This method is used for the stakeholders to identify their ideas that independently get into their heads and consider these as needs (initial goals). Each member thinks something related to the selected goal and makes it concrete as an idea, and writes down a generated idea on a paper card, so that all of the members can read it. This method requires customer expertise and may take a long time to determine what the customer wants from the system.

Therefore, most of the generating requirements techniques depend on the vision of the software's developer (requirement engineer) to the system. So, there is a need for an approach that organizes and manages user requirements in the form that simplifies the customer requirements by narrowing and bridging the gap which may result between the customer and the developer, and lead to a well documented specification.

3. THE PROPOSED APPROACH

The proposed approach represents a structured process consisting of some managed stages. Some documents are generated throughout these. This approach starts with the preparation stage and extends up to the final stage(enhancement of extraction) that represents the concluding stage of the approach. Figure 1 illustrates the conceptual overview of the proposed approach and the outcome documentations of its stages

The subsequent sections explain the stages of the proposed approach in more details.

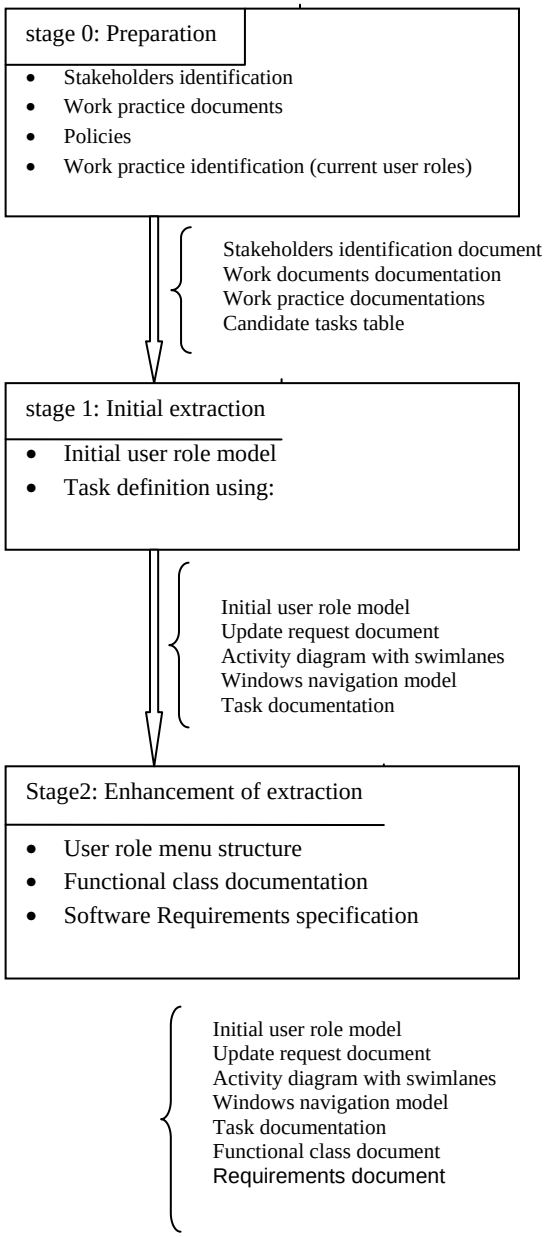

Figure 1: Conceptual overview of the proposed approach

3.1 PREPARATION STAGE

The objective of this stage is to understand the system to be developed and familiarizing the developer to the problem domain, to elicit and record the initial requirements from stakeholders correctly. The subsections explain in more details the activities to be done in this stage.

3.1.1 STAKEHOLDERS' IDENTIFICATION

Stakeholder is "anyone who benefits in a direct or indirect way from the system which is being developed" [1]. The first step towards discovering all the requirements is to understand who all the stakeholders are, and what roles they are expected to play i.e. we need to understand the project's sociology. An organization chart can be useful to identify the other stakeholders that might know why the system is requested.

