Functional Requirements Categorization
Grounded Theory Approach

Richa Sharma[1] and K. K. Biswas[2]
[1]*School of Information Technology, IIT Delhi, Delhi, India*
[2]*Department of Computer Science and Engineering, IIT Delhi, Delhi, India*

Keywords: Functional Requirements, Requirements Engineering, Software Engineering, Grounded Theory.

Abstract: The ever-increasing complexity of information system is making the requirements analysis an intricate and challenging task. The challenge is further intensified in the absence of well-defined body of knowledge as to which requirements must be looked for. Though the requirements are broadly classified as functional and non-functional requirements; however, a special concern is required for functional requirements as the information system, envisioned for an organization, is expected to meet the functional behaviour of that organization. We have used Grounded Theory approach to explore the granular level of functional requirements analyzed during requirements analysis. Based on this qualitative study, we propose a classification scheme for functional requirements in this paper.

1 INTRODUCTION

Software Engineering emerged as a well-defined discipline laying down the foundations for software development in the late 1960's with the famous NATO conferences (Naur and Randell, 1968), (Buxton and Randell, 1969). It was realized that the software is easy to modify than hardware (Boehm, 2006). This realization paved way to various programming paradigms having differing viewpoints to analyse the software systems to be developed. The procedural approach of structured programming (Royce, 1970) emphasized analysing the '*as-is*' processes of the organization. On the contrary, object-oriented programming approach (Booch, 1998) brought a new dimension of data abstraction to the analysis of the system. In late 1990's, productivity concerns led to the emergence of agile model of software development that radically changed the requirements analysis related activities (Boehm, 2006). Thus, ever-increasing demand on complexity, scalability and productivity of software kept bringing changes to requirements discovery and analysis. Realizing the crucial role of requirements to the design and development of the software, requirements discovery and analysis activities came to be recognized as "Requirements Engineering" (RE) with the publication of selected papers on RE (Thayer and Dorfman, 1990) and establishment of regular conferences on RE by IEEE Society. Since then, activities involved in RE, process models for these activities, various forms of expressing the requirements and frameworks for analysing the requirements have been explored by various researchers and practitioners. However, the proposed as well as practiced methodologies to ensure consistent, correct, complete and unambiguous requirements have not been able to exhibit the three defining parameters of engineering approach, namely *repeatability*, *quantifiability* and *systematic* thought-process. An attempt to associate these parameters with RE activities calls for a fundamental question – what are the possible types or categories of the requirements that a requirements analyst or RE practitioner must engineer (analyse and validate)? Though a relatively simple and easy answer to this question would be – functional requirements and non-functional requirements (NFRs); however, this answer raises next question – are there further granular levels of each of these categories of requirements? In an attempt to find an answer to this question, we found a lot of work dedicated to NFR-study in this context with limited work for functional requirements. We are, therefore, interested in exploring granular levels of functional requirements.

We have adoped Grounded Theory (GT) approach (Glaser and Strauss, 1967; Glaser, 1978) for the purpose of our study because GT approach enables exploring those areas that have not been thoroughly researched for. Secondly, GT allows studying a phenomenon through rigorous analysis of data. Both of these propositions of GT approach makes it a suitable choice for our study. Though we are clear with our interest area but we do not want to be guided by any literature study or experiences. Instead, we want data, the requirements specification documents in our case, only to guide us.

The organization of rest of the paper is as follows: section 2 briefly discusses the need (interest) for functional requirements categorization. In section 3, we present an introduction to GT followed by the details of our study including coding and analysis of data. We present a sample of GT process application to illustrate how the results of functional requirements classification were obtained. We discuss the limitations and challenges of the study in section 4 followed by the conclusion in section 5.

