CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE REVIEW

The literature study carried out in this chapter can be broadly classified into:

- Literature review on Requirements Engineering
- Literature review on defect tracing to requirements
- Literature review on Information Retrieval for Requirement Engineering
- Reviews on Non Functional Requirements

2.2 LITERATURE REVIEW ON REQUIREMENTS ENGINEERING

Pete Sawyer et al (1999) classified the practices as basic, intermediate, and advanced. Basic practices represent fundamental measures that protect a repeatable process. Though REGPG includes a CMM-like improvement framework, it was not meant for accreditation. Industry’s enthusiasm for SPI, lead to improving trend for internally driven SPI programs where accreditation was not the goal. The authors tried to exploit it by adopting an improvement framework that helps orient requirements process improvement with other SPI initiatives. Hence, REGPG tries to initiate consolidation of industrial Requirements Engineering practices and not prescribe a standard.
Crenguta Bogdan (2004) suggested the idea of a concern facet concept by centering thoughts on software processes and software system quality worries. To prove this the paper defines a group concern facets relating to software process and product quality and use them to develop software systems. Examples are given “from around” the architecture concern. In addition to scope, objectives and requirements, architecture concern is of paramount importance.

Axel Van Lamsweerde et al (1998) after listing the inconsistency types which arose in a requirements elaboration, delineated and explored their inter relationship and sought specific conflicts between goals and requirements either along or among stakeholders views. Many examples are used in the paper to highlight the relevance of the ideas and techniques put out. This paper suggests a formal outline to clarify different inconsistency types which arise in a RE process; special emphasis is on one inconsistency type which has not had much attention so far in literature. A key issue in this dissertation is to divergences early in the RE process i.e. at the goal level. The quicker divergence analysis is generated more the freedom to resolve it.

Ddembe Williams et al (1999) suggested a process simulation modeling framework for improving RE effectiveness founded on a system dynamics model to measure its quality relating to adherence to schedules and cost. The paper described the factors included in starting RE process model with improvements based on issues generated during development. Basic results of a pilot study suggests that a process model tool helps understand the RE process. This helps attain process effectiveness through learning/training aided by system dynamics tools. The model’s value both in theory and learning/training situations was emphasized. It was suggested that could be a framework to develop knowledge on RE process management.
Alistair et al (1998) invented a method and software aid tool for scenario-based RE which included use case approaches to object-oriented development and which is illustrated with a case study. The method begins with acquisition and modeling of a use case which is compared with a slew of abstract models representing application needs. Each model is connected with a host of generic requirements of its class where generic requirements can be reused by finding out to which class the use case belongs. Scenario paths are automatically created from use cases, following which exception types are applied to normal event sequences to locate abnormal events due to human error. Generic requirements are linked to exceptions for knowing ways to ways of deal with human error and related system failures. Scenarios are corrected by rule-based frames that detect problematic event patterns. The tool also suggests necessary generic requirements to deal with issues as they arise.

Ana Moreira et al (2005) suggested a multi pronged way to separate concerns in RE as well as a trade-off analysis of requirements specification from various viewpoints. The paper aimed rid the idea of a fixed functional base with regard to trade-offs from where Non-Functional concerns are usually observed, analyzed and solved which in turn led to an architecture that is misaligned with system requirements as related choices would have been driven by Functional Requirements. Also the related trade-offs were generally ignored. The suggested multi-dimensional approach corrects this problem by treating Functional and Non-Functional, as equals. This allowed the Requirements Engineer to analyze crosscutting functional properties influence on requirements in the system and also helped trade-off analysis generated from this and through negotiations with stakeholders. One main feature was the idea of a Meta concern space, a catalogue of Functional and Non-
Functional concerns which manifests repeatedly in various software systems. Till date RE approaches are mainly two dimensional in such analyses.

