ISSUES IN TESTING OF SOFTWARE WITH NFR

Pratima Singh[1] and Anil Kumar Tripathi[2]

[1]Department of Computer Engineering, IIT (BHU).Varanasi, U.P. India
pratima.singh.rs.cse@itbhu.ac.in
[2]Department of Computer Engineering, IIT (BHU).Varanasi, U.P. India
anilkumartripathi.cse@itbhu.ac.in

ABSTRACT

Software Development has started experiencing the need of consideration of NFR (Non Functional Requirements) for producing high quality acceptable software. Mostly software engineering literature has considered only for testing Functional Requirements. In context of such a need this work attempts to consider NFR, resulting from quality concerns of stakeholder, along with their impact and effect on testing .We identify and bring out issues, in testing of NFR that warrant, purposeful and meaningful considerations.

KEYWORDS

Testing issues, Non-functional Requirements, Testability, FR (Functional Requirement.), Goal Centric Traceability (GCT), GRL (Goal Oriented Requirement Language), SIG (Soft goal Interdependency Graph), soft goal.

1. INTRODUCTION

Testing happens to be an important phase in software development lifecycle. It consumes around 70% of resource required for developing a software system. [1,2,4,5,6,7,8]. According to Brian Lawrence[48] , among the top 10 risks of Requirement Engineering is "Overlooking a crucial requirement" and "Modelling only Functional Requirements". As stated by L. M. cysnerio[10] "Although Non-Functional Requirements (NFR) have been present in many software development methods, they have been treated as a second or even third class of requirement, frequently hidden inside notes and therefore frequently neglected or forgotten." Surprisingly, despite the fact that non functional requirements (NFR) are among the most expensive and difficult to deal with, even today there are only some work that focus on NFR as first class requirements" Chung[16]stated "Surprisingly NFR has received little attention by research and definitely less well understood than other less critical factors in software development". It is universally accepted fact that NFR play very dominant role in acceptability of software, but they have been treated very off handily, by industry for long, until they realized the fact that NFR cannot be neglected further, because NFR not satisfied, results into low acceptability which goes against the product because of increasing competitive market, expectations of stakeholders, failures of various critical system. [12]. In literature ambulance case[39], Moose-test of the Mercedes Benz A Class [17] and the Siemens mobile[40] have been commonly discussed with reference to software systems *not being acceptable* because of negligence of NFR."Essentially software utility is determined by both FR and NFR nonetheless, there have been more emphasis on FR and its testing, even though FR is not useful without NFR." In an NFR survey by Fridge

and Lister [38] it is stated that "most startling and disturbing, deficiency is the shortage of measure for Specification and design methodology of any form of NFR."

These are reasons enough for Software Engineering research community to take NFR as a research theme and look into the issues, challenges and problems in handling NFR while, specification, designing, coding and testing of software systems. With such an intention this work attempts to answer some possible Research Questions:

1) What are the problems in handling NFR?
2) What are the problems in testing a software with NFR?

Based on the above said, the rest of the papers are organized, as follows:
Section 2: Issues and challenges of handling NFR in software development process. Section 3: Literature Survey: Current methods to handle NFR at various stages of development. Section 4: Issues and challenges of Software Testing with NFR. Section 5: Difference between NFR testing and FR Testing. Section6: Future Research Directions. Section 7: Conclusions and Observations. Section8: Future Research Directions.

2. ISSUES AND CHALLENGES OF HANDLING NFR IN SOFTWARE DEVELOPMENT.

Non functional requirements refer to a whole list of "ileitis" such as usability, reliability and availability, apart from some others such as performance and security. [10, 12, 16, 19]. There is no universally accepted definition of NFR [12] "NFR not only introduce quality factors but also represent global constrains under which a system must operate". They are global in the sense that they arise from all parts of the system and from their interactions. [11,16]. NFR are also known as Quality Requirements [27,10] and unlike Functional Requirements that address specific problems and are therefore typically implemented through particular localized modules or components.NFR provide the justifications for design decisions and constraints showing the way in which the required functionality may be realized, for satisfying the associated quality concerns of the stakeholders.[10,11,16,14,5,50,12,].

Issue No1: Identification of NFR.