2 MOTIVATION

Requirements taxonomy has been of interest to RE researchers and practitioners. We are of the view that the interest in identifying types of requirements stems from the fact that requirements only form the basis for subsequent phases of software development. Information system development and testing relies on the quality of the requirements captured. Unterkalmsteiner et al., (2014) have suggested alignment of RE and software testing taxonomy while bridging the gaps between these two phases of software development. Despite the indispensable role played by software requirements in development and testing, efforts in exploring types of requirements have majorly concentrated towards NFRs like (Chung and Leite, 2009; Slankas and Williams, 2013) except for one instance of functional requirements (Ghazarian, 2012). An empirical study by Kamata et al., (2007) on current RE supports our observation that functional requirements need an in-depth and extensive exploration to refine RE processes and methodologies.

We believe that just as NFR classification has initiated focused studies for different types of NFRs (Cysneiros et al., 2005; Liu et al., 2003; Breaux and Anton, 2008), resulting in better comprehension of them, an understanding of functional requirements at a granular level will prove helpful in bringing engineering perspective to not only RE but also to complete software development. A well-established classification scheme for functional requirements will make both requirements elicitation and analysis more focussed. Along with NFRs, the granular level of functional requirements will enable quantification of requirements. This, in turn, will bring quantified and systematic approach to other phases of software development. Though these points will require validation-studies to prove the expected benefits; nevertheless, the vital role played by requirements motivated us to conduct qualitative research for functional requirements categorization.

3 GROUNDED THEORY APPROACH

3.1 Brief Introduction

Grounded theory is a general methodology for developing theory that is grounded in data systematically gathered and analysed (Glaser and Strauss, 1967; Glaser, 1978). A central feature of this analytic approach is a general method of comparative analysis; hence, it is often referred to as *constant comparative method*. This methodology allows researchers to generate new concepts by carefully studying and analyzing data. The sources of data can be interviews, recordings, field observations as well as documents etc.

Though initially proposed for sociology context, GT has been used as a research method by several researchers like (Coleman and Connor, 2007; Crabtree et al., 2009; Hoda et al., 2012) etc. to explore and study various problems in context of Software Engineering. Crabtree et al. and Coleman and Connor have studied software process improvement using GT approach. Hoda et al. have applied GT approach to study the practices of self-organizing Agile teams.

With difference in opinions of the proposers of GT, there are two major variants of the approach, namely Glaserian (1978) and Straussian (Strauss and Corbin, 1990). Glaser is of the view that the theory should emerge from the data during analysis, whereas Strauss and Corbin emphasize systematic coding technique by listing all possible meanings of the data. Glaser suggests avoiding literature review before starting the study, whereas, Strauss and Corbin advise that there should be some literature exposure prior to the study. Strauss and Corbin support framing research questions prior to the study though Glaser refrains from doing so and advocates analysing the data without preconceived notions. Though both the views have been used successfully for different problems in Software Engineering, we found Glaserian version more appropriate for our study. First, our area of interest is such that we would like to go with the coding paradigm of Glaser – "*What do we have here?*" instead of asking – "*What if?*" (Coding approach suggested by Strauss and Corbin). Secondly, we do not have any particular research question formulation except for exploring the nature of functional requirements. The absence of literature in our area of interest also motivated us to go by Glaser's approach. We did not refer to the only related existing work (Ghazarian, 2012) until the end of study. We present the details of our study in following sub-sections.

3.2 Data Collection

GT approach considers – '*everything is data*'. Following this premise, we referred to requirements documents from different domains as source of information for our study. Software systems range from safety-critical, mission-critical to enterprise applications, web-based systems to mobile application and, considering requirements from different types of systems (population in context of GT) could possibly be bewildering. We, therefore, restricted ourselves to enterprise-wide applications that can be desktop or web-based applications like an ERP system, financial applications, and retail applications etc. in order to arrive at substantive formulation of our exploration of functional requirements.

Our study is based on the analysis of requirements specification documents of five enterprise-wide projects drawn from different domains including academics, finance and health-care. The requirements are captured in the specification documents in the form of either section-wise free-flow text or in the form of structured textual use-cases (Cockburn, 2000). Three of the documents studied are in the free-flow textual form, and the rest two documents follow structured use-case format. The details on the size and the nature of textual representation of the studied requirements documents are presented in table – 1. The size of documents expressed in the form of use-cases corresponds to the requirements stated in 'process flow', 'extended flow' and 'alternate flow' sections of a use-case. We refer to each of these documents by numbers (eg. doc1, doc2 etc.) instead of their original names for confidentiality reasons.