Georgia Sousa et al (2004) used a few use-case driven activities of the Unified Software Development Process to provide the reasoning and separation of crosscutting concerns from requirements artifacts to design artifacts. This is illustrated through the study of an Internet Banking System and was the first step to total adaptation of the Unified Software Development Process to provide separation of crosscutting concerns in workflows.

Castro et al (2002) suggested a methodology named Tropos, founded on intentional and social concepts, and inspired by early requirements analysis. The proposed methodology has 4 phases like early requirements, late requirements, architectural design and detailed design, where behavior of each architectural component is detailed. Early results reveal that this methodology accepts Agent Oriented Programming Environments. Tropos is willing to accept these concepts, necessitated by requirements modeling research, to detail requirements and system design models so that the semantic gap between them can be narrowed.

Summing up the themes discussed at REFSQ workshops between 1994 and 2003, Andreas et al (2004) hinted at topics for further research. Research topics discussed emphasized RE in an enterprise-wide perspective where closer links between business and software architectures was needed. In this anniversary paper, the three members of the REFSQ Advisory Board presented an overview of the most important themes that were in 10 years of REFSQ workshops.

- Actors and goals
• Commercial off-the-shelf (COTS) systems and market-driven development
• Non-Functional Requirements
• Scenarios and use cases
• Traceability
• Tools

The discussion of research topics emphasized RE in an enterprise-wide perspective where a tighter coupling between business and software architectures is needed.

A procedure to use a goal-oriented strategy to integrate non-functional and functional requirements was presented by Cysneiros, et al (2001). This framework integrates NFRs into ER and OO models. This goal-oriented conceptual model is capable of representing non-functional aspects like confidentiality, performance, ease of use and timeliness. The suggested strategy deals with NFRs integrating them into conceptual models which are simple not requiring a huge investment to be practical. The procedure is anchored by a lexicon - the Language Extended Lexicon (LEL) - to integrate NFRs into ER and OO models, achieved through the use of lexicon symbols to construct both a NFR graph and conceptual model (through use of either ER or OO model). The study revealed that NFRs when handled from software development genesis and integrating this knowledge with functional and conceptual models resulted in cost savings and increased customer satisfaction. The proposed framework was validated by two case studies of which, one is very large. The case study results illustrates that early phase goal modeling could lead to a more productive and total modeling activity.
Joachim Karlsson et al (1997) developed a cost value based analytical tool to prioritize requirements which then ranked requirements according to their value to customers/users and their implementation cost. This was successfully implemented in two telecom software development projects.

Sjaak Brinkkemper et al (2004) addressed industrial research needs in requirements management, but restricted it product software. Product software, also called commercial software/standard software, is a new research area to investigate the issues when developing/implementing generic software for a market. The authors presented status qualification of research results in requirements management of product software, positioned in a framework based on the CMM-I process area of requirements management. For some aspects of requirements management there is an urgent need for research activity. Key research needs are release planning, process evidence, tracing and tracking, and requirements metrics. The authors plead to stop using the term RE and to use product management instead.

2.3 LITERATURE REVIEW ON DEFECT TRACING TO REQUIREMENTS

Carl A. Gunter et al (2000) planned a framework for key artifacts, its attributes, and relationships at a median level to delve into substantive properties robustly. The model was developed on five artifacts categorized into groups pertaining to the system and to the environment. These include domain knowledge which gives presupposed environment facts; customer’s needs from the system thought of in terms of its impact on environment; specifications to build a system that meets these needs; a program which implements them on the same platform and conversely a programming platform that becomes a foundation to meet customer requirements/specifications.
Islam et al (2004) illustrated the use of GOPCSD tool to develop aspect-based process control applications. The GOPCSD tool has goal driven requirements analysis concepts and was adapted from KAOS method to address process control systems. GOPCSD helps the user to express productivity (throughput), liveness, safety, economy and operational aspects as high-level goals. The requirements completeness/consistency checks can combine different aspects in like manner to remove requirements bugs that lower effort required by systems engineers as not requiring expertise in the process control field. The way different aspects are separated makes it easy for the user to decide which goal to weaken/strengthen when solving conflicts/incompleteness. Additionally it helps increase modifiability, traceability, and augmentability of the entire requirements model of the process control application.

