International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184 Volume 2, Number 3 (June 2013), pp. 99-112

http://www.meacse.org/ijcar

Software Requirements Elicitation Tools for Service Oriented Architecture: Comparative Analysis

Qasem Nijem
College of Computer Sciences and Engineering
Taibah University
Al-Madinah Al-Munawarah, KSA

Introduction

Software requirements are specification of what should be implemented; they are descriptions of how the system should behave, a system property or attribute, or they may be a constraint on the development process of the system [1]. Requirements are specifications of the services that the system should provide, the constraints on the system and the background information that is necessary to developing the system [2]. Finally, a requirement is a singular documented need of what a particular product or service should perform [3]. The latter definition is applicable in system engineering and software engineering. One can notice that the previous definitions are defining software requirements as specifications that cover system's behaviour, attribute, capability, characteristic, or quality of a system for a user to utilize and benefit from, satisfying planned or expected objectives.

Keywords

Software Engineering, Requirements Engineering, Non-Functional Requirements, Functional Requirements, and Service-Oriented Architecture

Importance of Software Requirements

Collecting software requirements and analysing them in a professional manner is a key element of information systems and information technology projects. Many software projects have failed because their requirements were poorly collected and analysed [4]. In an empirical study [5] on requirements elicitation subject, the researchers came up with the following results:

1- System requirements engineering problems are universal and very significant indeed,

2- There are several types of errors discovered during SW project development phases. The most expensive type is requirements type since the cost of fixing it could be up to 100 times the cost of fixing an implementation error,

3- 60% of errors in critical systems were the results of requirements errors,

4- A survey of 12 CMM-certified companies in 2002 came up with the result that out of 268 development problems cited almost 50% were requirements problems.

Defects introduced in requirements elicitation have an impact on all other development and deployment activities [6]. Errors encountered due to not fulfilling software requirements are typically found only after the testing phase and are mainly found after deployment, and tend to be expensive and difficult to correct [7]. Late correction of requirements errors could cost up to 200 times as much as correction during requirements engineering [8]. Reworking requirements defects in most software development project may cost 40-50% of the total project cost [9] while some other estimates go up to 80% of the project cost [8]. To great extent, incomplete and changing requirements are considered as mean reasons for projects failure (in term of missed deadlines, exceeded budget, and/or not meeting quality expectations), those two reasons cover the majority among other reasons that cause project to fail [10]. With respect to non-functional requirements (NFRs) power of effect, there is a recent research [11] that highlights the fact that NFRs are affecting all levels of software life cycle activities, so they should be identified at the analysis phase and their elicitation must be accurate and complete.

The importance of this research is to come up with a framework for NFRs elicitation for the purpose of improving software quality in the service-oriented architecture (SOA) by advising the service customers on how to select the best web service based on its NFRs.

NFRs elicitation is considered in few process models that include architectural design as a stage in the software development process [12] such as the practice adopted by the Rational Unified Process. In this research, NFRs elicitation process is seen as part of the analysis phase and that elicited NFRs need to be evaluated and tested through all subsequent phases. This opinion is supported by [12] in which it confirms that NFRs affect mostly the system architecture and they are considered early, at the moment of formulating the requirements for the system. Previous researches have emphasized that the lack of scientific NFRs elicitation approach will definitely lead to mismatch against functional requirements [13]. Such problem can easily result in projects that will take more time to be concluded, as well as bigger maintenance costs as well as time waste [14]. NFRs have been observed to be frequently neglected or forgotten in the software development process in spite of the fact that NFRs plays an essential role in the success or the failure of the software systems especially heavy quality-oriented systems [11]. As a result of ignoring NFRs, serious problems have been reported during the development of software systems such as extra cost, schedule overruns, software systems discontinuation, incompleteness of requirements, and dissatisfaction of software systems' clients [15]. [6] states that the consequences of neglecting NFRs are frequently more severe than the consequences of functional requirements' omissions. NFRs are also known as quality requirements [16, 7] and unlike functional requirements, NFRs state constraints to the system as well as particular notions of qualities a system might have such as, but not limited to, accuracy, usability, safety, performance, reliability, and security. Hence, we can say that while functional requirements state "what" the system must do in term of services and functions delivered to the user, NFRs state (and may strongly influence) "how" the system must accomplish the "what" [13]. As a result, NFRs are always linked to a functional requirement and influence architectural and design decisions [7] since they jointly contribute to the quality product attributes. As an example a computer program may run on text-based command line tool such as DOS, graphical user interface

environment as MS-Windows, batch mode, or as a web service through some internet browser. The actual functionality of the program will continue to be same but the external interface with the users may differ significantly and what NFRs are delivering is the "how-feel" of the program and its constraints, limitations, and environment.

