An Investigation into the Notion of Non-Functional Requirements

Dewi Mairiza, Didar Zowghi, Nurie Nurmuliani

Faculty of Engineering and Information Technology
University of Technology Sydney, PO Box 123 Broadway, NSW 2007, Australia
(mairiza, didar, nur)@it.uts.edu.au

ABSTRACT
Although Non-Functional Requirements (NFRs) are recognized as very important contributors to the success of software projects, studies to date indicate that there is still no general consensus in the software engineering community regarding the notion of NFRs. This paper presents the result of an extensive and systematic analysis of the extant literature over three NFRs dimensions: (1) definition and terminology; (2) types; and (3) relevant NFRs in various types of systems and application domains. Two different perspectives to consider NFRs are described. A comprehensive catalogue of NFRs types as well as the top five NFRs that are frequently considered are presented. This paper also offers a novel classification of NFRs based on types of systems and application domains. This classification could assist software developers in identifying which NFRs are important in a particular application domain and for specific systems.

Categories and Subject Descriptors
D.2.1 **[Software Engineering]**: Requirements/Specifications

General Terms
Documentation, Performance, Reliability, Security

Keywords
Non-Functional Requirements, classification, types, type of system, application domain

1. INTRODUCTION
Non-Functional Requirements (NFRs) are recognized as very important factor to the success of software project.[1-3]. If NFRs are not addressed adequately, a number of potential problems may occur. For instance software which is inconsistent and of poor quality; dissatisfaction of clients, end-users, and developers; causing time and cost overrun for fixing software errors [1]. In the software development life cycle, NFRs are considered as the constraints or qualifications of the operations [4]. NFRs place restrictions on the product being developed, the development process, and specify external constraints that the product must exhibit [5].

NFRs are often more critical than individual Functional Requirements (FRs) in the determination of a system's perceived success or failure [6; cited by 7, 8]. Neglecting NFRs has led to a series of software failures [3, 9-12], such as systemic failure in London Ambulance System [10, 11], the system failure because of performance-scalability problems in the New Jersey Department of Motor Vehicles Licensing System [13] and some other examples as described in [10, 13-15].

Though NFRs are widely recognized very important, literature review shows that NFRs are often neglected, poorly understood and not considered adequately in software development. In the development of software system, users naturally focus on specifying their functional or behavioral requirements, i.e. the things the product must do [1, 7]. Hence NFRs are often overlooked in the software development process [3, 16]. A number of studies investigating practices of dealing with NFRs in the software industry also report that commonly software developers do not pay sufficient attention to NFRs [3, 16-18]. NFRs are not elicited at the same time and the same level of details as the FRs and they are often poorly articulated in the requirements document [17, 18].

Further investigation shows that neglecting NFRs in developing a software system is strongly influenced by NFRs' characteristics, that are subjective, relative, interacting [1], abstract [10, 19] and not uniform in nature [2]. These characteristics cause NFRs difficult to deal with. It is more difficult to model, verify, test, and measure NFRs to compare with FRs [1, 8, 20, 21]. Also, capturing, specifying, and managing NFRs are still difficult to perform because most of software developers do not have adequate knowledge about NFRs and little help is available in the literature [22]. Majority of software engineering research, particularly requirements engineering research only deal with FRs, i.e. ensuring that the necessary functionality of the system is delivered to the user [23].

The term NFRs has been in use for almost three decades. However, studies to date indicate that currently there is still no general consensus in the software engineering community regarding the notion of NFRs [1, 7, 24, 25]. Glinz [24, 25] even argues to rethink the notion of NFRs due to the fact that there is not a clear concept about what a non-functional requirement really is. Literature review also shows that a number of essential

dimensions related to NFRs (e.g. a variety of perspectives in considering NFRs as well as the relevant NFRs in various types of systems and application domains) are not well understood. Due to these reasons, this research is motivated to perform an extensive and systematic investigation of the notion of NFRs in the software engineering literature in order to increase the understanding of this complex and multifaceted phenomena.

This study covers three essential dimensions: (1) definition and terminology; (2) types; and (3) NFRs in various types of systems and application domains. A number of research questions have been derived from these three parameters:

1. How many perspectives are there in the software engineering community when considering NFRs?
2. What is the typology of NFRs?
3. Which types of NFRs are commonly considered or often discussed in the literature?
4. Which types of NFRs are of concern in various types of systems?
5. Which types of NFRs are of concern in various application domains?

The major contributions of this paper are a novel classification of NFRs based on typology, definition, types of systems, and application domains. These contributions would benefit software engineering community (i.e. researcher and practitioner) in many ways. Mapping between NFRs and the types of systems as well as between NFRs and the application domains were developed by conducting a cross-referencing analysis of the literature.

This paper is organized in four sections. The first section is introduction that describes the importance of NFRs in the software development. The second section describes the research approach and source of information. Findings from investigation are described in section three. Then, section 4 gives a conclusion, discussion and future works by highlighting some open issues which are acquired from this investigation.