Challenges: NFR are too soft or subjective to be identified clearly. [13]. They are very casually treated as they are hidden somewhere, in the software specifications or mentioned in form of comments or some special requirements in SRS [13] .They are too "fuzzy" and as late thought even in the minds of stakeholders.

Issue No2: Handling the Diversity of NFR.

Challenges: Great diversity in no and type of NFR makes it difficult to be handled by a common methodology. Various quality concerns of stakeholders have their specific associated constraints, resource requirement for their specification, designing and testing. This problem of diversity, can be seen appearing in several work [5, 44] done in an attempt to be able to clearly classify NFR on several possible basis but still having no clear-cut definition of NFR [45]

Issue no3: Interplays among NFR.

Challenges: Conflicting interplays among NFR where one NFR impacts negatively or positively on other, conflicts resolution among NFR is a problem [13,3,10,14,16]

Issue 4: NFR affecting Design decisions specifications.

Challenges: NFR plays very important role in justification of a design decisions.[29] Design decisions are based on both NFR and FR , but NFR helps in justification of particular choice of design decision[52]

***Issue 5* Problems of scope of FR and NFR**.

Challenges: The two extreme cases of representation of Requirements, as use case or misuse cases which needs to be separately understood. Misuse case are there to represent negative cases (generally used for security based NFR) which behaves as a constraint or limiting boundary for use cases.[20,28,36]

Issue 6: Ambiguous specification in software requirement specification (SRS).

Challenge: SRS is the origin of all system related errors. Two extremes associated with specification of NFR can be Formal method specification or Natural language specification. Natural language specification is easy but inherently ambiguous, contradictory to Formal Method specification which is unambiguous but difficult to deal with because of required mathematical "proof of concept" [5].

Issue 7: Traceability among NFR design, code and test cases.

Challenge : Due to seemingly insignificant presence of NFR at the specification stage, where a FR specification dominates the scene, it is a challenge to save *NFR from Omission error* [14],and provide traceability of Non Functional Requirement to design and code level.

SECTION 3: LITERATURE SURVEY: CURRENT METHODS TO HANDLE NFR AT VARIOUS STAGES OF DEVELOPMENT.

This section captures the significant work done towards handling of NFR at various stages of development. At requirement Engineering phase several Goal Oriented Approaches to handle NFR have been suggested such as i*, framework and GRL (Goal –Oriented Requirement Language), SIG (Soft goal interdependence graph) [21, 25, 35, 34]. All these goal oriented approaches are based on identifying NFR as soft goal which needs to be satisfied (within an acceptable limit, or goals merely met) or "not satisfied" at all. NFR are refined, unless they are identified or decomposed as FR and their significance for design decision is made. Example of NFR Framework, as shown in figure 2, *security of information* is decomposed into the sub goals *integrity, availability, confidentiality* through an AND type of contribution (i.e. only if all sub goals are met the overall goal is achieved),while the goal of *system performance* is decomposed into *throughput* and *response time*. Interestingly, it is necessary to address interactions between different kinds of non-functional requirements even though the non-functional requirements were initially stated as separate requirements. Note that *cryptography* contributes negatively (show as "-") for *system performance*.

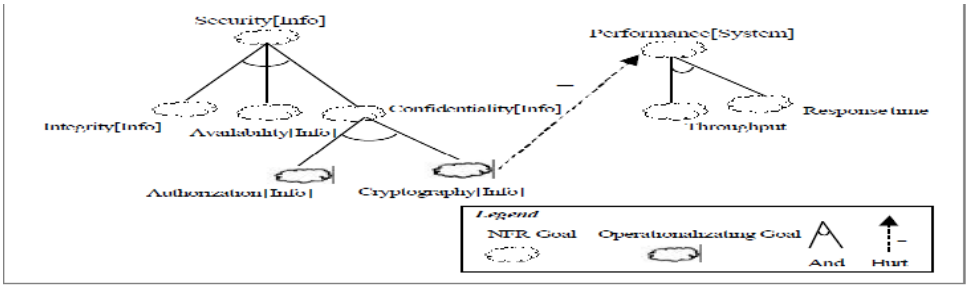

Figure 2. Decomposition of non-functional requirements using the NFR Framework

(above figure is adapted from[45])

The significant efforts in handling NFR in literature have been tabulated below: Table1