Table 1: Requirements Corpus Details.

Document	Type of Text	Size
Doc1	Free-flow	248
Doc2	Free-flow	798
Doc3	Free-flow	1164
Doc4	Use-case	390
Doc5	Use-case	460
Total		3060

3.3 Coding and Analysis

The analysis of data commenced with the first step of open coding. The task of open coding was carried out by the authors of this paper in conjunction with four graduate subjects who have taken course on 'Software Engineering'. Since it is not easy to code the size of data as mentioned in table – 1 individually, therefore, we took this task as group exercise. Secondly, individual coding can possibly be subjective. To remain objective in our study, we preferred to go for group-sessions while coding and analyzing.

3.3.1 Open Coding and Constant Comparison

The guiding question to code each of the requirements statements was: "*what does the requirements statement represent*?". The open codes were constantly compared statement by statement and also, document by document to allow reasonably fair understanding of open codes in terms of similarity and dissimilarity.

Constant comparison analysis resulted in emergence of categories, each having certain distinguishing properties. To illustrate how the process of constant comparison resulted in emergence of categories, let us consider an example of user-privilege category:

<u>RQ1:</u> *The system shall only allow a user with an Authorized Official (AO) role to create a new submission.*

The open codes for user-privilege category in this statement are underlined: available entities and role, concept of new submission; <u>associated privileges</u>.

<u>RQ2:</u> *The system administrator can create and activate a normal user or patient or disable a selected normal user.*

For RQ2 also, we present the underlined open codes for user-privilege category: system entities, <u>permissible functions.</u>

We constantly compared the open codes to find the category they represent. Few more codes that led to the emergence of *user privileges* category include: eligible operations, actions permitted.

Theoretical memoing accompanied the task of open coding. Memos are written records of how the codes, their relationship and agreement to an emerging category are identified. Memos also help in finding out the properties that are relevant to a particular category when theoretical coding and theoretical sorting are carried out. Memos played an important role in our case too during theoretical coding. An example of final memo of user-privilege category is presented in appendix-A.

3.3.2 Selective Coding

We moved to selective coding phase as our understanding of emerging categories and open codes gained clarity. We started selectively coding for the emergent core category after performing open coding for three of the documents listed in table 1. In the words of Glaser (1978) too, selective coding should be performed at a stage where one can "delimit coding to only those variables that relate to core category in sufficiently significant ways to be used in parsimonious theory."

Table 2: Open coding and Emergent Core Categories.

Sl. No.	Open Codes	Core Category
1.	System Entities	Entity Modeling Requirements
	Available Roles	
	Concept	
	Abstraction	
2.	Information Display	User Interface Requirements
	Information Layout on GUI	
	Look and feel of page	
	Navigation Details	
3.	Permissible function	User Privileges Requirements
	Associated privileges	
	Eligible operations	
	Actions permitted	
4.	Accessing Data on GUI	User Interaction Requirements
	Steps to manipulate data from GUI	
	Display Error/Info Message	
5.	Business Logic	Business Workflow Requirements
	Sequence of operations	
	Business Procedure	
6.	Regulatory norms	Business Constraints Requirements
	Policies/Guidelines	
	Technical Concern	
7.	Remote Communication	External Communication Requirements
	External Trigger	
	External Interface	

Selective coding and simultaneous referring to theoretical memos helped in saturating each of the core categories. Table 2 presents the open codes that helped in saturating core categories for functional requirements.

3.3.3 Theoretical Coding

Theoretical coding, though, is not necessarily required in Glaser's opinions; but we preferred to apply Glaser's recommendations on theoretical coding families (Glaser, 1978). Theoretical coding helped in gaining confidence that the emergent categories indeed are related and represent one of the meaningful and recommended theoretical codes (Glaserian family of codes). We found the 'Type' family best describing our categories. The discovered 'types' of functional requirements are described in detail in the following sub-section.