Next, the developer should write the needed information about the stakeholders of the system such as, their names, positions, relation types (i.e. direct, indirect) with the system (i.e. the stakeholder weight), and the relation descriptions of the stakeholders with the system (what the stakeholders roles can play in the system to be developed).

3.1.2 INITIAL REQUIREMENTS ELICITATION

In the beginning, the customer might not have enough ability to determine their requirements(needs) precisely. To simplify that, we take the user's tasks that currently perform according to the roles they play as a starting point of the negotiation. This will simplify the interaction between the customer and the developer. Also, it may lead to capture some details that the developer may need. Taking the customer's feedback about the mentioned initial requirements is important to clarify what the customer exactly needs. Figure 2 illustrates the proposed document for the user role description. Furthermore, addressing the collected work practice related documents may be useful to be considered as one of the requirements resources. These documents may consist of working documents and report documents, etc.

User role	
Work description	Related documents

Figure 2: work practice representation document

3.1.3 SETTING INITIAL REQUIREMENTS

At the end of the current stage, based on the previous document the developer identify (capture) list of the candidate requirements. These will be considered as the starting point of negotiation to identify the initial requirements. The developer lists the proposed needs in priority order (figure 3), where each need will consist of one or more tasks (i.e. each need may consist of one or more scenarios). So the advice is to consider each scenario as a task. UML use case diagram can be used to present customer's needs, the relation among them and the interactions among actors with these needs. Use Cases represent needs in abstract level without taking into considerations the tasks that each Use case may contain. So, Figure 3 is proposed to show the needs and the tasks that each need may contain.

Need.#	Task#	Task name	Related documents #

Figure 3 Candidate requirements document

3.2 INITIAL EXTRACTION STAGE

The stage starts with the tasks conducted by each user role i.e. the tasks which can be implemented by the computer, considering them as the initial requirements. The idea focuses on preparing a fast prototype for each task. Each prototype focuses on the main states each task may contain. These states are shown in the form of windows navigations. The conceptual overview of this stage is shown in figure 4 which shows visually the sequence of the stage. The subsequent sections of this section explain in more details the activities to be followed in this stage.

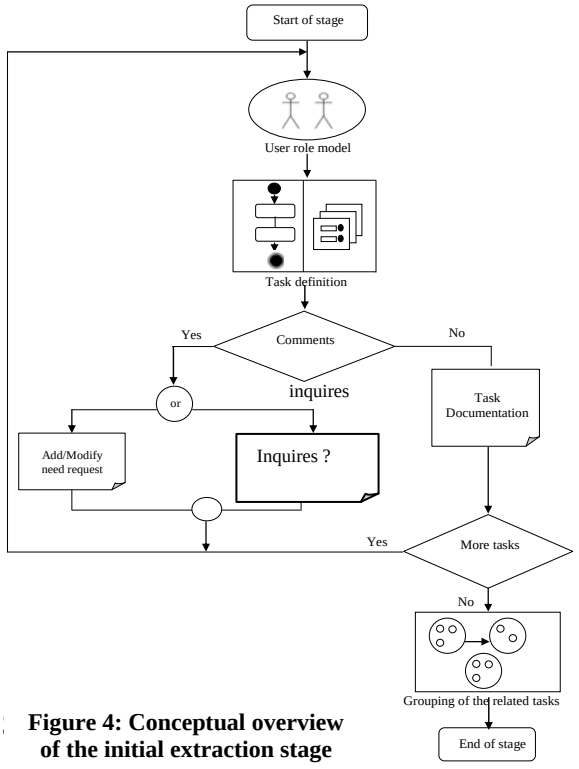

Figure 4: Conceptual overview of the initial extraction stage

After identifying the candidate list of tasks that can be implemented by computer, these tasks will be conducted to their relative user roles (to simplify what tasks the computer can perform from each user role perspective). These will be shown in the User role model which can represent the initial menu structure (just in requirements stage). The idea of the proposed model focuses on the system's end-users point of view. It shows the interactions between the system being developed and its proposed end-users according to their proposed tasks that they are responsible for (i.e. each role responsible for one or more tasks). Figure 5 bellow illustrates such model.