Non-Functional Requirements Elicitation

NFRs are not systematically gathered even with emerge of new software advanced tools and software standards that guide and recommend developers and users to plan for clear, precise and correct NFRs [11, 18]. In addition to NFRs elicitation difficulties, testing NFRs is not in the minds of people who define the NFRs [19]. It is widely known in software engineering discipline that decisions on NFRs determination affect mostly the software system structure and architecture [12]. Defining unambiguous NFRs of a system requires a lot of accuracy and expertise.

Why non-functional requirements are neglected during elicitation?

Neglecting NFRs is counted as one of the top risks of requirements engineering because of several reasons such as that they are very context dependent, they are vague, and interdependent due to their global nature as discussed in REFSQ'04 [20]. There are several reasons for not well-addressing NFRs by the current approaches implemented in the software requirements domain; the reasons are diversity, dependency, and conflicts.

First, NFRs have high level of diversity between SW requirements domains which makes the process of deciding on NFRs difficult for the development team or software engineers and the rest of stakeholders [15].

Second reason is the dependency of NFRs on the selected infra-structure components (SW and HW) that constitutes the design solution properties. Deciding the network topology (Star, Ring, Bus), data transmission (wired or wireless), cable type (fibre optic, co-axle, UTP), protocols configuration have main role to play during the NFRs elicitation process [21].

Third reason is the likelihood of having conflicts between NFRs which is very natural. An example on the conflict between security and performance, encryption algorithms have caused negative effect on the performance of the network as stated by [18]. [22] has proposed a technique that mitigates the conflict between maintainability and performance, the technique uses heuristics-based algorithm that has improved the computer system performance during testing software quality. Very few NFRs models handle the conflicts within diverse NFRs during the architectural design stage [12]. One approach to mitigate conflicts within NFRs is to make these conflicts explicit; this should include derived interdependencies as advised in REFSQ'04 [20].

Figure 1 shows a classical security trade off between security and functionality in Microsoft operating systems [23]. Windows XP operating system consists of 40 million lines of code, and end user applications are becoming equally complex- if not more- when systems become this large, bugs cannot be avoided [24]. Figure 1 shows how the complexity of Microsoft operating systems (measured in lines of code) has grown over the years. It is quite clear that the complexity increases the security level of the product at the one hand and affects negatively the functionality required by the stakeholders at the other hand.

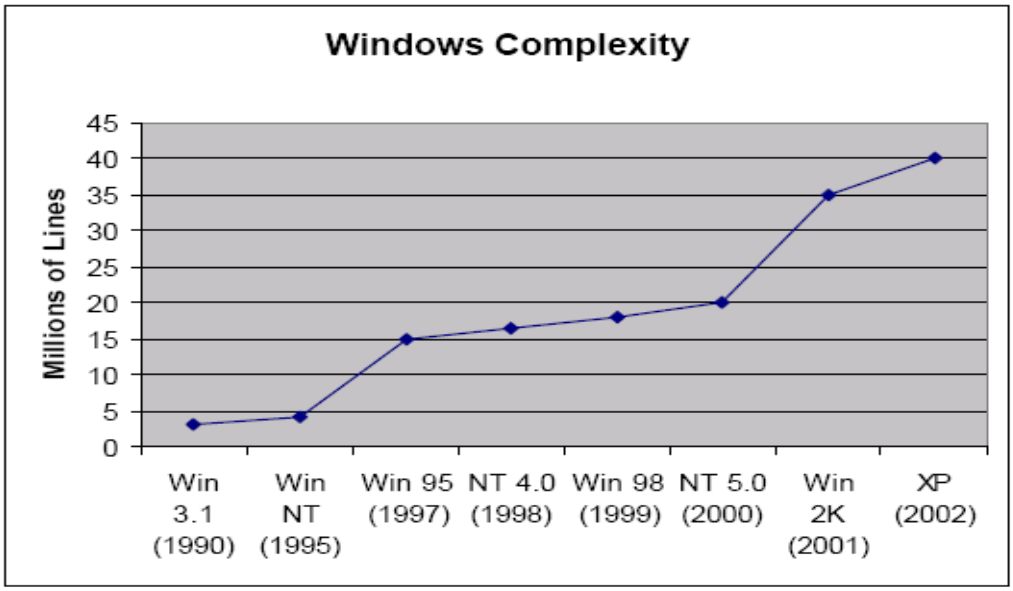

Fig. 1: growth of the Microsoft Operating System from 1990 to 2002 [23].