1)Significant attempts to handle NFR	NFR framework[10,11,12,14,16]
Salient Features of this attempt	Refining goal to sub goal till it is an operationalized goal
Advantage	Makes relationship between NFR & intended decisions explicit, as one design decision impacts multiple NFR positively or negatively.
Main issues attempted to catered to	issue no 1,2,3
2 Significant attempts to handle NFR	FRML(Formal Requirement Modelling Language)[26]
Salient Features of this attempt	Semi formal specification method bridging the gap between two extremes of specification by formal method and natural language specification.
Advantage	Takes advantage of Formal specification using temporal logic & ease of a modelling language
Main issues attempted to catered to	Issue No:6
3)Significant attempts to handle NFR	OORNF tool[11,15]

Salient Features of this attempt	Separate view of FR & NFR integrated by a LEL(Language Extended Lexicon).
Advantage	Smooth integration of FR and NFR makes both vertical traceability (among requirement design and code) and horizontal traceability (between FR to NFR) to happen
Main issues attempted to catered to	Issue No:7
4)Significant attempt to handle NFR	PREM(performance requirement evaluation model)[14,29]
Salient Features of this attempt	Agile approach to address the specification and testing of performance which is an important type of NFR
Advantage	To identify and specify performance requirement incrementally
Main issues attempted to catered to	Issue2: Focused attention on one of the diverse NFR.
5)Significant attempts for NFR	Misuse case, Abuse case or UMLsec[28,36]
Salient Features of this attempt	Misuse case is the inverse of use cases and describe functions that system should not allow
Advantage	More use full in analyzing Security threats
Main issues attempted to catered to	Issue1,3,5
6)Significant attempts for NFR	GDUC(Goal driven use case) ref[14]
Salient Features of this attempt	Deriving use case with goal

Advantage	Each use case is viewed as a process associated with a goal that it must achieve, optimize or maintain.
Main issues attempted to catered to	Issue 1,2,3
7)Significant attempts for NFR	GCT(Goal Centric Traceability)[21,34]
Salient Features of this attempt	QAM(quality assessment Methods) is the basis of this model
Advantage	Traceability of NFR to design decision is justified.
Main issues attempted to catered to	Issue 1,3
8)Significant attempts to handle NFR	i* (having its extension as TROPOS,URN)[18,37]
Salient Features of this attempt	It is an agent oriented approach in which actors are depicted as agents with intentional properties representing their belief, goal, abilities and negotiations
Advantage	Focuses directly on modelling NFR and soft goal It is a strategic dependency model,
Main issues attempted to catered to	Issue design decisions.
9)Significant attempts to handle NFR	KAOS[14]
Salient Features of this attempt	Uses Formal methods (acyclic graph)to represent both rational & satisfaction relationship

Advantage	KAOS relies on meta-models to provide a self descriptive and extensible modelling framework. This gives it the advantage of Model based analysis.
Main issues attempted to catered to	Resolves Issues No:1
10)Significant attempts to handle NFR	FDAF(Formal Design Analysis Framework), similar effort of formalizing requirement is (TLA+) extended temporal logic. [14]
Salient Features of this attempt	It assists the user in selecting formal methods and translates an extended semi formal UML design into formal notations.
Advantage	It exploits the benefit of formal method representation
Main issues attempted to catered to	Issue No 6
11)Significant attempts to handle NFR	"Constraint and object oriented programming styles.[14]
Salient Features of this attempt	Usage of exception handling mechanism
Advantage	Exceptional handling is the mechanism to identify constrains in these methods
Main issues attempted to catered to	Issue no1,
12)Significant attempts to handle NFR	FRIDA Model (From Requirements to Design using Aspects)[14]
Salient Features of this attempt	a)FRIDA determines a way to elicit and model FR and NFR separately. B)It uses conflicts matrix to resolve conflicts