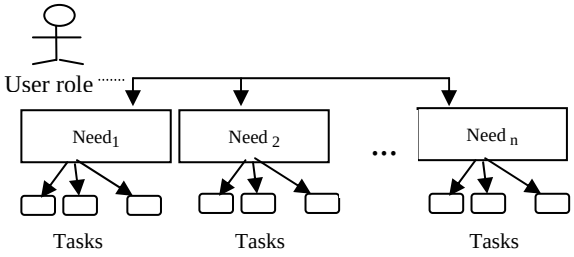

Figure 5: User role model

3.2.2 TASK DEFINITION

Each task needs to be expressed how it works. It will be demonstrated in the activity diagram with swimlanes to show the responsibilities of the system and the user to execute the task(i.e. to show the interactions between the user and the system). Figure 6 illustrates an example of checking participant information in a simple library system, and figure 7 shows its equivalent windows navigations which will be prepared to show most of states that the task will contain. Windows navigations consist of one main interface (as starting window) which is may be followed by one or more navigated windows and the event that cause the followed window(interface). The description under the line indicates what the event has caused.

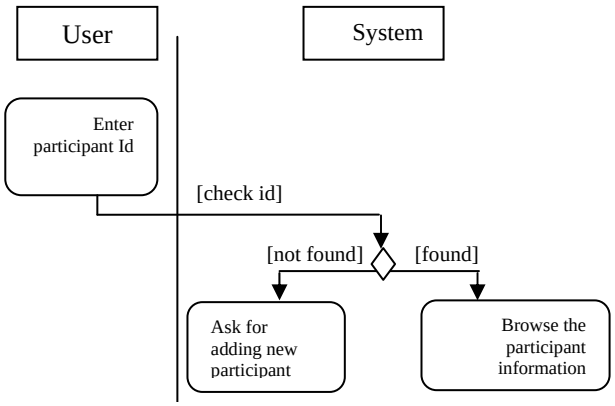

Figure 6: Activity diagram with swinmlanes example

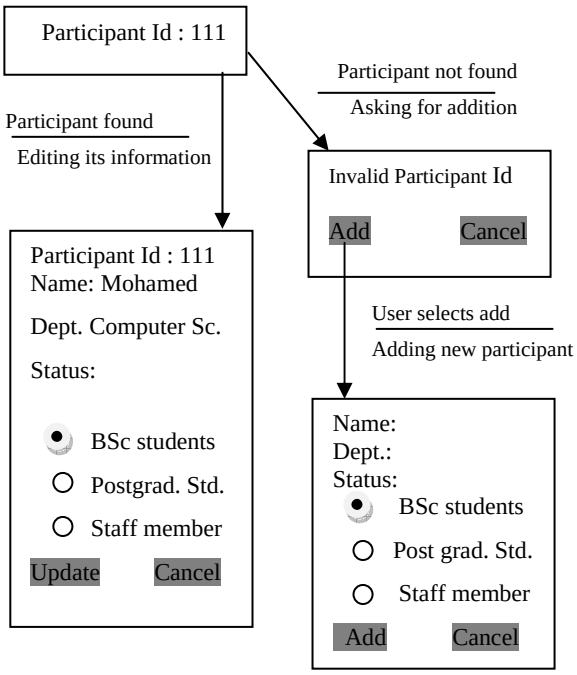

Figure 7: Windows navigation example

Customer's comments should be documented which may require an update or change in tasks or even adding new task. Documentation should contain the following:

Document number
Document name
Requirement number
Task number
Task name
Request type (add/Modify)
Source of request
Related documents
Impact of request & comments

Each updated and new added tasks will be demonstrated/re-demonstrated (when necessary) as mentioned above. Then each evaluated task will be documented.
On the other hand, the developer may face some inquiries which need to be interpreted and answered before or during demonstration of the prototypes to the customers. The proposed document contains the inquiries points and customer's feedback about the required inquires by the developer.