Len Bass et al (2004) highlighted a system where quality requirements act as quality scenarios progressing to design. This system used quality attribute reasoning frameworks to find out if requirements could be met, identify places in an emerging design that prevents them from being attained and guiding an architect as to the correct methods to solve the problem.

Michael Jackson (2004) explained how to understand problems. It leads to a need for composition, both in the problem world and in the solution world. Inevitably, aspect-oriented programming has focused on solutions; ‘early aspects’ aims to focus on problems. The goals of aspect technology would be clarified by such analysis, and the aspect technology may in turn offer some power in understanding and implementing the compositions.

Miguel – Angel Sicilia et al (2004) approached the problem of specifying imperfect information in Software Requirements and domain models as cross–cutting concerns, with examples of mapping such concerns
into aspects at the design and implementation stages. A tentative concern space analysis for information imperfection has been described, along with some examples of the mapping of concerns to specific aspect-oriented design options. Jones et al (1996) described some of the special difficulties currently encountered in the early stages of developing systems.

### 2.4 LITERATURE REVIEW ON INFORMATION RETRIEVAL FOR REQUIREMENT ENGINEERING

Elisa Baniassad et al (2004) invented the Theme/Doc approach to look at relationships behaviors in requirements documents to separate aspects in them. This helps a developer to find out the “aspect” and “base” elements. Theme/Doc views also provide a feature-related requirements set which in turn can be mapped to Theme/UML models. Theme/Doc and Theme/UML comprise the theme approach. The case study proved this to be effective in identifying requirements aspects and the functionality to enhance approach scalability.

Vinunzo Gervasi et al (2004) applied statistical lexical analysis techniques to a large corpus to characterize main topics and trends which emerged from ten years research in this area. The results provided an historical perspective (and some lessons) on the genesis of favored research topics, as well as suggestions for the future, in particular about topics which were not adequately addressed so far. This paper aims to investigate trends and topics in ten years of REFSQ papers objectively and quantitatively. From the results of the study, it appears that REFSQ has moved - in the last 10 years - from a specialized workshop with a focus on quality models for requirements engineering, to a general RE conference with a workshop discussion model. REFSQ is not delinked from the more ample RE community, and has been traversed at times by sudden peaks of interests for
emerging (or re-emerging) topics. Core issues have enjoyed popularity among researchers contributing to REFSQ.

Johan et al (2005) suggested semantic similarity on the basis of lexical features assuming that customer wishes and product requirements were the same if they used the same terminology. In a RE context this is an assumption, as language tends to be more precise than in literary text. Also customer wishes and product requirements both refer to the same domain. When submitting requirements to an automated process to establish linkage an imagined support system first performs several internal preprocessing steps. The evaluation results show the recall curve for the top lists of suggested BRs for each MR. For a reasonable top list size of 10 reached, a recall of 51 percent is good considering the pragmatic approach. It was found that for 690 of the BRs, the recall rate would be 100 percent using a top list size of 10. This indicates that every related MR for each of the BRs would be found within a top-10 list. The 690 BRs link to 1,279 MRs, giving an average of 1.85 MRs per BR. But not to exaggerate the gain, assume that manually, one search term is enough to locate all links for one requirement. This approach shows a promising step toward well-engineered systems to aid large-scale RE in companies that rely on communication in natural language.

