A CONCEPTUAL FRAMEWORK FOR SOFTWARE REQUIREMENTS VALIDATION

Ahmed A. Ahmed 1, Ayman E. Khedr 2*, Sherif A. Kholeif 3

1 Faculty of computers and information systems, Information systems dept., Helwan University, Cairo, Egypt
2 Faculty of computers and information technology, Future University in Egypt (FUE), Cairo, Egypt
3 Faculty of computers and information systems, Information systems dept., Helwan University, Cairo, Egypt
**ayman.khedr@fue.edue.eg*

Abstract

Requirements validation is one of the most significant and critical parts of the requirements engineering. This activity ensures that the set of requirements is accurate, right, complete, and consistent. Requirements validation is considered as the key activity because mistakes found in a software requirements document can lead to extensive rework costs when they are discovered either during development or after the system is in service. There are some commonly used bases to validate user requirements such as: Natural language, Design description languages, Graphical notations and Mathematical specification languages. Whereas the graphical notations are the most suitable means to be used in software requirements validation because it is easy to understand, and it can be easily created by analyst and time took. Therefore, this paper adopts the map concept which is a graphical technique for discovering the hidden flaws in software requirements in the early phases of software development lifecycle.

Keyword: *Requirements Validation, Requirement Engineering, Concept Map, Software development, Software Requirements Specification (SRS), Natural Language, Design Description Languages, Mathematical Specification and Graphical Notations.*

1. Introduction

Basically, the success of any software system depends on the degree to which it meets the customer requirements and the degree of customer satisfaction. Defective requirements always lead to defects in the final software system product, which is not desired by either the analyst, end user, designer or developer of the software product. Fixing such defects in later stages of the software development process or after the deployment of the software always is costly and difficult [1]. Hence, it is important to validate software requirements in a timely and by appropriate Methods. There are many validation requirement methods and ways adopted on the different basis. Based on which it can/not be suitable for the software development. In the following sections we will explain where and when requirements validation occurs and what concept map is.

1.1. Software engineering

Software engineering is the field of engineering that is related to the study and application of theoretical approaches to the development, operation, and maintenance of software. Software engineering is the basis of knowledge about the creation, development, and maintenance of a software system and which emanate from those two activities. First: the creation and maintenance of actual software. Second: research tools and methods to create software, understanding of the nature of the software system and how using it [2]. Software engineering consists of several interrelated sub-disciplines, including software design, software construction, software testing, software maintenance, requirements engineering, software configuration management, software quality management, software engineering management, software engineering process and software engineering tools and methods as shown in figure 1.

1.1.1 Requirements engineering

IEEE standard 610 ‘‘the glossary of Software Engineering Terminology" provides a definition of requirements thus, ‘‘A condition or capability that must be met, fulfill or possessed by a software system or software system component to satisfy a contract, standard, specification or other formally imposed documents recorded ‘‘[3]. The first appearance of the requirements engineering term was probably in 1979. [4] In the 1990s with the publications and establishment of a conference series on requirements engineering of IEEE (Institute of Electrical and Electronics Engineers) computer society about requirements engineering, it comes into general use. [5] Requirements engineering consists of several interrelated activities, i.e. it is based on several activities which are including requirements elicitation, requirements identification, requirements analysis, requirements specification, system modelling, requirements validation, and requirements management [6]. As shown in figure 1. These activities involved in requirements engineering vary widely, depending on the type of systems being developed and the specific practices of the organization(s) involved. These are sometimes presented as chronological stages although, in practice, there are a considerable interleaving of these activities.

1.1.1.1 Requirements validation

Requirements validation is one of the most challenging requirements engineering activity, this activity is checking that the documented requirements and models are consistent and meet stakeholder needs, but instead of a formal validation this could be done using simulation technology or converting the behavioral model into a human-readable textual description [6][1]. It's the main activity for requirements engineering in software engineering and a critical in the pursuit of quality software [6]. Validating requirements mean ensuring that the set of requirements are correct, complete, and consistent, a model that satisfies the requirements can be created, and a real-world solution can be built and tested to prove that it satisfies the requirements [7]. Requirements need to be validated at an early stage of software system analysis to address inconsistency and incompleteness issues [8]. The importance of correctly determining the requirements of a system at the very beginning of the development process it is a well-known fact. Experience shows that the incorrect definition of the requirements leads to the development of deficient systems, increases the cost of its development or even causes projects to fail [9]. In relation to the current practices in requirements validation [10] [11], there are some commonly used bases with which the requirements are specified, and the validation test cases are a natural language, design description languages, mathematical specification languages and graphical notations languages. As shown in figure 1, among these bases for specifying software requirements, graphical notations language are the most suitable means to be used in software requirements validation [12].

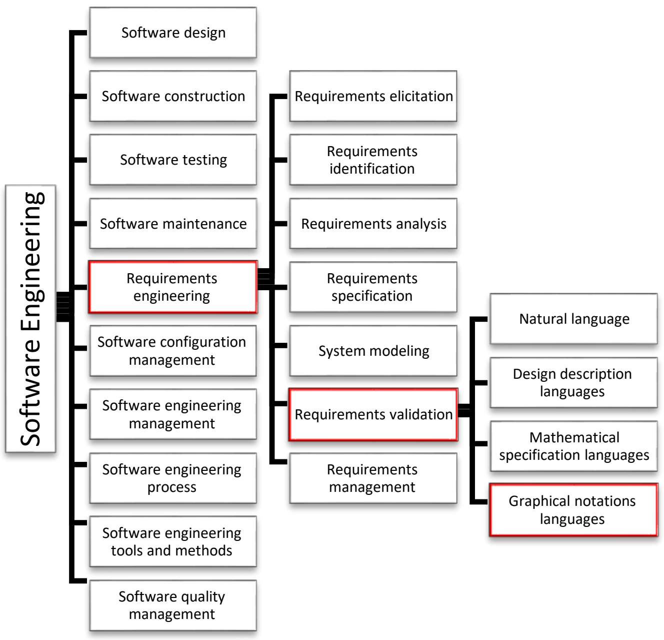

Software Engineering
Software design
Software construction
Software testing
Software maintenance
Requirements engineering
Software configuration management
Software engineering management
Software engineering process
Software engineering tools and methods
Software quality management
Requirements elicitation
Requirements identification
Requirements analysis
Requirements specification
System modeling
Requirements validation
Requirements management
Natural language
Design description languages
Mathematical specification languages
Graphical notations languages