Advantage	Used to accommodate view point of variant stakeholders using AOP(Aspect oriented Programming.)
Main issues attempted to catered to	Issues 3,2
13)Significant attempts to handle NFR	R++ is an extension of c++, similar effort is ILOG Jrules in java.[1,3]
Salient Features of this attempt	Is a combination of Rule based and object-oriented based development methodology
Advantage	Takes the advantage of Rule based techniques
Main issues attempted to catered to	Issue6
14)Significant attempts to handle NFR	B Method [14]
Salient Features of this attempt	Full formal method which uses set theory notations to specify, design & implement software Systems
Advantage	Refinement process transforms abstract non-deterministic specification into concrete deterministic system. Key merit of such refinement mechanism is the ability to preserve already proven system property in higher level models.
Main issues attempted to catered to	Issue 1,2,3
15)Significant attempts to handle NFR	Testing of NFR is in embryonic stage[ref 54] limited to only debugging distributed real time system.
Salient Features of this attempt	Limited to only replay & visualization of Computations

Advantage	Simple visualization by usage of probes provide effective means of identifying computational bottleNECK
Main issues attempted to catered to	Issue 5
16)Significant attempts to handle NFR	Cosmic FP[23,24]
Salient Features of this attempt	Extending COCOMO model for cost evaluation due to NFR
Advantage	One of the least significant attempts to evaluate cost due to NFR.
Main issues attempted to catered to	Issue 4
17)Significant attempts to handle NFR	POMSA(Process Oriented Metrics for software architecture adaptability)[51]
Salient Features of this attempt	Trace back the reason for taking design decision
Advantage	Gives justification for taking certain design decisions
Main issues attempted to catered to	Issue 4
18)Significant attempts to handle NFR	TRAGOSOMA(Traceability driven Goal Solution Mapping.)[47]
Salient Features of this attempt	It uses goal oriented method to guide the design activity, support conflict resolution, decision making and classification of solutions.
Advantage	a)Gives justification of how NFR affect architectural decisions b)Tracing of requirement specification to design
Main issues attempted to catered to	Issue 1,2,3,4
19)Significant attempts to handle NFR	ZCL

Salient Features of this attempt	First Order Temporal logic are utilized to deal with idea of including NFR in software design through architecture
Advantage	Usage of formal method for an existing configuration model CL framework .
Main issues attempted to catered to	Issue 6

Section 5: Prevalent Testing issues in light of NFR.

Testing cannot be thought about suddenly, in a day[41,42,] in the last phase of release when there is highest pressure for the release, of the product. It has to start right from the first phase of development. Therefore specify for testability, design for testability and code for testability should be done so that, we get good quality and testable product, at the end when then pressure of delivery is prioritized over that of quality of delivered product [7, 2]

Issue1: Specify for testability of NFR.

Challenges: Testability refers to "the degree to which a system or a component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.[32,33]. NFR in SRS poses enormous challenges. specification in natural language is inconsistent. This becomes more prevalent in case of NFR which are too "soft", abstract or fuzzy even in the minds of stakeholders. Formal method specification holds the key to unambiguous specification but it has its own limitation of being difficult to be usable by all different types of stakeholders. No doubt it increases the challenges of Requirement Engineers, Architects, Designers and coder community ,but this effort is worth it as one of the seven Testing Principles emphasizes on" Early Testing"[4,8] i.e." stitch in time saves nine" This issue has been addressed well for FR but remains to be studied in case of NFR.

Issues 2: NFRs role in decision making at design level.

NFR affects various decisions made while choosing various design options. NFR justify the reasons behind the choice of a particular architecture. [3] which is of great importance when there is conflicting requirement. [29]

Issue 3: Code for testability of NFR.

It can intuitively be improved if the stake holders concerns which are cross cutting across various modules can be coded separately as "concerns". And related test aspects for these concerns can be written to test these "aspects" or "viewpoint" as and when required at these "join points" and "point cut" [5, 30, and 31]

Issue4: Cost and Effort Estimation of Testing of NFR:

There has been very few models contributing towards the cost and effort estimation of different development models such as COCOMO Model, FP model or LOC(lines of code) based

models.[9] There seems no model known specially for estimating the cost of testing a software . [45]. No doubt cost of testing an NFR is very high due to need of testing various permutations and combinations of configurations required for acceptance of product, then there is need of dynamic test environment, or test scenarios or no. of test scripts for NFR testing.

Issue 5: Testing of Mission Critical Systems.

Testing of , real time mission critical system is difficult, due to need to generate near "life life environment". It is very difficult to create details regarding customers hardware setup, deployment information and test data .Due to confidentiality

Issue involved with mission critical system, type of data used may vary far from the actual type needed by the customers. Test data is built based on sample data collected by testers or is collected from similar or related products. MBT((model based testing) may be explored for testing of NFR[49,50], because intuitively modelling Non Functional Requirement may yield test data very close to actual data, especially where mission critical systems have strict "entry criteria" and "exit criteria".