With respect to NFRs conflict, [25] reports a case study in which implementing of two steps password-check to have strong security has resulted in usability compromise. In another subject related to NFRs stakeholders, [11] confirms that NFRs play an important role in checking the satisfaction level at stakeholders in addition to alerting them on the trade-off between some NFRs since this role can easily reduce stakeholder-dissatisfaction.

Service-Oriented Architecture (SOA)

The SOA is a very powerful and new shift paradigm in the world of information systems; its introduction makes "Software as a Service" a unique possibility. As a vehicle for SOA implementation, the web services technology is promising that automated composition of services can be achieved [26]. Solutions are available for service discovery, composition, and description. Users do not have to be tightly connected with particular hardware or software vendor and their independence will be achievable through selecting preferable middle ware and other protocols (HTTP, XML, and SOAP). Moreover, highly cohesive and loosely coupled web service is going to be de facto [27, 28] in the IT world since the web service is expected to perform one single task with known interface and results in addition to matching quality of service level expected by the customer as per Service Level Agreement between the two parties i.e. the service provider and the service requester. From the management point of view, meeting the organization's vision, mission, and strategic objectives will be doable via using the web service technology [29, 30]. SOA provides high abstraction level and it enables the organization using its existing legacy systems which could be using COBOL, Object-Oriented, Component-Based Objects among other technologies.

Modern Requirements Elicitation Techniques

Software requirements serve as a contractually binding specification, and guide the design, implementation, and testing efforts. In most projects, the requirements are proactively elicited from a broadly representative group of stakeholders through a

carefully coordinated series of informal interviews, surveys, studies, and organized brainstorming sessions such as JAD [31].Collecting software requirements activity is considered by various levels of users as difficult, time consuming, and inefficient. Abilities of communication services within the intended organization going to conduct software requirements elicitation has a major role to play in this context and it affects considerably consequent activities, tasks, and processes [32]. People's opinions differ completely and significantly during this activity due to naturally existing flavours between individuals. One of the most difficult tasks in building systems is in acquisition and encoding of the requisite domain knowledge. Knowledge for systems must be derived from expert sources and elicitation of right knowledge can take several years and cost huge amount of money [33] especially in case of expert systems or knowledge-based systems. The software requirements process, including the tasks of eliciting, analyzing, and specifying the functional and non- functional requirements of a system, represents one of the most critical phases of the software development lifecycle [31]. The main objective of the elicitation techniques is to support the maximum level of stakeholders' collaboration, cooperation, and contribution. In the following sections various elicitation techniques will be shown.

Creativity technique

The need for creativity techniques in requirements elicitation is emerging in the field of computer applications under the notion that that "requirements are not just out there to be found" [34]. In spite of such recommendation on elicitation approach, there are unavoidable topics such as NFRs that must remain unaddressed so as to guarantee some focus [34]. One can say that Maiden's recommended elicitation technique is human-intensive requirements generation technique. An example of newly creative techniques in requirements elicitation is RESCUE creativity techniques [35].

Search-based technique

Other researchers are promoting different approach for requirements elicitation. This approach (which was announced in REFSQ'08) intends to reduce RE problems, like requirements selection and prioritization, to search problems, and then apply multi-objective search-based techniques to lead the search of an optimum [35]. This technique is machine-like and formalized.

Parameterized Online Simulation Environment (POSE)

Another group of researchers [35] have developed a quality-oriented tool called POSE for requirements elicitation. Its function is to facilitate the elicitation of quality requirements through interactive simulation of the system which is going to-be maintained or developed. The tool is fed with a description of both the business processes and the organizational environment in which the business processes are going to take place. Then, through a "business game" option in the utility, new requirements can be elicited, or various versions of the processes can be compared according to how they match some quality requirement. One can say that there is a possible high cost of this approach which requires complete process and context specification and a non trivial set-up. Future case studies will investigate cost-benefit analysis of such tool (POSE)

Elicit Method

Software process elicitation (developed by Madhavji et al.) highly recognized and notable work in the field of requirements elicitation. It provides strong framework for

eliciting descriptive processes. The framework consists of three stages View, Method, and Tool. Elicit Method is technical and platform dependent which impacts its ability for practical application by everyday software professionals involved in software process improvement. The consequence of this limitation is that not all process users have the same level of technical knowledge that can be leveraged in elicitation [32].