Figure 1: Requirements validation

1.2 Concept map (Cmap)

Concept map is a graphical approach for representing knowledge and their relations to each other that is gathered about an issue or subject in diagrammatic format, by putting this knowledge in concepts or words format and then placing a words in a box or oval and using arrows or lines to link it to other words, showing the relationship between these concepts [13] [14]. Concept map (Cmap) helps to structure and organizing the knowledge that is built by concepts; this guarantees a better representation of knowledge [15]. Concept maps enable beginners and experts alike to understand and share knowledge and their bases [16]. In a concept map, each concept or word is connected to another and linked back to the original concept, word or phrase. Concept maps (Cmaps) are diagrammatic models of knowledge, which are based on a theory of human learning and 45 years of research. In the 1970s, science educator Joseph Novak developed a Concept map to assess how well his students understand science. By the early 1990s, it had become clear that a tool that facilitates the expression of one understands of any topic, in a simple graphical format that is easy to comprehend by others, could be used by people of all ages and for purposes beyond schools [17] [18]. Thus, Cmaps were introduced to the workplace as a solution for problem-solving.

1.3 Software Development

Software development is a very complicated process of developing from requirement analysis through sequential phases regularly and deployment [19]. Therefore, the software development process may include new research or development or design templates, modification or re-use or re-engineering, maintenance, or any other activities that may lead to the production of the software product. One of the three most important purposes of software development is to meet the special needs of a customer, user or company (as is the case with custom software). Software development passed through several phases which is known as software process or software development life cycle, and these phases are: - planning, analysis, design, development, testing, and deployment as shown in figure 2.

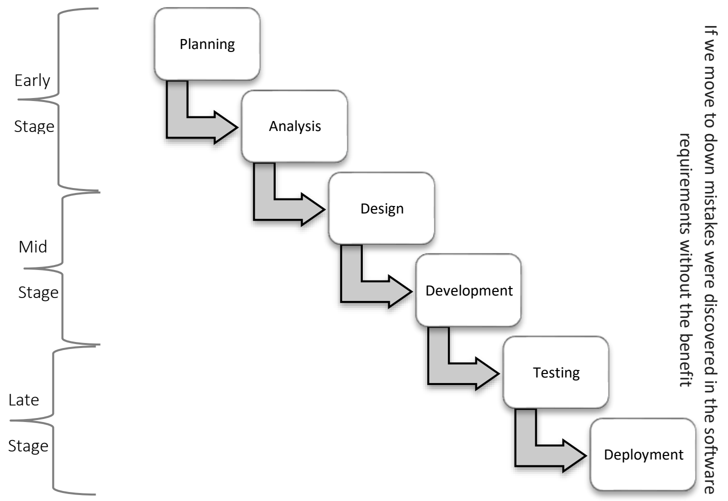

Early Stage
Mid Stage
Late Stage
Planning
Analysis
Design
Development
Testing
Deployment
If we move to down mistakes were discovered in the software requirements without the benefit

Figure 2: the early stage in the Software Development Lifecycle 'SDLC'

It also can be placed in three stages as follows early stage include planning and analysis, middle stage includes design and development and late-stage include testing and deployment as shown in figure 2.

2. Background

Researchers began developing approaches and methods based on these bases to requirements validation. Each one of them deals with software requirements validation from a different angle but most of them confirm that relying on the users to validate requirements is the best strategy. But what is the meaning of each base? What are the advantages and disadvantages of each bases? What is the best of these bases and why? All this will be clarified through the following section:

2.1. Natural Language

Natural language is one of the most important bases to software requirements validation. In software engineering field the natural language means a human language that has evolved naturally as a means of communication among people as shown in figure 3 such as Arabic, English, French, German, Italian, Hindi, or Chinese.

Figure 3: Natural language

Natural language is the sounds and symbols used in human communication through speech and writing. In software requirements validation case, the analyst after gathering requirements from stakeholders you can validate them through talk statements written or by a set of predefined questions [20]. Often it uses natural language systems analyst in gathering and documenting software requirements due to its flexibility and ease of understanding of widely among people [12]. So, there are some approaches and models that were adopted based on natural language in the software requirements validation. Although advantages of natural language may introduce ambiguity especially when talk audibly because different stakeholders may have different interpretations of the talk audible or statements written in natural language.