2. RESEARCH METHODOLOGY

The investigation was conducted from 182 sources of information published over the last three decades. Majority of them are articles from academic resources within the discipline of software engineering in general and requirements engineering in particular (e.g. journal paper, articles from conference proceeding, and IEEE/ISO standard), and a few are industrial reports (e.g. technical reports and white papers). All of these articles cover various issues of NFRs, as illustrated in Figure 1. The starting point for selection of the papers to be reviewed was the study conducted by Chung et al. [1]. The detail composition of software engineering literature investigated for this research is presented in Table 1.

Table 1 - Sources of Information

Type of Literature	Number of Literature
Journal	78
Conference Paper/Proceeding	70
Book	16
Others (technical report, standard, etc)	18

Each article was then analyzed systematically using content analysis technique [26, 27]. Content analysis was selected because it enables researchers to identify trends and patterns in the literature through the frequency of key words, and by coding and categorizing the data into a group of words with similar meaning or connotations [27, 28]. This technique is also applicable to all domain contexts [26, 29].

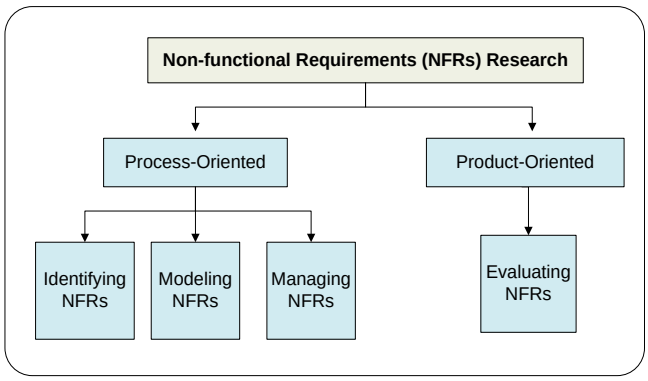


Figure 1 - NFRs Research

Three essential dimensions were defined as the research baseline: (1) definition and terminology, (2) types, and (3) NFRs in various types of systems and application domains. For the first dimension, definition and terminology, each of definition and terminology that were presented in literature were catalogued. Then, the similarity and dissimilarity aspects among them were analyzed. From this investigation, the different perspectives of how software engineering community considers NFRs and terminologies introduced to represent NFRs in each perspective were visualized.

For the second dimension, types of NFRs, all types from the catalogue of NFRs types were collected. Their definition and attribute[1] were recorded. Then, frequency analysis for each type in order to identify the more commonly considered NFRs, i.e. NFRs that are frequently listed in the catalogues of NFRs types, was conducted.

For the last of those three dimensions, NFRs in various types of systems and application domains, a mapping was created between types of systems (as well as application domains) and the types of NFRs considered in each type of system (or application domain). In this study, five different types of systems and their relevant NFRs were identified, while for the application domain, a well-known application domain taxonomy from Digital's Industry Taxonomy [30, 31] was adopted.

3. FINDINGS

With respect to the research questions, findings in this investigation can be categorized into three groups: definition and terminology; types; the relevant NFRs in various types of systems and application domains.

[1]In this paper, the term attribute is considered as the major components of each NFRs type. In the literature, attribute is also referred as quality subfactors [32-34] or NFRs subtypes [1].