Spearmint

This model was development by (Becker-Kornstaedt et al.) for software process elicitation. It focuses on technical support for the process engineer. Spearmint provides capabilities for classifying and representing process models in an application environment that is web based where its power rests in the Electronic Process Guide generator (EPG) tool to create process specifications and it provides direct support for the process engineer [32]. Spearmint directly impacts efforts for documentation of processes in process improvement.

Prospect

This model was development by (Becker-Kornstaedt et al.) to formulate a decision model to address the problems faced by inexperienced process engineers conducting elicitation efforts [32]. Prospect techniques are defined according to a template, this template includes the preconditions for the application of a technique, the information source types (roles or document types) the technique can be applied to, and the information a technique can elicit in terms of process model elements. Prospect technique uses natural language and questions to elicit the process model data.

Win-Win

This approach assumes that the computer project can be a successful one if and only if stakeholders collaborate and make everybody a winner. Win-Win process provides a collaborative environment in which stakeholders brainstorm their needs, identify and prioritize stakeholders' win-conditions, identify Issues, Constraints, and Options, and then finally negotiate Agreements [31]. In this model, Conditions are recorded as the stakeholders' desired objectives, Issues capture the conflicts between win conditions and their associated risks and uncertainties, Options are used to propose alternatives and solutions to resolve an issue; and Agreements describe achieved negotiation result [36]. In term of market tools there are some tools that support this approach. EasyWinWin tool is a groupware system designed based on Win-Win to involve multiple stakeholders and help them to gain a better and more thorough understanding of the problem and supports co-operative learning about other's viewpoints. WikiWinWin is another tool based on the Wiki technology and Win-Win approach to facilitate the collaborations during requirements negotiation.

Joint Application Development (JAD)

Joint Application Development (JAD) brings users and technical professionals together and is designed mainly to discover end user's functional requirements (FR) in software development [37]. JAD gathers a broad range of stakeholders together in highly structured and focused meetings to gather information and specify requirements [31]. The focus of JAD is a structured workshop known as the JAD Session, in which all stakeholders design a system. JAD includes five distinct phases:

Project Definition Phase, Research Phase, Preparation Phase, the JAD Session, and the Final Document.

In the first two phases of JAD, it is intended to interview the stakeholders. The JAD Session Phase (the heart of JAD) addresses work flow, data elements, screens, reports, and open issues. In the opinion of the researcher, JAD can be used in conjunction with other methods to deal with quality requirements since JAD has proved itself as a reliable method to show FRs which means that we can benefit its strength in that area but not in NFRs area.

The researcher looks at JAD as an effective way of gathering end-users' functional requirements [37] but not NFRs since NFRs elicitation require different approach per each NFR. For various types of NFRs, the development teams may follow unique approach for some NFRs such as Security which has different aspects in comparison with other NFR such as maintainability. As a result, JAD does not provide specific way to elicit quality requirements.

Issue-Based Information Systems (IBIS)

IBIS was developed in the 1970s to improve the definition, discussion, and resolution of "wicked" problems where it intends to address issues that are ill- defined or strongly challenged among stakeholders. In IBIS, all problems are decomposed into issues, which are phrased in the form of open questions to the stakeholders [37]. IBIS enables stakeholders to collaboratively verbalize and explore key issues and related arguments in order to find and articulate solutions [31].

Each issue is then resolved by proposed positions, which are resolutions to the issue put forth by the stakeholders. Every position has corresponding arguments that either support or oppose the position [37]. Since IBIS has a good supporting CASE tool freely available from the Compendium Institute's Web site [http://compendium.open.ac.uk/institute].Using such tool, optionally, the requirements engineer is tasked with recording the communication of issues, positions, and arguments. The results are then presented in the form of an IBIS Map (IM). The IBIS maps are analyzed by the requirements engineering team and stakeholders to elicit the requirements [37]. IBIS was effective for documenting complex decision-making discussions but did not provide a structured way of generating NFRs such as security [37].

Unfortunately the researcher could not found a tool / mechanism that transform the IBIS maps into plain requirements.

Meta-Synthesis Approach (MSA)

MSA is one of the approaches to deal with open complex giant system problems form the view of systems. The heart of the approach is to synthesize the intelligence of experts, the high performance of computer and data, information to understand the problems comprehensively and to make better decisions [36].

Hall of Workshop for Meta-Synthetic Engineering (HWMSE)

MSA has evolved into Hall of Workshop for Meta-Synthetic Engineering (HWMSE). It is composed of three parts: Machine system, Expert system and Knowledge system. Experts guide and determine key points through the involvement of creative thinking and Machines carry out the repetitive and tedious work. The knowledge system is not static but dynamic as it continuously changes by interacting with the environment [36]. All the

three parts are synthesized and integrated by modern high and new technology. Within this approach, human experts make full use of advanced information technologies to achieve data, information and knowledge support for quantitative analysis.