2.2. Description Languages

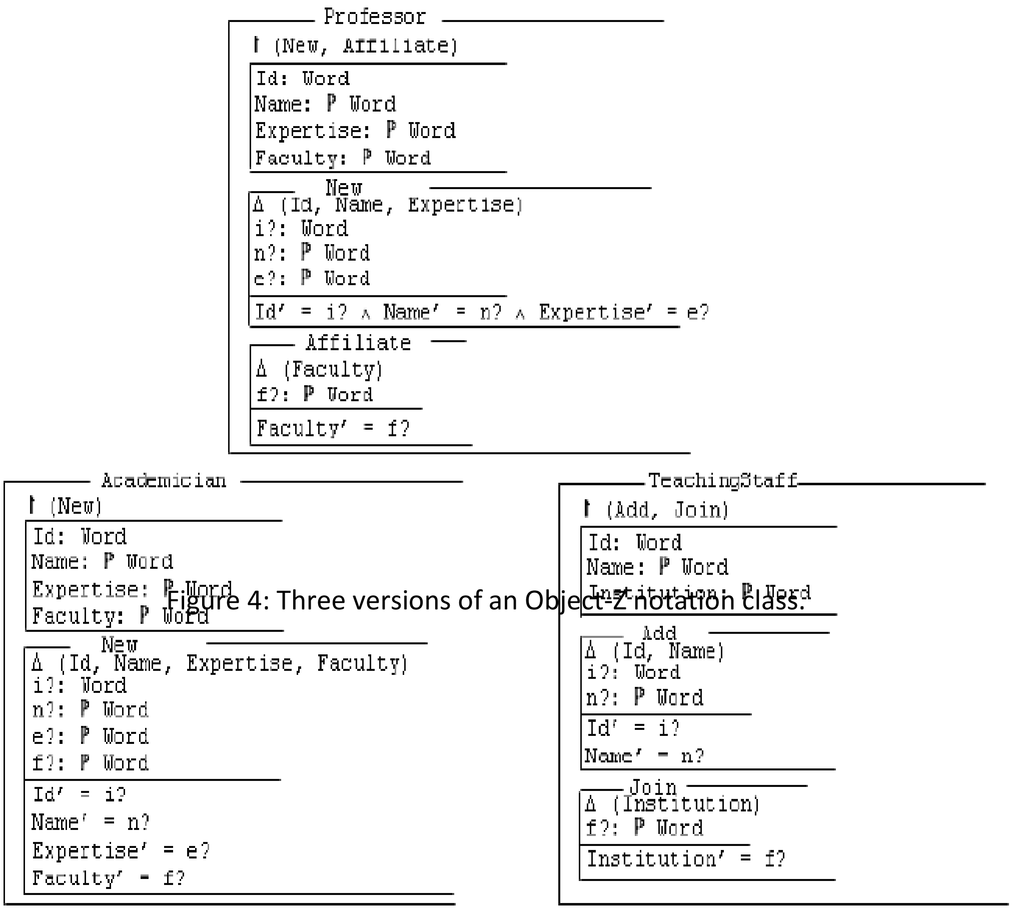
Description language is very similar language to a programming language, but with more abstract features to specify the requirements. There are many types of languages that depend on the approach of description languages, Architectural Description Language (ADL) and design description Language. I.e. description software system requirement in a format like a programming language but without a programming syntax or identify specific programming language.

Architectural description language is a formalism that allows the representation of architectures of the software system [21]. It is a formal language that can be used to represent the architecture of a software system. Design description language is a language style similar to a programming language but with more abstract additions and features to specify the software requirements [20]. It is the abstraction and specification of patterns and formulae of functionality that have been or will be achieved, design focus or specifies functional requirements, whereas architecture focus on non-functional decisions and partitions functional requirements. One of the most important advantages of using a design description language for software requirements validation is that the specification can be transformed directly to a certain programming language. However, not all customers and stakeholders are familiar with the specific description language so that there may exist some connection and communication problems [22].

2.3. Mathematical Specification Languages

These specifications are derived based on mathematical expressions, such as the Z notation [21]. The specifications are very precise and concise. Validation of the requirements can be achieved through mathematical proofing mechanisms. However, a major obstacle to adopting mathematical expressions is that it is difficult for the common users to fully understand the mathematical specifications of the software requirements.

Z notation is a formal specification language use mathematical notation for describing and modelling the properties which software systems must have [23]. Object-Z notation has been used to specify the three versions as shown in figure 4.


```
Professor
↾ (New, Affiliate)
Id: Word
Name: ℙ Word
Expertise: ℙ Word
Faculty: ℙ Word
  New
  Δ (Id, Name, Expertise)
  i?: Word
  n?: ℙ Word
  e?: ℙ Word
  Id′ = i? ∧ Name′ = n? ∧ Expertise′ = e?
  Affiliate
  Δ (Faculty)
  f?: ℙ Word
  Faculty′ = f?

Academician
↾ (New)
Id: Word
Name: ℙ Word
Expertise: ℙ Word
Faculty: ℙ Word
  New
  Δ (Id, Name, Expertise, Faculty)
  i?: Word
  n?: ℙ Word
  e?: ℙ Word
  f?: ℙ Word
  Id′ = i?
  Name′ = n?
  Expertise′ = e?
  Faculty′ = f?

TeachingStaff
↾ (Add, Join)
Id: Word
Name: ℙ Word
Institution: ℙ Word
  Add
  Δ (Id, Name)
  i?: Word
  n?: ℙ Word
  Id′ = i?
  Name′ = n?
  Join
  Δ (Institution)
  f?: ℙ Word
  Institution′ = f?
```
Figure 4: Three versions of an Object-Z notation class.

In mathematical specification languages case, we can get precise and concise results, because of the accuracy and clarity of the mathematical operations. So, any approach or technique depends on mathematical specification languages will be difficult for customers to understand and absorb it.

2. 4 Graphical Notations

Graphic notation is the representation of data, information or knowledge using text annotations and visual symbols. Graphical models are commonly used to define the functional requirements

for the software system [1], the beginning of graphic notation term and the use of symbols to transfer and record information, starting with a representation of musical sounds during the 1950s and 60s as shown in figure 5.

fl.
p
vla.
cl. + fg.
mf
ob.
p
pp str.

Figure 5: Standard music notation.