2.5 REVIEWS ON NON FUNCTIONAL REQUIREMENTS (NFR)

Mairiza et al (2010) presented results of an extensive/systematic analysis of extant literature over three NFR dimensions: (1) definition and terminology; (2) types; and (3) relevant NFRs in different system types and application domains. It describes two different perspectives to consider NFRs. A comprehensive NFR types catalogue as well as top five NFRs frequently used is presented. 252 types of NFRs were identified of whom 114 are NFRs which were specifically discussed with respect to system quality. Of these, performance, reliability, usability, security, and maintainability are five, most
frequent NFRs listed in a NFRs catalogue. Mapping between NFRs and various system types and also between NFRs and application domains were presented as the paper’s original contribution. The most common NFRs among five system types (real time systems, safety critical systems, web systems, information systems, and process-controlled systems) considered in the study include performance, security, and usability. Performance and usability requirements are two NFRs considered in almost all application domains (seven out of eight application domains).

Ameller et al (2012) presented findings of an empirical study based on 13 interviews with software architects wherein questions like who decides NFRs, types of NFRs that matter to architects, their documentation, and how NFRs are validated was addressed. The results are contextualized with existing works. The study concluded that architects mostly elicited NFRs; that software architects considered non-technical NFRs as relevant and technical; that software architects were happy with NFR fulfillment. NFR elicitation is iterative; were not often documented; just a few NFR types underwent validation; that the two most important technical NFRs were performance and usability; that software architects didn’t want automatic NFR-based decision-making tools but were willing to accept architect-driven tools.

Daniel Gross and Eric Yu (2001) suggested a methodical treatment of NFRs in pattern descriptions when applying them during design. This method organizes, scrutinizes and refines Non-Functional Requirements, and guides and supports when applying patterns in software system design. Here a represented solution is used, structures with UML notations, Applicability of Patterns during Design, Patterns with Alternative Implementations, Domain-Dependent Specialization, Improved Representation in Pattern Languages, Better pattern Retrieval/Knowledge-Based Support, NFR Goal Graphs allowing design pattern reuse, NFR Goal Graphs forming the base for
Hyperlinked Textual Representations and Linking High-Level Business Goals to System Requirements are studied. The paper ends with the idea that the NFR-driven approach and pattern approach complement each other. The patterns approach requires a route to connect to requirements, while the NFR approach needs to consolidate high quality solutions. The NFR approach and pattern approach are goal driven and solution driven respectively. One is top-down and the other bottom-up. As most design decisions are obtained in the middle it is suggested that both be combined.

Cleland-Huang et al (2007) described a technique to automate NFR detection/classification in relation to properties like security, performance, and usability. The approach detects/classifies stakeholders’ quality concerns across requirements specifications with scattered and non-categorized requirements, and across freeform documents like meeting minutes, interview notes, and memos. The proposed classification algorithm is evaluated for efficiency through reporting many experiments based on 30 requirements specifications developed by DePaul university MS students as term projects. An iterative approach is introduced to train/retrain a classifier to detect/classify NFR in datasets dissimilar to initial training sets. Though NFR classifiers cannot detect all NFRs, it supports an analyst in the error-prone task of manual NFR discovery, and further is quickly used to analyze large and complex documents to search for NFRs.

Isabel Brito et al (2004) constructed two ideas (i) integration of catalogues to identify specific concerns and (ii) improve composition rules through a casual definition of new operators. The specialty was inclusion of catalogues like the NFR framework to identify/specify concerns and refinement of composition rules using newly inspired LOTOS operators. The discussion of research topics emphasized RE in an enterprise-wide
perspective where close links between business and software architectures was required.

Barbara Paech et al (2004) reviewed state-of-the-art in NFR Engineering and defined the plans for further NFR research. State of the art review was split into two parts. To begin with criteria for evaluating state-of-the-art literature was available according to requirements. For the former identified requirements on a NFR method, is the ideal NFR method Non Functional Requirements Engineering (NFRE). The author defines requirements on a NFR method, compares it with current approaches and sketches ideas on filling gaps between present methods and requirements.