For each evaluated task, the developer documents its descriptions which contain the purpose of the task, the assumptions, the constraints/security, the

primary user, the secondary user, the related updated documents, input, process and output of the steps that the task contains.

3.2.3 GROUPING THE RELATED TASKS

If there are no more requests, the developer groups the related tasks and considers the tasks that complement the same group as one functional class taking into consideration the users roles in the grouping operation.

Furthermore, by grouping the tasks, we can manage various complexities and organize the number of tasks in the system, best for the developer to prioritize the functional classes. Figure 8 shows an example of two functional classes with their relations (the arrow direction shows the dependency or relation type). The documentation of the grouping tasks may consist of the functional class name, the indication or the description of its belonging tasks as well as the other functional classes which have relations with it.
Getting customer's final evaluation on the functional classes to be performed is also important for good specifications. The customer's feedback may include changes, updates or addition of new functional classes or tasks.

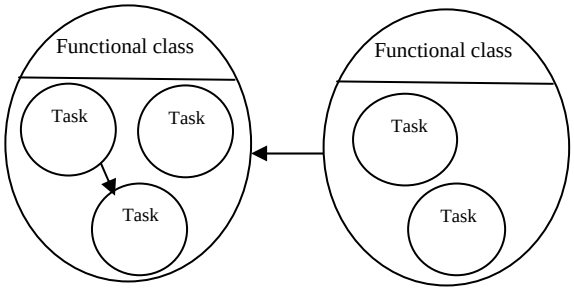

Figure 8: Example of two functional classes dependencies

3.3 ENHANCEMENT OF EXTRACTION STAGE

In this stage, the developer goes more in depth to orient the customer more towards defining the functions to be performed by the system. It may be that some more requirements need to be expressed. Figure 9 illustrates the conceptual overview of this stage which shows visually the sequence of the stage and the subsequent sections of this section will explain this stage in more details.

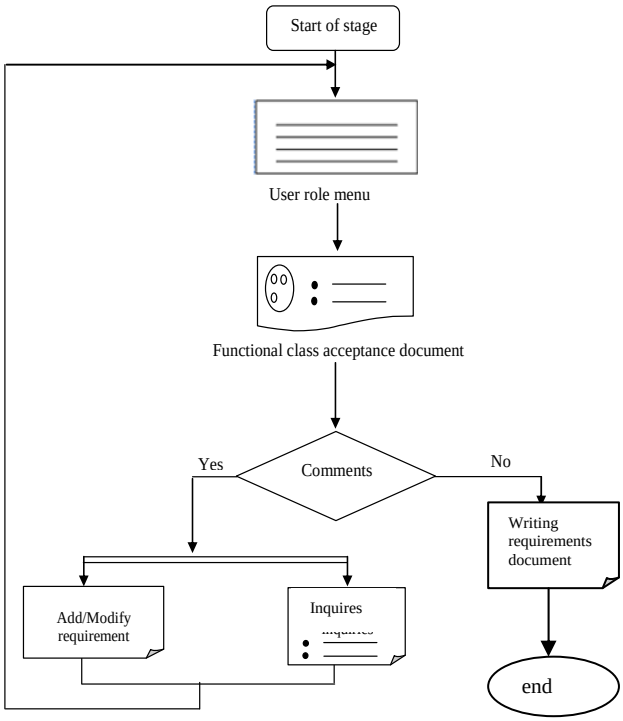

Figure 9: Conceptual overview of the Enhancement extraction stage

3.3.1 REFINED USER ROLE MENU STRUCTURE

Demonstrating the menu structure for each refined user role may lead the customer to more update, change or even addition of new requirements which would increase the number of ideas and suggestions for requirements specifications and improvements. Furthermore, one of the useful reasons for demonstrating the menu structure of each user role is to check whether all functions required by the customer are included.
The first window in the menu (i.e. the user's parent window) contains one or more functional classes. Each Functional class contains one or more main tasks. Each task will be represented as a choice in the function's class parent window. Each choice will be represented as a scenario. "A scenario is a story which tells us how a specific task instance is executed" [9]. Figure 10 shows the proposed user role menu structure.
Getting customer's final evaluation on the functional classes to be performed is also important for writing good specifications. Each functional class should be checked with customer to get the feedback. The feedback may contain modification or addition of more tasks or functional classes.