3.4 Functional Requirements Categories

We present brief descriptions of the emergent categories of functional requirements with relevant examples. These descriptions are drawn from the final memo of each of these core categories:

1) *Entity Modeling Requirements:* An organization is constituted by the entities (actors/business users), who are responsible for smooth conduct of organizational operations. These requirements correspond to the domain model of the organization. The business users are the ones who act as agents for actions or operations in an organization. The actions often make use of or, often impact certain domain-relevant concepts. These domain-relevant concepts are also modeled as entities while implementing the information system for an organization. The relationships between various business users often generate domain-relevant concepts. For example: Consider the following requirements statements representing entity-modeling requirements:

RQ1: *The system shall only allow a user with an Authorized Official (AO) role to create a new submission.*

This statement illustrates business user, namely *authorized official (AO)* and concepts namely *system, user* and *submission* as possible entities present.

RQ3: *The system initiates the allocation of courses to students based on their preferences.*

In RQ3, *allocation* is an abstract concept relating the entities – *course* and *student*.

2) *User Interface Requirements:* These requirements correspond to the organization and presentation of the information (including data input and output) on the graphical user interface which is used by the users to interact with the information system. All those statements that describe the layout of information on interface; or, flow of information from one level to another interface level belong to this category of requirements. Following requirements statements illustrate instances of user interface requirements:

RQ4: *Any entity/text on the user interface that is a link should be in blue font and underlined.*

RQ4 provides information on how an entity should appear on the user interface of the information system.

RQ5: *Home page will provide links to Forms, Resources and other tabs which are available on the ABC Web application.*

RQ5 provides information on the information flow available on home page as links for an ABC web application.

3) *User Privileges Requirements:* These requirements relate to the description of various roles played by the business users in the organization and the permissible privileges associated with that role. These requirements inform the rights of the users, and not how those rights are executed. Consider for example the requirements statement, RQ1 above:

We observe that the entity (actor in this case), user is associated with another entity: *authorized official* (AO) role and the latter one has the right to create an abstract concept: *submission*. This statement, therefore, indicates information about the privileges of business user – Authorized official (AO). Another example illustrating the privileges associated with '*system administrator'* user:

RQ2: *The system administrator can create and activate a normal user or patient or disable a selected normal user.*

4) *User Interaction Requirements:* These requirements are closely related to user interface requirements. However, their scope delves deeper to the business users' interaction with the system. For example: to create an abstract entity through user interface, the input information needs to be provided; the validation checks for the input information; any assistance, error or prompt messages, together constitute the form of interaction of a business user with an information system. We mark the statements providing such information as user interaction requirements. An instance of this category of functional requirements statements is as:

RQ6: *The system shall allow the user to edit a submission by clicking on the Facility column. The system shall allow the Facility column to be clicked only when the submission is still underway.*

The above-mentioned statement indicates how the '*submission'* concept will be edited through user interface.

5) *Business Workflow Requirements:* These requirements are representative of business rules, policies and procedures. These requirements state the business logic and rationale for flow of actions stated either in the form of use-case or free-flowing text. These business rules and policies only provide justification to the business users' behavior within the organization. Consider the following requirements statement for example:

RQ7: *Before submission a Certification Statement must be signed by AO, certifying that the information provided on the Web form is complete and accurate.*

The above statement tells that *submission* first needs to be signed by AO, along with a justification for the same. RQ7 also represents business workflow requirement in addition to describing user interaction requirements. The business logic embedded in RQ7 provides rationale for the action-flow of the entity – *Primary Submitter*.