Bubenko et al (1994) detail an extended requirements and information modeling paradigm, based on interrelated meta-models which reflect the enterprise, its objectives, NFR on the system, re-use of existing specification components, and a model for managing/tracking specification development process. The transition from informal to formal requirements constitutes a conceptualization activity within which a developer might make use of domain knowledge partly expressed in the enterprise model, and partly in existing requirements specifications. Reflecting back from formal to informal requirements is a process of validation which may take a number of different forms including prototyping, and explanation as to the decisions made in producing a requirements specification. The paper believed that model types introduced in F3 led to an improved RE paradigm.

Abade et al (2012) extended model refinement idea to non-functional software properties and proposed a stepwise refinement framework with conformance checking. This is between abstract and concrete descriptions of system model using model transformation with this approach being extendable to all quantitative and quantitative non-functional properties. The proposed development process brings two techniques used to ensure
model-driven software quality together. They include non-functional engineering and software refinement. This approach’s novelty is integrated refinement view of non-functional properties based on the standard MDA description. Using sophisticated NFR language descriptions like RDSEFF formalism of Palladio Component Model or CQML+, developers can predict NF in the initial stages of development. They also check to see if the final product meets NFRs and whether other suitable measures are required.

Venkata Durga Kumar (2009) created a user friendly and intelligent tool to assist Software Engineers in the analysis and design stages for extracting Software Requirements and develop different UML representations. The key feature of the proposed tool is the automatic generation of classified requirements, Use Case Diagram, and Object Interaction Diagram (OID) from a text based case or voice which is first converted to text. The paper also discusses existing modeling tool issues and implementing a new modeling tool using speech reorganization tool, and an artificial Intelligence concept. Using speech recognition-based language tool software demonstrably improves the mean analysis abilities of a heterogeneous group of developers.

Liu et al (2010) proposed a NFR modeling approach analyzing impact on design and complementing the original design models for MDD methods. First, existing UML models of an application's functional properties are considered as NFR modeling process input. And with a repository, NFRs are refined and operationalized. Then based on analysis results, modeling implements them with the original UML models being augment in reverse by NFR modeling results. A meta model is proposed in particular to model new constructs and facilitate integration. With this approach, an integrated design model is obtained which considers both functional and NFR. The proposed
approach’s feasibility is illustrated with a design example of a simplified credit card system.

Mylopoulos et al (1992) proposed a comprehensive framework of NFR use during a development process. The framework includes 5 basic components which represent NFR with regard to interrelated goals: A set of goals to represent NFRs, design decisions and arguments in support of or against other goals, a set of link types for relating goals/goal relationships to other goals; a set of generic methods to refine goals into other goals, a collection of correlation rules to infer potential interactions among goals and a labeling procedure that determines how a NFR is addressed by a design decisions set. The framework’s effectiveness is provided through the study of information systems accuracy and performance requirements.

Adam Steele et al (2006) proposed an automatic speech recognition technique to capture NFR spoken by stakeholders at meetings/interviews during requirements elicitation. As system qualities related statements like security, performance, and portability are usually scattered throughout functional need statements, the ability to “listen in” on a conversation and correctly capture such statements into a single view is helpful. The suggested approach enhances but doe not replace existing elicitation methods where stakeholders describe their needs directly. Training a speech detection tool to recognize individual users consumes time while speech detection for un-enrolled users is very tough. The proposed approach uses context-free grammar to boost recognition accuracy, segment stakeholders’ utterances and then classify recognized statements based on quality. NFR classification accuracy is limited not only by completeness and correctness of indicator terms, but also by through underlying speech recognition accuracy.

Michael Nagel et al (2010) suggested using mobile devices as recording infrastructure which helped users to bridge gaps between informal
communication and formal project model elements. The mobile recording application establishes links between a central audio repository conversation recording and project model elements in a project management and CASE tool. Project participants use application to capture audio and context, create work for other participants, engage in asynchronous audio discussions and review project artifacts.