The most important basis, from the beginning in representation of musical sounds during the 1950s to concept map with Joseph D.Novak in the 1970s, Unified Modelling Language (UML) in the 1990s, Entity-Relationship Diagram (ERD) and Data Flow Diagram (DFD); we will explain some of them in detail in this chapter. Requirements generally fall into two types functional and non-functional; the difference between these two types is straightforward. A functional requirement which specifies what the software system should do, whereas non-functional requirement which specifies how software system performs a certain function.

Graphical notations are not limited to functional requirements only but can be used to representation functional requirements [1] and non-functional requirements [25] as shown in figure 6.

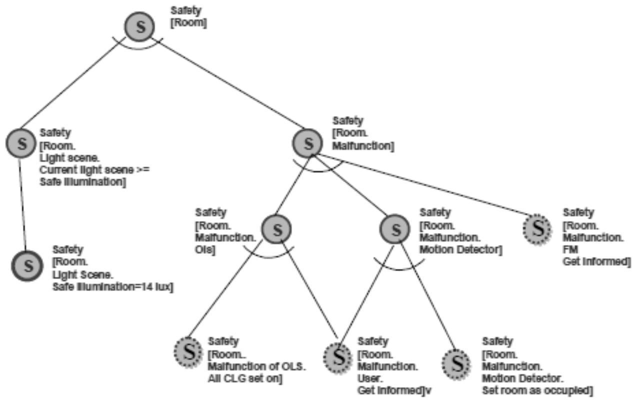

Safety [Room]
Safety [Room. Light scene. Current light scene >= Safe Illumination]
Safety [Room. Malfunction]
Safety [Room. Light Scene. Safe Illumination=14 lux]
Safety [Room. Malfunction. Ols]
Safety [Room. Malfunction. Motion Detector]
Safety [Room. Malfunction. FM Get Informed]
Safety [Room. Malfunction of OLS. All CLG set on]
Safety [Room. Malfunction. User. Get Informed]v
Safety [Room. Malfunction. Motion Detector. Set room as occupied]

Figure 6: An example of non-functional requirements graph [25].

Figure 6 shows the safety [Room] example, of a non-functional requirements graph, where can represent the operationalization in two different ways, Static and dynamic operationalization [25].

3. Related work

Based on this basis, many studies, papers and scientific papers are a search for the best basis for relying on the development of a method or approach to validate the requirements of the software. But we will highlight more on the most important of these studies that based on the graphical notations as a basis for the development of the method, approach, or model and the most important results achieved and what are the weaknesses and strengthen for each one of them.

3D Visualization [9] is a method to comprehend knowledge information using 3D graphics diagrams to represent it, as data are transformed into geometric representations. Once the software application is specified, developers will define the graphical representation of the software application requirements for visualizing and animating the specification concepts in a 3D graphics world, and so validated this software by the users. The software application is described in a way that the customer can understand.

Because of software requirements validation in the earlier stages of the development process, the total effort to develop a software system is reduced. But the building of 3D graphics presentations is a difficult creation and time-consuming task. 3D graphics representation requires specific knowledge and even artistic skills. No direct relation between the developer and the customer.

Unified Modelling Language (UML) state machine model is a popular modelling tool in software engineering for specifying dynamic perspective of a software system and its interactions with the user's system through sequences of transitions. By traversing the state machine model, can be obtained from feasible transition sequences. Each feasible transition sequence represents an operational scenario that describes the desired movement and behaviour of the target software system. Each sequence of transitions derived from the state machine model can form a scenario that describes the behaviour which represents a set of situations of common characteristics that might reasonably occur. [12]

In case of Unified Modelling Language, state machine model is a popular language tool for modelling software requirements from a behavioural perspective. Can be specified the operational characteristics of a software system and its interactions with users in a state machine model. [26]

Abstract State Machines (ASMs) are formal methods and helpful to validate software system requirements. Validate software systems requirements through a simulation of the generated ASMs. Transform use case models of software systems requirements into ASM-based executable specifications and then validate these generated formal models through simulation. [27] Extend a formal language Timed Abstract State Machine (TASM) with two newly defined constructs event an observer [28].

Westmere is an automated acceptance testing tool, which generates abstract test cases and executable (UI) user interface to facilitate the software requirements validation process. Facilitate the software requirements validation process by generating abstract test cases and executable (UI) user interface. Implementing the black-box testing strategy, this strategy only looks at the abstract level of specification, without concerning the internal behaviour of a software system [29].

3.1 Concept Map Approach

A concept map as the approach is a diagram or graphical notation way that applies on knowledge topic to represents relationships between different ideas. Most concept maps depict ideas as circles or boxes (also called nodes), which are structured often in a hierarchical form and connected with arrows or lines (also called arcs). These lines are labelled with linking phrases and words to help explain the connections between ideas and concepts [13]. The concept is defined as "perceived regularity or pattern in event or object, or record of event or object, designated by a label" and is depicted as a shape in the diagram. Concept maps deepen understanding and comprehension.

3.2 Concept Map Types

A concept map can take many forms; there are four major types of concept maps. These are distinguished by their different formats for representing knowledge and information. Provided it includes concepts as nodes and illustrates relationships through lines and proposition statements in different layouts. Where multiple and integrated concepts map types to gives them a strong to be able to cover any topic. They include about seven species, there are main and special [17] [30]. Concept maps consist of two main categories: the main group of maps and the specific group, each comprising several different types in form and composition as shown in table 1.

Concept Map Types	
Main types	Special types
Hierarchy	Visual landscape
Spider	Multidimensional
Systems	Mandala
Flowchart	

Table 1. Types of Concept Maps

There are many types of maps of the concept and integrated and this is the most important characteristic of where it can cover many cases, whether simple or complex, we will reach access to the most appropriate type to express the case in the form of graphical notation understandable by most people. This gives the concept maps a high degree of flexibility.