Accelerated Requirements Method (ARM)

ARM (variation of JAD [38]) is facilitated requirements elicitation and description activity process [39] and is designed to elicit, categorize, and prioritize requirements and includes three phases [38]. ARM is an effective and rapid method of collecting requirements developed for use in a commercial environment [38]. Till today, ARM has no CASE tool. Figure 2 shows the three phases of ARM.

Fig. 2: ARM three phases.

In ARM, the team leader should choose the correct focus question and should prepare relatively large number of questions to be asked to the stakeholders per each requirement [38] in order to gather systematically the maximum amount of related information about FRs and NFRs in the same context. ARM is extremely effective and rapid method of collecting requirements particularly security requirements as shown in several case studies [37]. In the same reference [37] ARM has outperformed IBIS and JAD due to its suitability in NFRs subject.

In spite of its powerful over other techniques such as JAD and IBIS in NFRs area, ARM is not designed to collect NFRs for SOA.

Misuse Cases

This approach has been given this name to distinguish it from standard use cases since use cases describe system behaviour in terms of functional (end-user) requirements. This approach views the software from the perspective of an adversary in which the authorized user interaction with the system is diagrammed simultaneously with the hostile user's interactions [40]. Misuse cases approach, is based on UML, to handle NFRs and they are the inverse of the standard use cases and it describe functions that the system should not allow [11]. Each misuse case drives a requirement and corresponding test scenario for the software specially in developing requirements for security functionality [40]. Misuse cases apply the concept of a negative scenario that is a situation that the system's owner does not want to occur in a use-case context where major significant attribute of misuse cases approach is that it leads to quality requirements, such as safety and security [38]. Although this approach is very effective in gathering certain NFRs but it is not to cater for services concept.

Quality Function Deployment (QFD)

QFD is an overall1 concept that provides a means of translating customer requirements into the appropriate technical requirements for each stage of product development and production [39]. Although QFD covers a broad portion of the product development life cycle, the earlier stages of the process are applicable to requirements elicitation for software engineering; these stages include [38]:

1. Identifying the customer (stakeholders)
2. Gathering high-level customer requirements
3. Constructing a set of system features that can satisfy customer needs
4. Creating a matrix to evaluate system features against satisfaction of customer needs.

The problem of QFD is that it does not specify how to gather customer requirements and therefore it is not suitable for the research objective.

Tropos

Tropos (http://www.troposproject.org/) methodology considers not only the system functional requirements but also NFRs such as security, reliability, and performance [11]. Tropos offers support to systematically deal with the NFRs elicitation and reasoning [41]. Tropos is based on the idea of building a model of the system that is incrementally refined and extended from a conceptual level to executable artefacts, by means of a sequence of transformational steps [11]. The Tropos material (by Giorgini et al.) is a self-contained life-cycle approach; it is very specific in terms of how to go about requirements specification. The first version of Tropos was not sufficient for some NFRs such as security and that has led to the development of Secure Tropos [40].

Problems of Elicitation

Two increasingly prevalent software engineering practices have undermined the effectiveness of traditional requirements elicitation strategies and techniques [31]. First, the growth of the global IT marketplace has meant that customers and developers are often geographically distributed, and in-person requirements meetings are not feasible on a regular basis. Secondly, the growth in size and complexity of software systems and the associated increase in the number of stakeholders, introduces significant problems in managing and coordinating the human-intensive requirements elicitation process in large projects. The elicitation of NFRs is heavily neglected area due to lack of specialized elicitation methods in this area [39]. While on the side of functional requirements (FR), we may find lots of elicitation methods specialized for that area. One of these several functional-requirements based methods is JAD which is brainstorming-based approach that cannot fit for NFRs elicitation. The NFRs elicitation is considered in several process development models (during the architectural design stage) however, they do not consider, explicitly, a particular approach for building the quality model which risk having unfulfilled computer system with quality level [12].

Elicitation Conflicts

Managing requirements elicitation effectively requires detecting and handling conflicts and misunderstandings [42].

Documentation and lack of tools

Requirements elicitation and documentation are complex activities [42]. NFRs are among the most difficult and expensive type of requirements to deal with and the lack of support to deal with them could put in danger the success of the computer software [41].

Stakeholders' participation

Not only the requirements themselves will evolve but also the people involved and the means for managing the requirements will evolve too during the project. So, stakeholders' participation improves the quality of requirements products [42].