Tsadimas et al (2009) proposed a model-based approach which emphasized NFRs exploring the NFR effects on system design process. As system architecture definition is related to system performance, NFR plays a significant role during information system design. To facilitate designers to effectively define/handle requirements during architecture design, many system views are proposed, each focusing on discrete design issues and satisfying varying requirements. A consistent requirement model is defined representing how NFRs are related between them and to system components to form overall system architecture. Enterprise Information System (EIS) Architecture design is generally performed through proper integration of EIS components already defined by other stakeholders efficiently. Hence, NFRs require emphasis. Provision of a well-defined model representing NFRs and relations between them in a separate view improves system designer’s perception of specific design decisions which might affect others. This is because they are practically represented through derived requirements.

SysML is the modeling language that enables requirement definition and can be extended formally. Though SysML provides requirements description, this is supported abstractedly. NFRs and particularly performance requirements are extensively described using quantitative parameters during system design, while their derivations from each other must be expressed quantitatively. SysML requirement entity use is heavily extended to provide this functionality. Requirement derivation process is also discussed. A case study
where proposed concepts are applied in practice while redesigning the legacy system of a large-scale organization is also suggested.

Ilan Steinberg et al (2010) presented a system which acquires and classifies users’ hand gestures from images/videos. Using inputs from low resolution off-the-shelf web cameras, the proposed algorithm identifies the depicted hand gestures location and shape and classifies it any predefined gestures. The method initially applies image processing techniques on images to cancel background and noise effects on image; it relevant features are extracted for classification and finally gesture features are classified through the use of a multiclass Support Vector Machine classifier. The algorithm is robust and operates well on several different backgrounds, lighting and noise conditions. The proposed method achieves an average 97.8% accuracy in many cases and is suitable for real-time and offline classifications.

Casamayor et al (2010) suggested semi-supervised text categorization for automatic NFR identification/classification. Hence, a minimum requirement number, possibly identified by a requirement team during elicitation, enables learning an initial NFRs classifier which can identify type of further requirements in an iterative process correctly. The proposed approach aims to integrate into a recommender system to assist requirement analysts/software designers in architectural design process. NFRs detection/classification is performed through semi-supervised learning techniques use. Classification is dependent on a lower number of categorized requirements by taking advantage of knowledge provided by uncategorized, like certain text properties. The learning method uses user feedback to upgrade classification performance. Semi-supervised approach led to accuracy rates above 70%, which was very high and more than the results obtained with supervised methods using standard collections of documents. Empirical evidence revealed that semi-supervision required minimum manual
effort to label requirements than fully supervised methods. This could be further improved based on analysts feedback. The proposed approach outperforms previous supervised classification proposals and could be further enhanced by analysts feedback exploitation.

Cleland-Huang et al (2006) introduced an information retrieval based automation approach for NFR detection/classification. Early NFR detection is useful as it helps consideration of system level complaints and ensures incorporation of the same into early architectural design as against being refactored later. Candidate NFR is detected in structured/unstructured documents and includes requirements specifications with both scattered and non-categorized NFR, and also freeform documents like meeting minutes, interview notes, and memos containing stakeholder comments documenting their NFR related needs. The classification algorithm is described which evaluates effectiveness on an experiment based on fifteen requirements specifications developed as term projects by De Paul university’s MS students. An additional case study is described where the approach classifies NFR from a Siemens Logistics and Automotive Organization obtained large free form requirements document.

Wehrmeiste et al (2007) suggested use of aspect orientation in design and analysis of Distributed Embedded Real-time Systems (DERTS) which have several requirements directly related to NFR, their main characteristic. This includes to orthogonal properties, conditions, and restrictions spread out over the system. As pure object-oriented methods fail to successfully address crosscutting concerns, new technologies, like aspect orientation, are used to fill this gap. The proposed DERAF - Distributed Embedded Real-time Aspects Framework is an extensible and high-level framework (i.e. implementation-independent) to handle NFR of DERTS at early design stage. DERAF combines use RT-UML aspects with plans to
separate handling of Non-Functional from Functional Requirements in Model Driven Design of DERTS. DERAF, the proposed method was tried out on an Unmanned Aerial Vehicle (UAV).