In the software industry, we need all these types to cover all the software requirements in one or more graphic. But we may need all or some of these types depending on the circumstances of the case. In figure 7 we present a concept map containing more than one type of concept maps, illustrating the natural sciences. Although this is far from the software industry filed, it emphasizes the utility of integrating more types of concept maps to cover a specific case.

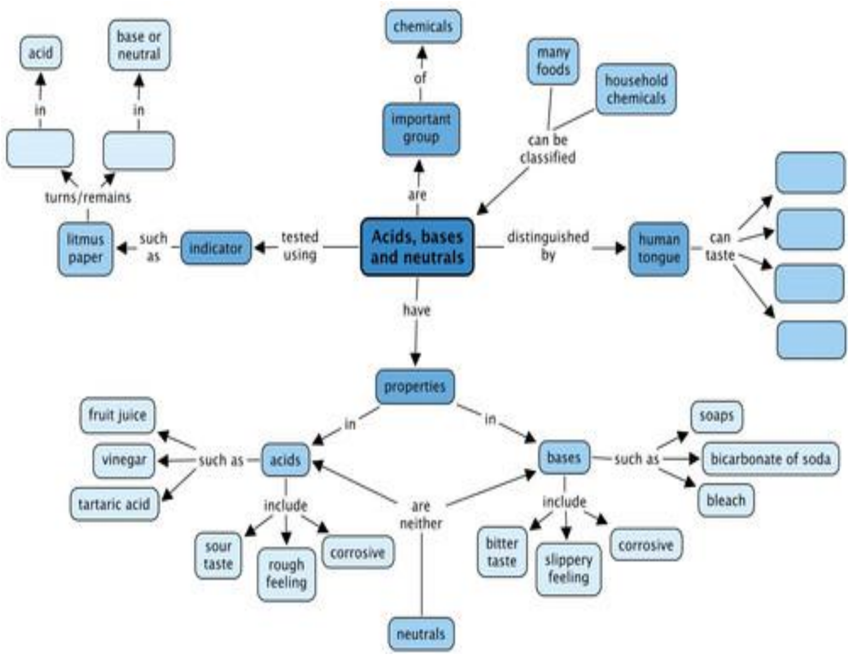


Figure 7: An examples of mixing concept map[42].

3.3 KANO's Model

Customer needs have increasingly become the focus of software enterprises in a highly competitive market. Hence, understanding and satisfying customers' needs play a vital role in software business success. Capturing the right requirements of customers and users depends substantially on how well getting them involved in system development .

Several studies [31] [32] have found that understanding the needs of users and how they operate in the context of the proposed software and in the wider context of the organizational system where the product will be installed can help identify requirements with increased accuracy and completeness and increase customer satisfaction providing more potential for the success of the software product. In the 1980's Professor, Noriaki Kano [33] presented the model known as "Kano's model" for service and product quality measurement, can apply this model to measure the degree of satisfaction with the software product as it explains how customer satisfaction would change when requirements are met by the developer. In as shown in figure 8 and explained by different studies [34] [35].

Kano's model enables one to explore the components of product or service quality that affect customer satisfaction, the characteristics of the product or service have a major role in the degree of customer satisfaction [37].

This model represents the relationship between customer satisfaction and quality requirements and classifies customer preferences into three distinct categories each of which impacts customer satisfaction with the service or product in different ways. These categories are represented in basic quality (must be), one-dimension quality (performance) and excitement quality (attractive) [36].

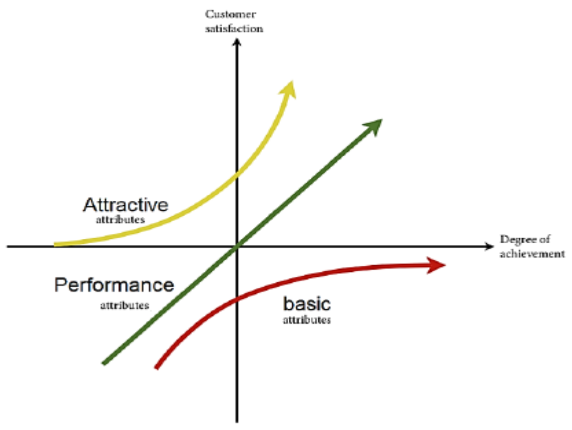


Figure 8: KANOs' Model of Customer Satisfaction [33].

Basic attributes can be defined as the must be attributes of service or product quality in terms of customer satisfaction. In other words, basic attributes are essential for all customers. Therefore, if these features are not available, it will lead to a great dissatisfaction among the customers. On the other hand, available must be featured will not lead to high levels of satisfaction see figure 8.

One-dimensional attributes (performance) customer satisfaction can be described as a linear function. Can see it in figure 8, performance attributes will lead to customer satisfaction when being available and will lead to customer dissatisfaction when not available. Customer satisfaction level increases linearly with the level of such features. Consequently, they are both enough and necessary condition for customer satisfaction.

Attractive attributes of service or product lead to high levels of customer satisfaction when it's available, and these features do not cause any dissatisfaction when not available because they are not expected by customers. These attributes not a necessary condition for customer's satisfaction.

3.4 Software Requirements Classification According to KANO's Model

Although there are a lot of models that measure customer satisfaction about a product or service, perhaps the most important of all is kano, SERVQUAL and Quality Function Deployment (QFD). The Kano's model was selected because it's a very convenient for software engineering as a general and more accurate in results than other models and the classification of factors in customer satisfaction is done into three simple categories. And can be used before or after production to measure customer satisfaction with the service or product.