Lack of support in UML

SOA projects are normally not based on Use Cases but on business processes and because processes can be easily transformed to service compositions. Therefore, the business processes must be documented as well as part of requirements elicitation. Use Cases are used to elicit and document the requirements from the point of view of an actor(s) and due to their textual nature they can be understood by normal users who are not accustomed to Unified Modelling Language and other technical notations [43]. One can say that Use Cases are used for functional requirements elicitation and deriving business processes in the requirements specification phase but do not fit for NFRs.

Conclusion

In SOA, the NFRs (security, fault tolerance, the high availability of services, and the dependability) of services should be described separately from their functional requirements behaviour (i.e., business logic) because different applications use services and connections in different non-functional contexts [44]. Distributed Software Systems Group (DSSG) at the University of Massachusetts, Boston defines NFRs are security and fault tolerance requirements for a service in the SOA approach [45]. NFRs should be captured in an early project development phase and transformed to code or configuration files in order to improve development productivity, this activity incurs time-consuming and error-prone manual efforts to implement and deploy NFRs in later development phases (e.g., integration and test phases) [46]. In order to eliminate incorrect assumptions about web services' behaviour, services are required to announce, clearly, their non–functional requirements and capabilities in an agreed, machine readable format [47]. As an example of a product and its relationship with NFRs, [48] states that the final design and performance of complete Virtualization Engine solutions depends on numerous factors such as NFRs. These NFRs include requirements for solution characteristics, such as reliability, availability, security, maintainability, and performance since NFRs associated with any solution have the most influence over its final shape. NFRs construct a natural complement to other requirements' capture techniques such as use-case analysis [49]. Current techniques rely on the textual contents of the service descriptions interfaces to find web services with disregard to NFRs parameters of the service, this create some problems as stated by [50]. The problems that can be encountered due to the lack of structured description of the NFRs attributes within the web service are(i) Low-precision results are returned and (ii) No methods are provided enabling the stakeholder selecting among several services that can perform the same function [50].

Limited contributions have been brought to support the selection of services on the basis of their non-functional characteristics. As a consequence, software developers have no automated tools to support the analysis aimed at characterizing the performance and reliability (among long list of critical NFRs) behaviour of software applications based on the behaviour of the services and the architecture of the application [51].

One can say that the there is a need to address this area due to the lack of tools, approaches, frameworks, and methods that help development community selecting most appropriate web service(s) based on its NFRs suitability to the users requirements.

References

[1] Sommerville, Ian. Software Engineering. s.l. : Pearson, 2008.
[2] Classification of Research Efforts in Requirements Engineering. Zave, Pamela. 1997, ACM Computing Surveys, 29(4).
[3] A Tool Suite for Aspect-Oriented Requirements. Ruzanna, Chitchyan, et al. Shanghai : ACM, 2006. EA'06. pp. 1-7.
[4] Requirements Negotiation Using Multi-Criteria Preference Analysis. Olson, David and Peter In, Hoh. 2004, Journal of Universal Computer Science, pp. Vol. 10, no. 4 (2004), 306-325.
[5] An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement. Sommerville, Ian and Ransom, Jane. 2005, ACM Transactions on Software Engineering and Methodology, pp. Vol. 14, No. 1.
[6] Ebert, Christof. Dealing with nonfunctional requirements in large software systems. s.l. : Annals of Software Engineering, Volume 3, Number 1/January, pp. 367-395, 1997.
[7] Non-Functional Requirements: From Elicitation to Modelling Languages. Cysneiros, Luiz and Sampaio, Julio. s.l. : SIGSOFT: ACM Special Interest Group on Software Engineering, 2002. Proceedings of the 24th International Conference on Software Engineering.
[8] Mead, Nancy. How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods. s.l. : TECHNICAL NOTE CMU/SEI-2007-TN-021, 2007, TECHNICAL NOTE, CMU/SEI-2007-TN-021.
[9] Firesmith, Donald. The Business Case for Requirements Engineering. s.l. : RE'2003 http://www.sei.cmu.edu/library/assets/case.pdf, 2003.
[10] Miller, Kieran, Dawson, Ray and Bradley, Malcolm. IT Project Success – The Computing Holy Grail SQM. 2007.
[11] Matoussi, Abderrahman and Laleau, Regine. A Survey of Non-Functional Requirements in Software Development Process. Paris : University Paris, Technical Report TR–LACL–2008–7, 2008.
[12] Attribute-Based Techniques To Evaluate Architectural Styles For Interactive Systems. Losavio, Francisca, Chirinos, Ledis and Pérez, María. 2002, COMPUTER SCIENCE, Acta Científica Venezolana, Vol 53, no. 2, pp. 130–138.
[13] Chung, Lawrence, et al. Non-Functional Requirements in Software Engineering. s.l. : Kluwer Academic Publishers, 2000.
[14] Using UML to Reflect Non-Functional Requirements. Cysneiros, Luiz and Sampaio, Julio. Canada : s.n., 2001. Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research.
[15] Non-Functional Requirements for Object-Oriented Modelling. Neto, Jaime, Cesar, Julio and Cysneiros, Luiz. 2000. III Workshop de Engenharia de Requisitos.
[16] Identifying Quality-Requirement Conflicts. Boehm, Barry and Peter In, Hoh. 1996, IEEE Software, pp. 25-35.