Issue 7 Deciding the stopping criteria. [9,8]

Deciding the test coverage criteria [23], i.e. How much testing is enough is dominant testing issue, This becomes more dominant in case of NFR because of its nonlinear availability across various modules. The penetration of these concerns across various modules makes it difficult to decide as to how much testing is sufficient, or what are the test coverage criteria for NFR. Clear cut establishment of Entry and Exit criteria followed by its execution can give a basis for deciding the stopping criteria for NFR testing.

Issue 8 Generation and Execution of Efficient test Suite:

In NFR, testing needs reusability of test suites because of need to test, variant configuration, or environment in which acceptance testing may take place because of variety of customers environment, configuration possible at customers end.[2] One often needs to change the test-script to check the variety of test scenario generated. Test Suite is a collection of Test cases. It should be an optimized collection, of new and old reusable test cases. NFR like maintainability and reusability heavily depend on optimization of existing or new test cases. Generically speaking all the issues in testing boils down to generation, execution, optimization and evaluation of test cases in the test suite. These problems get aggravated with NFR testing, which holds its own share of problem

Issue 9 Test Oracle Generation:

what should be the basis of evaluation of test result is an issue of software testing as a whole. Usually SRS (as the contract document), or existing similar software forms the test oracle.NFR testing is a part of system testing which is done against SRS. Due to vague and ambiguous specification of NFR, SRS becomes weak oracle for NFR testing.

Issue 10 Testing Metrics:

There has been very little success in development of any model on Test metric for NFR, as opposite to various development metric available in testing of FR.[9,41,and 42] . Basic elements of product and project metrics are difficult to identify, for NFR due to its complex nature , where there is need of high volume of sample data for analysis. NFR testing results in huge collection

and analysis of data[2]. It needs expertise knowledge of product, domain, design and statistical skills. There are various commonly existing testing metrics for FR. like, defect find rate, defect fix rate, defect cause distribution, defect classification trend etc.

11) **Test Automation:**

How much automation is possible? for what all testing activity is a concern of testers. Various NFR such as performance testing, security testing, stress, load and GUI testing lending themselves easily to automation process, where there is need to analyze large volume of data or exponential combinations of configuration or environment needed to be tested for acceptance of a product.[6,8] Creating test cases, test environment, or test data to simulate actual load or stress condition is good scenario for automation. Thus "automation" is a fertile area for NFR Testing because of inherent problems/ limitations / issues raised by NFR testing which are very nicely handled by Automation. There are several successful automation tools available commercially, for NFRs like stress , load ,GUI or performance testing like load runner win runner , Rational Robo, QTP, Test Control etc. There is plethora of tools available for testing of different types of NFR.

Section 6: Testing of FR vs. NFR. [5, 2]: Table2

Sl.No.	FR Testing	NFR Testing
1)	Testing of FR is Testing of Functionality	Testing of NFR is testing of constrains over those Functionality.
	FR Testing states "what" the system must do	NFR constrain "how" the system must accomplish the "what".[5,14].
2)	Testing of FR involves product features & functionality.	Testing of NFR involves quality factor
3)	Testing is done through simple steps written to check expected results.	Testing yields huge data set collected and analyzed.
4)	Testing focuses on defect detection	Testing focuses on qualification of results.
5)	Testing requires knowledge of product and domain	Testing requires knowledge and experience of product, domain, design, architecture and statistical skills.
6)	Failure is normally due to code	Failure is normally due to code, design and architecture.
7)	Testing phase involves usually unit,	Testing phase involves usually

	component and integration level.	system level testing.
8)	Testing of FR is easy because of its well defined goal.	NFR testing is difficult due to its inability to operationalize the soft goals into concrete testable objective.
9)	Result of testing varies due to product implementation	Result of NFR testing varies due to product implementation, resources and configurations.
10)	FR Testing talks about clear pass or fail criteria	NFR requires test results be documented in Qualitative as well as quantitative form.ie apart from verifying pass or fail the effort required in test execution[2]
11	Testing of FR requires one time setup for a set of test cases	Testing of NFR requires configuration changes for each test case.