This supports the basis of our paper and validation of the software requirements of the client at a very early stage with the same customer participation from the software development lifecycle to ensure the greatest possible customer satisfaction. As shown in figure 9.

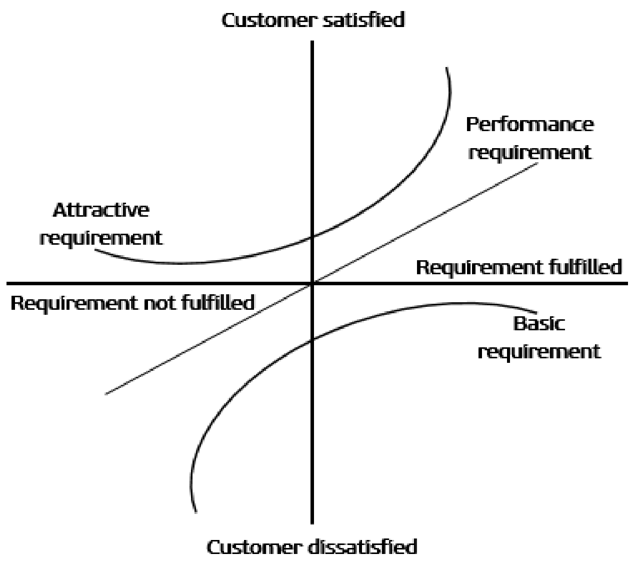


Figure 9: Software requirements classification according to KANO's

3.4.1 The Basic Requirements (Must be)

The basic requirements (Must be) are those that the customer assumes exists by the product or service, i.e. the product or service does not exist without, where apply to most functional requirements. If the customer has explicitly requested or not requested it, it must be found in the software system. As in as shown in figure 9, the entire basic requirements line is in the dissatisfaction area, and this means when basic requirements are found in the software system it will not add to the customer satisfaction level. [36] These requirements are very necessary for the customer, and the software system must fulfil them [32].

3.4.2 The Performance Requirements (Linear)

The performance requirements (Linear) also called one-dimensional, where apply to most non-functional requirements. It represents a straight line as in as shown in figure 9 that represents the positive relationship between the degree of customer satisfaction and the degree of fulfilment of these requirements in a software system. This means if such requirements are fulfilled, customer satisfaction will increase.

3.4.3 The Attractive Requirements (Delight)

The Attractive Requirements (Delight) are unexpected features for the customers and are the ones that most influences customer satisfaction.as shown in figure 9. Customers do not expect these requirements to exist in the software system, and when they do, they greatly increase their satisfaction with the software system. Entire attractive requirements line in the satisfaction area, this means that customer satisfaction is directly related to the relationship with attractive requirements.

4. Conceptual framework

In this section will present the conceptual framework of development which is dependent on both concept map approach and theory of classification of customer satisfaction factors in the Kano model.

4.2 Building components of a conceptual framework

We will build a conceptual framework by integrating and consolidating both the concept map approach and the classification of software requirements according to the KANO's Model. Therefore, a conceptual framework will take steps that are very similar to the steps of creating a concept map.

4.2.1 Concept map creation steps

The basic steps for creating a concept map (Cmap) have been widely defined and described in seven steps as shown in figure 10 [38] [39] [40]:

Step1. Define the main concept of "requirement".

Write down the main concept (requirement) for the software project, often the type of project required to be developed such as real estate applications, on-demand service applications, shopping applications, social networking applications, news applications.... etc.

Step2. Identify the key concepts "requirements".

Write key concepts each concept inside rectangular shape, this customer requirement might include: basic requirements and performance requirements.

Step3. Spatially arrange the concepts "requirements".

Sort the concepts “requirements" by some notion of priority or inclusiveness and putting concepts "requirements" that NOT understand and unclear to one side. Also put aside those that ARE NOT related to any other. The concepts "requirements" left over are the ones you will use to construct the concept map.

Step4. Create links between concepts "requirements".

Arrange the concepts "requirements" so that related requirements are close to each other.

Step5. Revise spatial arrangement.

Revise spatial arrangement for concepts "requirements" and draw lines or arrows between the relevant requirements.

Step6. Create cross-links.

Create cross-links between concepts "requirements" and write on each line the description or type of relationship between requirements.

Step7. Iterate.

If you put any requirements aside in step 3, and then go back and see if some of the requirement will fit into the concept map you have constructed. If they do, be sure to add the arrow or lines and relationships of the new requirements.

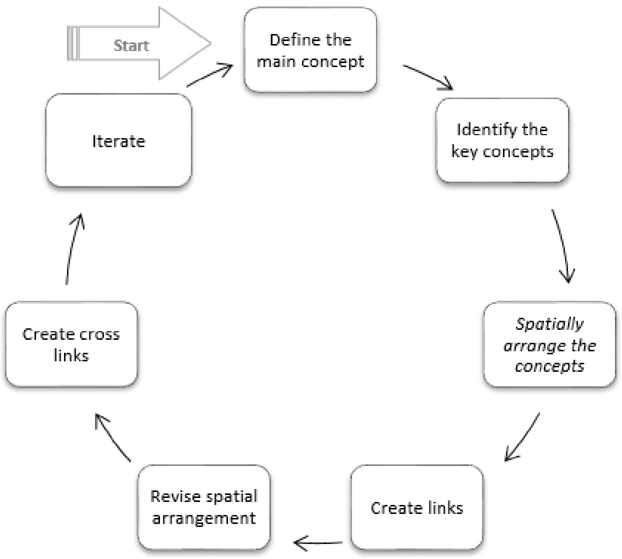

Start
Define the main concept
Identify the key concepts
Spatially arrange the concepts
Create links
Revise spatial arrangement
Create cross links
Iterate

Figure 10: Concept map creation steps.