6) *Business Constraints Requirements:* These requirements correspond to the constraints added to the information system apart from business workflow logic. Such additional constraints often arise because of organizational policy, external regulatory bodies or market regulations in which the organization is operating or possibly, technical constraints. For example: a financial system is governed by government and market policies; a healthcare system has to follow guidelines suggested by corresponding medical regulatory bodies. Following statement represents an example of business constraint requirement:

RQ8: *Since it is expected that users from all over the world will be using the system, and since Microsoft Windows is the most popular and widespread platform, it is decided that the system should be able to run on Windows XP, Windows 2000, and Windows NT desktops.*

The above-statement reflects constraints because of two reasons – first, market outreach of the information system and second, technical constraint that Microsoft Windows is most popular. One more example showing constraint due to technical reason:

RQ9: *The user must have JavaScript enabled for the message prompts to occur.*

7) *External Communication Requirements:* An information system does not exist in isolation; it often has to interact with other information systems or, process information coming in from another system or possibly, send information to other systems. All those statements that describe interaction of the information system with other systems or agents outside its scope represent external communication requirements. Consider for example the requirements statements:

RQ10: *Updates to the ABC database in the system are commonly performed via Remote Data Transfer. Remote data transfer is commonly accomplished using FTP over the Internet.*

The requirements statement, RQ10, is an example of external communication requirements where the *database* of the information system is modified by *an external trigger*.

4 DISCUSSION

4.1 Limitation

GT is said to be limited to the scope of the context under study. However, as recommended by its originators Glaser and Strauss (1967), GT is an effective tool for moving towards higher-level 'general' theory by comparative analysis of substantive theories in different contexts. Our study is also limited to the context of enterprise-wide projects. A similar study in other contexts may result in formalizing 'general' theory for functional requirements classification.

A second limitation that can be thought of with GT is that of subjective coding. We have mitigated this limitation by carrying out open-coding and memoing in group-sessions to come up with objective opinions after lot of brain-storming.

4.2 Challenges

A major challenge in most of the GT-based studies is that of data collection. However, we faced a very different challenge in our study. During the course of our study, we faced lot of dilemma while discovering the emergent categories and finding a core category. We observed considerable overlaps as well as conflicts in the requirements statements at the time of open coding. For example: the core categories: *user privileges* and *user interaction* were quite confusing in the beginning of the study. Theoretical memos played a crucial role in saturating these categories. The deliberations that went in writing memos helped in extracting the distinguishing features for each of these categories. We faced similar such confusion with business workflow and business constraints categories; the confusion was again resolved through theoretical memoing and sorting.

4.3 Comparison with Ghazarian's Study

We referred to Ghazarian's (2012) study, the only relevant related work after completing our study. Ghazarian's work is based on an empirical study with 12 proposed categories of functional requirements. Ghazarian's study reveals that most of the functional requirements space is specified in terms of small set of core requirement types, namely: data input, data output, data persistence, event trigger and business logic. Studying the description of Ghazarian's categories, we observed that these are too specific in nature and are close to the solution domain (developed code) and not the problem domain (requirements specification). Nevertheless, these categories form specific cases of the core categories emerging in our study as evident from the description of Ghazarian's categories. *For example:* data input and data input are specific cases of entity-modeling requirements. Ghazarian too has suggested that his proposed taxonomy of FRs may further be partitioned to form more specialized classes or grouped together to form generalized classes, if required.

5 CONCLUSION

In this paper, we have presented substantive study of functional requirements using GT approach in the context of enterprise-wide applications. The study has resulted in identification of seven types or classes of functional requirements for such applications. Though it may seem that these categories are evident however a systematic study of requirements specification has culminated in grounding our observations in the requirements data. We have compared our results with an empirical study conducted earlier for functional requirements space to find that generalization-specification relationship exists between our study and that work. We are confident that well-defined functional requirements categorization will prove beneficial to not just RE, but to the software development as well. Future studies may extend our substantive theory to a more 'general' theory for functional requirements.

REFERENCES

Naur, P. and Randell, B., 1968. Software Engineering Report on a conference sponsored by NATO SCIENCE COMMITTEE, Garmisch, Germany, *Scientific Affairs Division NATO*, Belgium.

Buxton J. N. and Randell, B., 1969. Software Engineering Techniques, Report on a conference sponsored by NATO SCIENCE COMMITTEE, Rome, Italy, *Scientific Affairs Division NATO*, Belgium.