Section 8: Conclusion and Observations:

Testing Techniques have evolved through various phases, starting from not being separable from Debugging to Current phase of preventing faults in requirement, design and implementations. [43]. Testing of NFR has traditionally been done informally using Ad hoc Approaches. [4, 6].There have been multiple reasons for this. Handling NFR is inherently difficult due to its soft, subjective and subtle nature, and all the more due to great diversity, giving rise to conflicting requirements. Informal treatment of NFR, leads to non traceability of NFR from requirement through design, coding and testing phase leading to low testability. Testing of NFR imposes its own set of challenges against FR. Currently, various approaches to handle the issues of NFR specification to testing is centred around “Goal oriented approaches” [10, 11, 12, 13, 14, 1516, 17, 19, 21]

Section 9: Future Research Direction:

The subsequent research directions for future exploration are:

1) Testing of NFR can be made more effective if NFR can be specified, designed and coded for high testability. Requirement engineering has handled the specification of Functional Requirements very clearly and measurably [5], may be because of clear-cut identification and specification of FR in SRS, so their design and testing does have numerous measures and metrics for their evaluation. There are various well defined metrics for FR at design level, such as: No. of Modules, Interfaces, Level of Cohesion and Coupling or KLOC. The same does not hold good for NFR testing. one of the possible future directions can be working out testability measures for NFR particularly.

2)NFR results from quality concerns of stakeholders. Aspects (in AOP) deal with crosscutting concerns and hence Aspects can be purposefully used not only for designing software with NFR but also for their testing. Similarly test aspects can be written for testing software with respect to an NFR.

3) MBT can be purposefully used for NFR Testing. A possible approach for this purpose can be worked out. As realized, after survey, NFR can be handled more concretely by MBT because, it models the real life situations yielding concrete test cases from abstract formal models. [49, 50]

REFERENCES

[1] Pressman, R. S. (2005). Software Engineering: A Practitioner's Approach (6th ed.). New York:-McGraw-Hill Publication.

[2] Srinivasan Desikan,Gopalaswamy Ramesh ,Software Testing :Principles and Practices, Pearson,2006

[3] Len Bass,Paul Clements ,Software Architecture in Practice,2nd edition,2003 Pearson

[4] Rex Black, Foundations of software testing2008, cengage learning press,

[5] Sommerville, I. Software Engineering. Low Price Edition.

[6] Pankaj Jalote: Practical approach to software engineering,

[7] Rajib Mall ,Fundamentals of Software Engineering, Third Edition, PHI Publication.

[8] Glenford J Myaers ,The Art of Software testing, second Edition ,John willey

[9] Fenton, N.E. and Pfleeger, S.L. "Software Metrics: A Rigorous and PracticalApproach" 2nd ed., International Thomson Computer Press, 1997.

[10] Luiz Marcio Cysneiros, Member,Julio Cesar Sampaio do Prado Leite, Member, Nonfunctional Requirements:From Elicitation to Conceptual Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 5, MAY 2004

[11] Cysneiros,L.M., Leite, J.C.S.P. and Neto, J.S.M. "A Framework for Integrating Non-Functional Requirements into Conceptual Models" Requirements Engineering Journal —Vol 6 , Issue 2 Apr. 2001, pp:97-115.

[12] Lawrence Chung1 and Julio Cesar Sampaio do Prado .T. Borgida et al. On Non-Functional Requirements in Software Engineering (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 363–379, 2009.© Springer-Verlag Berlin Heidelberg 2009

[13] Saeed Ullah, Muzaffar Iq bal, Aamir Mehmood Khan,A Survey on Issues in Non-Functional Requirements Elicitation 978-1-61284-941-6/11 2011 IEEE

[14] A. Matoussi, R. Laleau. A Survey of Non-Functional Requirements in Software Development Process, October 2008 Laboratory of Algorithmic, Complexity and Logic (LACL) University Paris 12 (Paris East) Technical Report TR–LACL–2008–7

[15] Luiz Marcio Cysneiros and Julio Cesar Sampaio Leite. Using UML to reflect Non-functional Requirements. In CASCON, page 2, 2001.

[16] Chung, L., Nixon, B., Yu, E. and Mylopoulos,J. "Non-Functional Requirements in Software Engineering": Boston Kluwer Academic Publishers 2000.