[17] Chung, Lawrence. Representing and Using Non-Functional Requirements: A Process Oriented Approach. Ph. D. Thesis. s.l. : Dept. of Computer Science. University of Toronto, June 1993.Also tech. Rep. DKBS-TR-91-1., 1993.
[18] Peter In, Hoh and Boehm, Barry. Using WinWin Quality Requirements Management Tools: A Case Study. s.l. : Annals of Software Engineering (Kluwer), Volume 11, Pages (141—174), 2001.
[19] Dagnino, Aldo. Improving the Definition of Quality Attribute Scenarios By Using Requirements Patterns. s.l. : SATURN Conference 2009, http://www.sei.cmu.edu/saturn/2009/images/Improv_Def_Quality_Attr_Scenarios.pdf, 2009.
[20] Non-Functional Requirements Engineering - Quality is essential. Paech, Barbara and Kerkow, Daniel. s.l. : REFSQ'04, 2004. Proceedings of the 10th Anniversary International Workshop on Requirements Engineering: Foundation of Software Quality.
[21] Cysneiros, Luiz. Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation. 2001.
[22] Satisficing the conflicting software qualities of maintainability and performance at the source code level. Andreopoulos, Bill. s.l. : http://www.cs.yorku.ca/~billa/SIG/SIG.xml, 2004. In WER - Workshop em Engenharia de Requisitos.
[23] Steven, John. Are All Quality Goals Created Equal. Pittsburgh : SEI Software Architecture Technology User Network, www.sei.cmu.edu/library/assets/quality_steven.pdf, 2005.
[24] Software Security: Thought leadership in information security. McGraw, Gary. 2003. Proceedings of the Cigital Software Security Workshop.
[25] Non-functional Requirements: From Elicitation to Conceptual Models. Cysneiros, Luiz and Sampaio, Julio. 2004, IEEE Computer Society, IEEE Transactions on Software Engineering, www.cas.mcmaster.ca/~smiths/CSRS/Cys, pp. Vol. 30, no. 5, pp. 328-350.
[26] Towards a Task-Oriented, Policy-Driven Business Requirements Specification for Web Services. Gorton, Stephen and Reiff-Marganiec, Stephan. 2006, Springer-Verlag.
[27] The Landscape of Service-Oriented Systems: A Research Perspective. Kontogiannis, Kostas, et al. s.l. : IEEE, 2007. International Workshop on Systems Development in SOA Environments (SDSOA'07).
[28] Yu, Qi, et al. Deploying And Managing Web Services: Issues, Solutions, And Directions. The VLDB Journal Springer-Verlag 2006. 2006.
[29] McCabe, Francis, et al. Reference Architecture for Service Oriented Architecture Version 1.0 Public Review Draft 1,. s.l. : OASIS, 2008.
[30] Service-oriented Knowledge Architectures –Integrating Learning and Business Information Systems . Leyking, Katrina. s.l. : European Conference on Technology Enhanced Learning, 2007. EC-TEL 2007 – Second European Conference on Technology Enhanced Learning.
[31] A Recommender System for Requirements Elicitation in Large-Scale Software Projects. Castro-Herrera, Carlos, et al. s.l. : ACM, 2009. Symposium on Applied Computing, Proceedings of the 2009 ACM Symposium on Applied Computing.
[32] Exploring Language in Software Process Elicitation: A Grounded Theory Approach. Crabtree, Carlton, Seaman, Carolyn and Norcio, Anthony. s.l. : IEEE Computer Society,