Boehm, B., 2006. A view of 20th and 21st century Software Engineering, In *Proceedings of the International Conference on Software Engineering (ICSE'06)*, Shanghai, China, pp. 12-29.

Royce, W. W., 1970. Managing the development of large Software Systems, In *Technical Papers of Western Electronic Show and Convention*, Los Angeles, USA, 1970, pp. 328-338.

Booch, G., 1998. Object-oriented Analysis and Design with Applications, 2nd ed, CA, Addison-Wesley.

Thayer, R. H. and Dorfman, M. (eds.), 1990. System and Software Requirements Engineering, *IEEE Computer Society Press*, Los Alamitos, CA, 1990.

Glaser, B.G. and Strauss, A.L., 1967. The Discovery of Grounded Theory: strategies of qualitative research, Aldine Transactions, New Brunswick, USA.

Glaser, B.G., 1978. Theoretical Sensitivity: Advances in the methodology of Grounded Theory. *Mill Valley, CA: Sociology Press*.

Unterkalmsteiner, M., Feldt, R. and Gorschek, T., 2014, A Taxonomy for Requirements Engineering and Software Test Alignment, *ACM Transactions on Software Engineering and Methodology*, 23 (2).

Chung, L. and Leite, J.C.S. do Prado, 2009. On Non-functional Requirements in Software Engineering, In *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhari, P. Giorgini and E. Yu (eds.), Springer, pp. 363-379.

Slankas, J. and Williams, L., 2013. Automated Extraction of Non-Functional Requirements in Available Documentation, In *International Workshop on Natural Language Analysis in Software Engineering (NaturaLise), co-located with ICSE-2013*, pp. 9-16.

Kamata, M. I., Yoshida, A. Y., Yoshida, H. and Aoki, N., 2007. Figure Out the Current Software Requirements Engineering - What Practitioners Expect to Requirements Engineering? In *Proceedings of 14th Asia-Pacific Software Engineering Conference (APSEC 2007)*, pp. 89-96.

Cysneiros, L.M., Werneck, V. and Kushniruk, A., 2005. Reusable Knowledge for Satisficing Usability Requirements, In *Proceedings of 13th IEEE International Requirements Engineering Conference*.

Liu, L., Yu, E. and Mylopoulos, J., 2003. Security and Privacy Requirements Analysis within a Social Setting, In *Proceedings of 11th IEEE International Requirements Engineering Conference*, pp. 151-161.

Breaux, T.D. and Anton, A.I., 2008, Analyzing Regulatory Rules for Privacy and Security Requirements, *IEEE Transactions on Software Engineering*, 34(1), pp. 5-20

Coleman, G. and Connor, R.O., 2007. Using grounded theory to understand software process improvement: A study of Irish software product companies, *Information and Software Technology*, 49(6), pp. 654-667.

Crabtree, C. A., Seaman, C. B. and Norcio, A. F., 2009. Exploring language in software process elicitation: A grounded theory approach, In *Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement*, pp. 324-335.

Hoda, R., Noble, J. and Stuart, M, 2012. Developing a grounded theory to explain the practices of self-organizing Agile Teams, *Empirical Software Engineering*, 17(6), pp. 609-639.

Strauss, A. and Corbin, J., 1990, Basics of Qualitative Research: Grounded Theory Procedures and Techniques, *Sage Publications*, Newbury Park CA.

Cockburn, A., 2000, Writing Effective Use Cases, *Addison Wesley*.

Ghazarian, A., 2012. Characterization of Functional Software Requirements space: the law of Requirements taxonomic growth, In *Proceedings of 20th IEEE International Requirements Engineering Conference*, pp. 241-250.

APPENDIX

Final Memo: User-privilege Requirements

Privileges indicate the rights enjoyed by some authority. While privileges are associated with the roles that represent some authority, but these roles reflect the concepts of the domain and roles should be considered along with domain model. Roles are noun-concepts, whereas, privileges are action-concepts – emphasizing the actions under the purview of the authoritative role, i.e the description of the role. How these privileges or actions are carried out is a different consideration.