[17] Chung, L., Nixon, B. "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach" Proc. 17th Int. Con. on Software Eng. Seatle,Washington, April pp: 24-28, 1995.

[18] John Mylopoulos, Marco Pistore, and Paolo Traverso. Model Checking Early Requirements Specifications in Tropos. In RE '01: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE '01), page 174. IEEE Computer Society, 2001.

[19] Martin Glinz. On Non-Functional Requirements. In 15th IEEE International,Volume , Issue , 15-19 Oct, pages 21–26, 2007.

[20] Jan Jurjens. UMLsec: Extending UML for Secure Systems Development. In UML'02: Proceedings of the 5th International Conference on The Unified Modeling Language, pages 412–425, London, UK, 2002. Springer-Verlag.

[21] Sam Supakkul and Lawrence Chung. Integrating FRs and NFRs: A Use Case and Goal Driven Approach. Proc. SERA 04, pages 30–37, 2004.

[22] Steffen Zschaler. Formal Specification of Non-functional Properties of Component-Based Software., Workshop on Models for Non-functional Aspects of Component-Based Software (NfC'04) at UML

conference 2004, September 2004. Technical Report TUDFI04-12 Sept.2004 at Technische Universit¨at Dresden.

[23] Mohamad Kassab, M. Daneva, and Olga Ormandjieva. Scope Management ofNon-Functional Requirements. In EUROMICRO '07: Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications,pages 409–417. IEEE Computer Society, 2007.

[24] A.Keshav Bharadwaj, T.R. Gopalakrishnan Nair, Mapping General System Characteristics to Non-Functional Requirements,2009 IEEE International Advance Computing Conference (IACC 2009)

[25] Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods Evangelia Kavakli* , Pericles Loucopoulos** DOI: 10.4018/978-1-59140-375-3.ch006,ISBN13: 9781591403753, ISBN10: 1591403758, EISBN13: 9781591403777 http://www.igi-global.com/chapter/goal-modeling-requirements-engineering/23011

[26] Greenspan, S., Mylopoulos, J., & Borgida, A. (1994, May 16-21). On Formal Requirements Modeling Languages: RML Revisited. Paper presented at the 16th International Conference on Software Engineering (ICSE-94), Sorrento, Italy.

[27] M. R. Barbacci, M. H. Klein, T. Longstaff and C. Weinstock, "Quality Attributes",Technical Report CMU/SEI-95-TR-021, Software Engineering Institute, Carnegie Mellon,University, December 1995.

[28] Ian Alexander Alexander Misuse Cases Help to Elicit Non-Functional Requirements, Engineering Use Cases with DOORS, Address given at RE'01, IEEE Computer Society p 264, 2001

[29] X. Franch, P. Botella, X. Burgués, J.M. Ribó.,Putting Non-Functional Requirements into Software Architecture1, In Proceedings of 9th Software Engineering and Knowledge Engineering Conference (SEKE), Madrid (Spain), 1997.

[30] Milena Guessi, Lucas Bueno Ruas Oliveira, and Elisa Yumi Nakagawa. Extensions of UML to Model Aspect-oriented Software Systems ,CLEI ELECTRONIC JOURNAL VOLUME 14 NUMBER 1 PAPER 3 APRIL 2011

[31] Wehrmeister, M. A. (2009). An Aspect-Oriented Model-Driven Engineering Approach for Distributed Embedded Real-Time Systems. PhD thesis, Federal University of Rio Grande do Sul, Brazil. www.inf.ufrgs.br/˜mawehrmeister/wehrmeister_thesis_final.pdf.

[32] Bret Pettichord., Design for Testability ,Pacific Northwest Software Quality Conference, Portland, Oregon, October 2002.

[33] Testability IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 610.12-1990.

[34] Jane Cleland-Huang, Raffaella Settimi, Oussama BenKhadra, Eugenia Berezhanskaya, Selvia Christina ,Goal-Centric Traceability for Managing Non-Functional Requirements,ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA. Copyright 2004 ACM 1-58113-963-2/05.