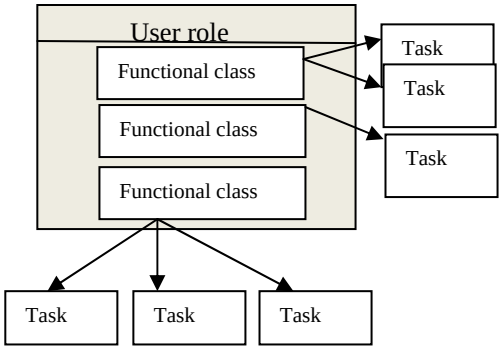

Figure 10: User role menu structure

3.3.2 FUNCTIONAL CLASS DOCUMENTATION

For each finally evaluated Functional class, the developer has to record its description which contains the last versions of the tasks. Each task will be considered as a scenario in the functional class. The task descriptions are illustrated in more detail in the task document. The document for documenting the functional class contains the following:

- Functional Class Number
- Functional Class Name
- Purpose of the Functional Class
- Related Functional Class(es)
- Constraints or Security required
- Task(scenario) Number, Task Name and Data Items in the task (i.e. the data items including the ones appearing in the interface prototype and those not appearing in it (i.e. the internal data items))

Finally, this stage ends with the combining of all the documentations obtained during the various stages of the proposed approach into a single document, which is focused in only functional requirements parts of the systems with its resources documents, while other parts of the IEEE standard 830, 1993(software requirements specification) is out of the scope of this work. The template of this standard is illustrated in [9].

4. CONCLOSION

We have proposed an approach which aims to extract requirements specification in a systematic way. In this approach, we have attempted to define a way to get customers needs in the form that could meet their satisfaction. We trace, control, validate and manage the customers' requirements by documenting them in the form that makes the developers and customers understand each other.

In this approach the work practice document has been used as a basis for identifying the candidate needs which can be used as a starting point of negotiations between the developers and customers by using the work breakdown structure for these needs to get the tasks that each need contains. These tasks are demonstrated using the prototyping approach in the form of Windows Navigation Model. Then structured guide lines are provided to document the requirements. Finally, we conclude that generating requirements based on tasks technique may help to find the estimation of the project size and its complexity.

5. REFERENCES

[1] Pressman S. Roger, Software Engineering: A practitioner's Approach, 6th edition, 2005.

[2] Daivy b. and Cope c., Requirements Elicitation-what's missing? Journal of Issues in Informing Science and Information Technology (IISIT), 5, 543-551 2008.

[3] Rajagopal P. and et.al, A new approach for requirements elicitation, proceeding of 6th conference on software engineering, IEEE, 2005.

[4] Nafo A., Master thesis: Applying validation and verification in local software systems, Higher studies academy, Benghazi, 2007.

[5] MARCOS A., Carlos H and Edger T., Identifying dependability requirements for space software systems, J. Aerosp.Technol. Manag., São José dos Campos, Vol.2, No.3, pp. 287-300, Sep-Dec., 2010.

[6] Snyder C., Using Paper Prototypes to Manage Risk, http://www.snyderconsulting.net/us-paper.pdf, retrieved date: 22.8.2007.

[7] Metzger Andress, Stefan Queins, A Reuse- and Prototyping-based Approach for the Specification of Building Automation Systems, University of Kaiserslautern, Germany, 2006.

[8] Oshiro K., Kenji W., Motoshi S., Goal-Oriented Idea Generation Method for Requirements Elicitation, 11th IEEE International, requirements Engineering Conference, 2003.

[9] Vliet Van, Software Engineering: Principles and practice, Biddles ltd, King's lynn, Norfolk, 2004.