4.2.2 Software requirements classification

Basic Requirement.

Are the basic requirements that cannot be included by the application or otherwise become without any benefit to the customer, for example, most applications must contain the case of user registration and login to the application. Without these features, the application becomes useless and lacks the permissions between users, there is no distinction between users of the application. Basic requirements should include a description of:

- The data inserted in the software application
- Who can insert the data into the software application?
- The operations performed by each page or form
- The workflows executed by the software application
- System reports or other problems
- How the application meets the applicable regulatory requirements.

Performance Requirement.

Performance requirements also referred to as software application qualities, performance requirements are as important as functional capabilities, epics, stories, and functions. They ensure the efficiency and ease of use of the entire application. Failure to respond to any of them may lead to systems that do not meet the internal needs of companies, users or the market, or that do not meet the mandatory requirements of regulatory or standardization bodies.

Delight Requirement.

Are all requirements not directly requested by the customer but must be present to ensure customer satisfaction, for example In the Gmail app, it may be great to have a smart messaging system to prioritize your behaviour, to see which messages you're interested in, and to give you first-hand messages about those messages that you always ignore. Another example in iTunes it may be great to have a cloud platform to upload the songs and audio files you always hear.

Sometimes delight requirement includes Basic or Performance Requirements not directly requested by the customer. User Experience (UX) such as usability testing, research, persons and experiences map. and Some Parts of User Interface (UI) such as layout, visual design, user guides and branding, often fall under delight requirement; the user experience differs from the user interface by responding to a question "why is this interface?", While the user's face responds to the question "what is this interface?".

4.2.3 Conceptual framework

The stages of conceptual framework development are divided into three stages: an input stage, then processing stage then the output stage.

First. The input stage consists of determining the basic requirement and agreeing on the name or type of software system required to develop such as dating applications, real estate applications, shopping applications, on-demand service applications, social networking applications, news applications. etc.

This is then divided into required parts or sub-requirements.

Second, the processing stages are done by processing these requirements and organizing them in the form of a clear diagram of the participation of the customer himself and relying on our main approach is the concept maps. According to the division of these requirements into three clear categories.

Thirdly, the output stage is the creation of graphs to validate the requirements of the software with the participation of the customer and thus ensuring the greatest satisfaction at the early stage of software development as shown in figure 11.

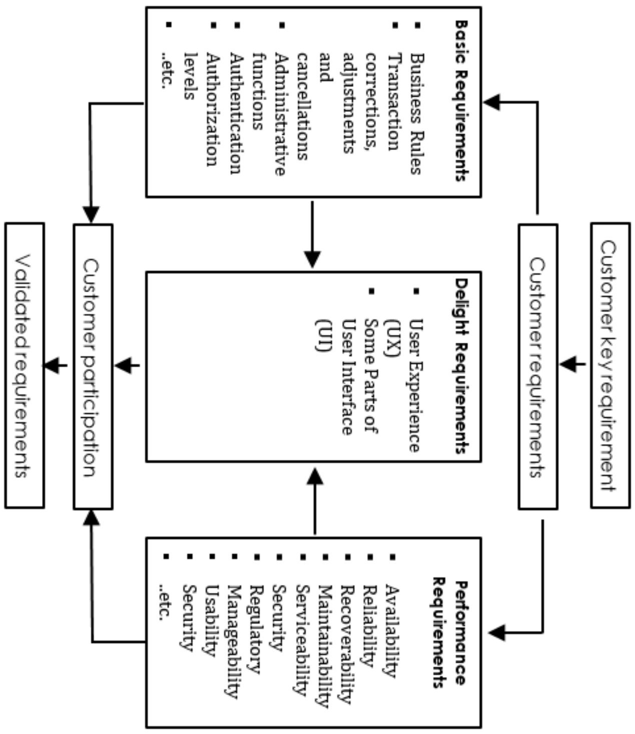


Figure 11: Conceptual framework diagram.

5. Conclusions

In this paper, we validate the software requirements by customer Involvement in the early stage of software development, because it is the best person knows what is required of this software system. We provide an adaptation of the concept map as a graphical notations approach in a conceptual framework for software requirements validation through representing the detailed relations between software requirements in diagrammatic format. This application is implemented in validation software applications, we can get the requirements in accurate and correct as high as possible. In the very early stages of software development.

REFERENCES

[1] *Ian Sommerville.: 'Software engineering', (Amazon, 2011, 9th Ed).*
[2] *MartinMonperrus, University of Lille.: 'Introduction to Empirical Software Engineering', 2015.*
[3] *Murali Chemuturi.: 'Requirements Engineering and Management for Software Development Projects', Springer, 2012.*
[4] *Alfor, M W, Lawson, J T.: 'Software Requirements Engineering Methodology (Development)', Defense Technical Information Center (DTIC), 1979.*
[5] *Richard H. Thayer and Merlin Dorfman.: 'Software Requirements Engineering' (Amazon, 1997, 2nd Ed).*
[6] *Volodymyr Bolshutkin, Claudia Steinberger, and Mykola Tkachuk.: 'Knowledge-Oriented Approach to Requirements Engineering in Ambient-Assisted Living Domain', Springer, 2013.*
[7] *Nelly Condori-Fernández, Sergio España, Klaas Sikkel, Maya Daneva, Arturo González.: 'Analyzing the Effect of the Collaborative Interactions on Performance of Requirements Validation', University of Twente Research Information, 2014*
[8] *A. Terry Bahill, Steven J. Henderson.: 'Requirements development, verification, and validation exhibited infamous failures', Wiley Online Library, 2005.*
[9] *Massilia Kamalrudin, John Grundy.: 'Generating Essential User Interface Prototypes to Validate Requirements', Researchgate, 2011.*
[10] *Teyseyre, Alfredo Raúl.: 'A 3D Visualization Approach to Validate Requirements', Repositorio Institucional de la Universidad Nacional de La Plata(SEDICI), 2002.*
[11] *Van Lamsweerde, Axel.: 'Requirements engineering: from system goals to UML models to software specifications', Digital access to libraries, 2009.*
[12] *Capers Jones.:'Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies', Amazon, 2010.*
[13] *Hong Kong Polytechnic University.: 'A Study of requirements validation with UML', Asian Journal of Natural & Applied Sciences, 2013.*
[14] *Khattri N, Miles GL.: 'Cognitive mapping: a review and working guide', Centre for Policy Research (CPR), 1994.*
[15] *Stephanie Sutherland, Steven Katz.: 'Concept mapping methodology: A catalyst for organizational learning', the experiential media methodology and theory I (EMMTI) 2005.*
[16] *A.Sharaf, M.El Defrawi, A.Elsayed.: 'Bootstrapping Domain Knowledge Exploration using Conceptual Mapping of Wikipedia', Citeseer,2013*