[35] Jonathan Lee and Nien-Lin Xue, Analyzing User Requirements by Use Cases: A Goal-Driven Approach, National Central University. IEEE Software J u l y / August 1999 0740-7459/99/1999 IEEE

[36] Alexander I.: Misuse cases help to elicit non-functional requirements. Computing and ControlEngineering Journal 14(1), 40–45 (2003)

[37] TROPOS, URN ,i*, Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. Information Systems 27(6), 365–389 (2002)

[38] C.J. Fidge and A.M. Lister," A disciplined approach to real-time systems design Information and Software Technology,34(9):603–610, September 1992.

[39] Finkelstein, A. and Dowell J. "A comedy of Errors: The London AmbulanceService Case Study" Proceedings of the Eighth International Workshop on Software Specification and Design, IEEE Computer Society Press pp 2-5 1996

[40] J.-L. Lions, ARIANE 5 Flight 501 Failure: Report by the Inquiry Board, http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf, 1996.

[41] Cem Kaner, J.D, The Ongoing Revolution in Software Testing,Software Test & Performance Conference, December 8, 2004

[42] Timişoara, Romania , Software Testing – State of the Art and Current Research Challenges,5th International Symposium on Applied Computational Intelligence and Informatics • May 28–29, 2009

[43] Lu Luo, Technology Maturation and Research Strategy Class Report for 17-939A Software Testing techniques.

[44] Aburub F., Odeh M. and Beeson I., "Modeling Non-Functional Requirements for Business Process",Information and Software Technology Journal, vol.49, no. 11-12, pp. 1162-1171, 2007.

[45] M. Kassab, C. Constantinides, and O. Ormandjieva,"Specifying and separating concerns from requirements to design: a case study", Proc. the IASTED International Conference on Software Engineering (ACIT-SE 2005), Novosibirsk,Russia, 2005.
[46] J. Zhao, "Towards A Metrics Suite for Aspect-Oriented Software", Technical Report SE-136-25, Information Processing Society of Japan (IPSJ), March 2002.
[47] Stephan Bode,Matthias Riebisch, Tracing the Implementation of Non-Functional Requirements,Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
[48] Brian Lawrence, Karl Wiegers, and Christof Ebert "The Top Risks of Requirements Engineering" IEEE SOFTWARE November/December 2001 0 7 4 0 - 7 4 5 9 / 0 1 / $ 1 0 . 0 0 © 2 0 0 1 I E E E
[49] Dr. Bruno Legeard, Model-based Testing: Next Generation Functional Software Testing, Dagstuhl Seminar Proceedings 10111Practical Software Testing : Tool Automation and Human Factors http://drops.dagstuhl.de/opus/volltexte/2010/2620
[50] Ibrahim K. El-Far and James A. Whittaker: Model-Based Software Testing Model-based Software Testing This paper appears in the Encyclopedia on Software Engineering (edited by J.J. Marciniak), Wiley, 2001
[51] Lawrence Chung ,Nary Subramanian, Process-Oriented Metrics for Software Architecture Adaptability, Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on2001,Page(s): 310 - 311 .
[52] Ruth Malan and Dana Bredemeyer, Functional Requirements and Use Cases, 2001 BREDEMEYER CONSULTING WHITE PAPER 8/3/01 2, http://www.bredemeyer.com/pdf_files/functreq.pdf

Biography of Author:

Anil Kumar Tripathi (M.Sc. Engineering (computer) from Odessa National Polytechnic University,Ukraine 1984, Ph.D. in Computer Engineering, Banaras Hindu University, 1992, Varanasi, India. Working as professor in Computer Engineering Department Indian Institute Of Technology, (Banaras Hindu University), Varanasi, India. He is engaged in teaching and research at IIT(BHU) for last more than 27 years in the area of Parallel/Distributed computing and Software engineering. He has to his credit some 50 research papers in journals. He has co-authored two research monographs: one from Springer USA and the other from John Wiley USA.Fourteen students have completed their Ph.D under his supervision.

Pratima Singh , MCA 2001,M.Tech from UPTU 2009 has been working as Assistance Professor in BBDNIT,Lucknow and AKGEC Ghaziabad. UP. since 12 years,. She has been teaching Software Engineering, Software Project Management and Computer Organization to the students of B.Tech and MCA .She has cleared ISTQB (Internationl Software Testing Qualification Board) Certification in 2007, subsequently tested few Products on Adhoc basis. Presently pursuing Ph.D. from IIT BHU .