2009. Proceedings of the 2009 Third International Symposium on Empirical Software Engineering and Measurement (ESEM '09).
[33] Knowledge-Based Modeling Approach for Performance Measurement of Parallel Systems. Chhabra, Amit and Singh, Gurvinder. 2009, The International Arab Journal of Information Technology, Vol. 6, No. 1, p. WWW.IAJIT.ORG.
[34] Inventing Requirements: Experiences with an Airport Operation System, REFSQ'08. Maiden, Neil, Ncube, Cornelius and Lockerbie, James. s.l. : REFSQ, 2008. REFSQ (Requirements Engineering: Foundation for Software Quality).
[35] Report Working Conference on Requirements Engineering. Paech, Barbara, Heymans, Patrick and Persson, Anne. Montpellier : ACM SIGSOFT Software Engineering Notes, 2008 . Requirements Engineering Foundation for Software Quality (REFSQ'08). pp. Volume 33, Number 5, Page 22-26.
[36] Distributed Requirement Elicitation and Negotiation Based on the Hall for Workshop of Meta-Synthetic Engineering. Dai, Chao-fan and Wang, Ming-li. 2009. Key Lab of Science and Technology for National Defense of C4ISR Technology, ICIS.
[37] Mead, Nancy. Requirements Elicitation Case Studies Using IBIS, JAD, and ARM. [Online] 2008. https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/532-BSI.html.
[38] Chung, Lydia, et al. Security Quality Requirements Engineering (SQUARE): Case Study Phase III. Pittsburgh, PA 15213-3890 : SPECIAL REPORT: CMU/SEI-2006-SR-003, Networked Systems Survivability Program, 2006.
[39] Experiences in Eliciting Security Requirements. Mead, Nancy R. 2006, CROSSTALK The Journal of Defense Software Engineering (CERT, Software Engineering Institute).
[40] IATAC and DACS. State-of-the-Art Report (SOAR). s.l. : Joint endeavor by Information Assurance Technology Analysis Center (IATAC) and Data and Analysis Center for Software (DACS), 2007.
[41] Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar. Cysneiros, Luiz, Werneck, Vera and Yu, Eric. 2004. In Proceedings of 7th Workshop on Requirements Engineering. pp. 40-55.
[42] Wiki-Based Stakeholder Participation in Requirements Engineering. Decker, Björn, et al. 2007, IEEE SOFTWARE , Fraunhofer Institute for Experimental Software Engineering.
[43] Dealing with User Requirements and Feedback in SOA Projects. Lubke, Daniel and Knauss, Eric. Hannover : s.n., 2007. Proceedings Workshop on Software Engineering Methods for Service-Oriented Architecture (SEMSOA).
[44] A Service-Oriented Design Framework for Secure Network Applications. Wada, Hiroshi, Suzuki, Junichi and Oba, Katsuya. s.l. : IEEE, 2006. The 30th Annual International IEEE Int'l Conference on Computer Software and Applications Conference (COMPSAC).
[45] Surfing the Net for Software Engineering Notes. Doernhoefer, Mark. s.l. : ACM, 2008. ACM SIGSOFT Software Engineering Notes, Volume 33 Number 3, May 2008.
[46] A Model-Driven Development Framework for Non-Functional Aspects in Service Oriented Grids. Wada, Hiroshi, Suzuki, Junichi and Oba, Katsuya. s.l. : IEEE, 2006. Proceedings of the IEEE International Conference on Autonomic and Autonomous Systems ISAC.

[47] Metadata For Trust In Service-Oriented Architectures. Coetzee, Marijke and Eloff, Jan. s.l. : Available at www.icsa.cs.up.ac.za/issa/2005/Proceedings/Full/032_Article.pdf, 2005. Proceedings of the 5TH Annual International Information Security South Africa (ISSA) conference.
[48] Cook, Jim, et al. IBM Virtualization Engine Platform Version 2 Technical Presentation Guide. WWW.IBM.COM/REDBOOKS. [Online] March 2006. WWW.IBM.COM/REDBOOKS.
[49] OASIS. OASIS Reference Architecture Foundation for Service Oriented Architecture Version 1.0. s.l. : OASIS, 2009.
[50] Enhancing Web Service Selection by QoS-Based Ontology and WS-Policy. Chaari, Sodki, Badr, Youakim and Biennier, Frederique. s.l. : ACM, 2008. Proceedings of the 2008 ACM symposium on Applied computing.
[51] Composition and Tradeoff of Non-Functional Attributes in Software Systems: Research Directions. Pasqualina, Potena. s.l. : ACM, 2007. ESEC-FSE '07: Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software.
[52] Kotonya, Gerald and Sommerville, Ian. Requirements Engineering: Processes and Techniques. s.l. : John Willey & Sons, 1998.