[17] *Alexander Feinman, Jonathan Oexner, and Subrata K. Das.: 'Using Formalized Concept Maps to Model Role-Based Workflows', Institute for Human & Machine Cognition (IHMC), 2006.*

[18] *Kumar R, Sarukesi K, Uma G V.: 'Exploring Concept Map and Its Role as Knowledge Assessment Tool (2009-2012)', International Journal of Advanced Research in Computer Engineering & Technology, 2013.*

[19] *DP Ausubel, JD Novak, H Hanesian. 'Educational psychology: A cognitive view ', spiked, 1986.*

[20] *Hongbing Kou, Philip M. Johnson.: 'Automation of Test-Driven Development Validation with Software Microprocessor', IEEE, 2016.*

[21] *Aleksandar Bulajic and Radoslav Stojic.: 'The Generalized Requirement Approach for Requirement Validation with Automatically Generated Program Code', Informing Science Institute (ISI), 2014.*

[22] *Seza Adjoyan and Abdelhak Seriai.: 'An Architecture Description Language for Dynamic Service-Oriented Product Lines' Le Centre pour la Communication Scientifique Directe (CCSD), 2015.*

[23] *JM Spivey.: 'The Z notation: a reference manual'*, (books.google, 1992, 2nd Ed).

[24] *Lieven Verschaffel, Mark Reybrouck, Marjan Janssens, Wim Van Dooren.: 'Using graphical notations to assess children's experiencing simple and complex musical fragments', sage journals, 2009.*

[25] *Luiz Marcio Cysneiros, Julio Csar and Julio Cesar Sampaio do Prado Leite.: 'Using UML to Reflect Non-Functional Requirements ', Researchgate, 2001.*

[26] *Ng Pin.: 'Applying Formal Concept Analysis in Requirements Validation with UML State Machine Model', Run Run Shaw Library, 2010.*

[27] *P Scandurra, T Yue, A Arnoldi, M Dolci.: 'Functional Requirements Validation by transforming Use Case Models into Abstract State Machines', ACM Digital Library, 2012.*

[28] *Jiale Zhou, Yue Lu, and Kristina Lundqvist.: 'A TASM-based Requirements Validation Approach for Safety-critical Embedded Systems', Springer, 2014.*

[29] *Nor Aiza Mokhtar, Massilia Kamalrudin, Safiah Sidek, Mark Robinson.: 'TestMEReq: Automated Acceptance Testing Tool For Requirements Validation', ISoRIS'14, 2014.*

[30] *'Concept Maps - Lehman College', www.lehman.edu, accessed 12 January 2018.*

[31] *J Coughlan, RD Macredie.: 'Effective communication in requirements elicitation: a comparison of methodologies ',citeseerx, 2002.*

[32] *P. Nascimento, R. Aguas, D. Schneider1, J. Souzal.: 'An Approach to Requirements Categorization using Kano's Model and Crowds ', PESC - Programa de Engenharia de Sistemas e Computação, 2012.*

[33] *Noriaki Kano.: 'Attractive quality and must-be quality ',CiNii Articles - National Institute of Informatics, 1984.*

[34] *C. Parker, and B. P. Mathews.: 'Customer satisfaction contrasting academic and consumers' interpretations', Emeraldinsight, 2001*

[35] *E. Sauerwein, F. Bailom, K. Metzler and H. H. Hinterhuber.: 'The Kano model: How to delight your customers', KFUPM, 1996.*

[36] *Balsam A.Mustasfa.: 'Classifying Software Requirements Using Kano's Model to Optimize Customer Satisfaction', Researchgate, 2014.*

*[37] Cigdem Basfirinci, Amitava Mitra.: 'A cross-*cultural *investigation of airlines service quality through the integration of Servqual and the Kano model', Elsevier, 2015.*
[38] Belinda Ng'asia Wafula.: 'Automatic Construction of Concept Maps', UEF Electronic Publications, 2016.
[39] W. Martin Davies.: 'Concept Mapping, Mind Mapping, and Argument Mapping: What are the Differences and Do They Matter?', springer, 2011.
[40] H. Hussain and N. R. Shamsuar.: 'Concept Map in Knowledge Sharing Model ', search ProQuest, 2013.
[41] 'On Differencing Object-Oriented Formal Specifications', http://www.jot.fm/issues/issue_2010_01/article5/, accessed 16 January 2018.
[42] 'An examples of mixing concept map of natural sciences '. http://www.mstworkbooks.co.za/natural-sciences/gr7/gr7-mm-03.html, accessed 19 January 2018.