3.1 Definition and Terminology

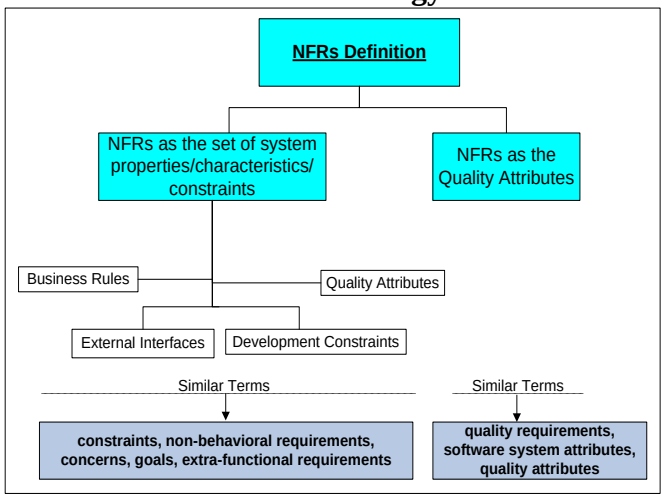


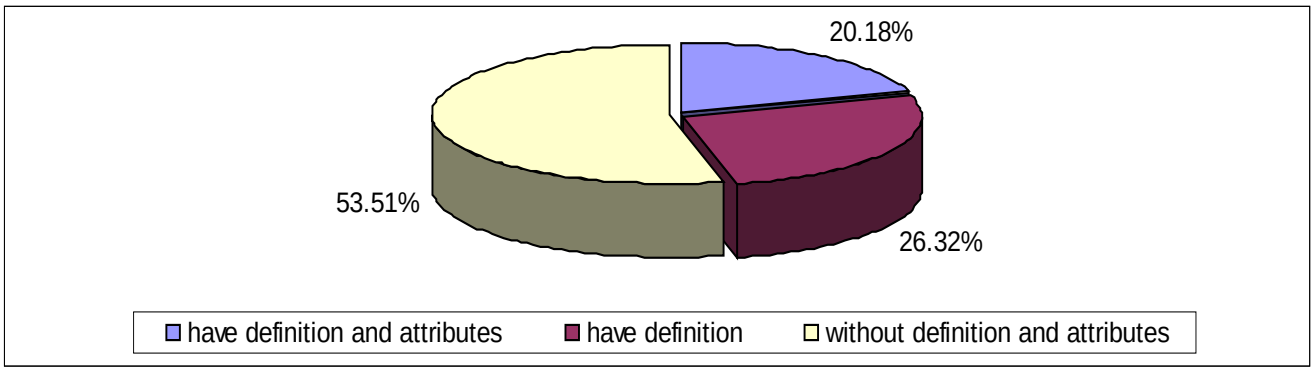
Figure 2 - Definition and Terminology NFRs

Figure 2 shows the result of investigation about definition and terminology NFRs in literature. This figure illustrates that generally, the term NFRs is considered for two different perspectives: (1) NFRs as the requirements that describe the properties, characteristics or constraints that a software system must exhibit; and (2) NFRs as the requirements that describe the quality attributes that the software product must have.

In the first perspective, NFRs consist of several aspects, such as development constraints, business rules, external interfaces, quality attributes, and any other requirements that do not describe the functionality of the system. The term constraints, nonbehavioral requirements, concerns, goals, and extra-functional requirements are also used to represent NFRs in this perspective. The second perspective takes the narrow focus of NFRs by only considering the quality attributes. Therefore, this perspective is the subset of the first perspective. The term quality requirements, software system attributes, and quality attributes are also used to represent NFRs.

1. Accessibility/Access Control
2. Accountability
3. Accuracy
4. Adaptability
5. Additivity
6. Adjustability
7. Affordability
8. Agility
9. Analyzability
10. Anonymity
11. Atomicity
12. Attractiveness
13. Auditability
14. Augmentability
15. Availability
16. Certainty
17. Changeability
18. Communicativeness
19. Compatibility
20. Completeness
21. Complexity/Interacting Complexity
22. Composability
23. Comprehensibility
24. Comprehensiveness
25. Conciseness
26. Confidentiality
27. Configurability
28. Conformance
29. Consistency
30. Controllability
31. Correctness
32. Customizability
33. Debuggability
34. Decomposability
35. Defensibility
36. Demonstrability
37. Dependability
38. Distributivity
39. Durability
40. Effectiveness
41. Efficiency/Device Efficiency
42. Enhanceability
43. Evolvability
44. Expandability
45. Expressiveness
46. Extendability
47. Extensibility
48. Fault/Failure Tolerance
49. Feasibility
50. Flexibility
51. Formality
52. Functionality
53. Generality
54. Immunity
55. Installability
56. Integratability
57. Integrity
58. Interoperability
59. Learnability
60. Legibility
61. Likeability
62. Localizability
63. Maintainability
64. Manageability
65. Maturity
66. Measurability
67. Mobility
68. Modifiability
69. Nomadicity
70. Observability
71. Operability
72. Performance/Efficiency/ Time or Space Bounds
73. Portability
74. Predictability
75. Privacy
76. Provability
77. Quality of Service
78. Readability
79. Reconfigurability
80. Recoverability
81. Reliability
82. Repeatability
83. Replaceability
84. Replicability
85. Reusability
86. Robustness
87. Safety
88. Scalability
89. Security/Control and Security
90. Self-Descriptiveness
91. Simplicity
92. Stability
93. Standardizability/ Standardization/Standard
94. Structuredness
95. Suitability
96. Supportability
97. Survivability
98. Susceptibility
99. Sustainability
100. Tailorability
101. Testability
102. Traceability
103. Trainability
104. Transferability
105. Trustability
106. Understandability
107. Uniformity
108. Usability
109. Variability
110. Verifiability
111. Versatility
112. Viability
113. Visibility
114. Wrappability

Figure 3 - The List of NFRs Types



have definition and attributes

accessibility; adaptability; availability; efficiency; fault tolerance; functionality; integratability; integrity; maintainability; modifiability; performance, portability; privacy; readability; reliability; reusability; robustness; safety; scalability; security; testability; understandability; and usability

have definition

accuracy; analyzability; attractiveness; changeability; communicativeness; completeness; complexity; composability; confidentiality; consistency; correctness; defensibility; dependability; evolvability; extendability; flexibility; immunity; installability; interoperability; learnability; likeability; localizability; maturity; operability; quality of service; recoverability; replaceability; stability; suitability; survivability

without definition and attributes

accountability; additivity; adjustability; affordability; agility; anonymity; atomicity; auditability; augmentability; certainty; compatibility; comprehensibility; comprehensiveness; conciseness; configurability; conformance; controlability; customizability; debuggability; decomposability; demonstrability; distributivity; durability; effectiveness; enhanceability; expandability; expressiveness; extensibility; feasibility; formality; generality; legibility; manageability; measurability; mobility; nomadicity; observability; predictability; provability; reconfigurability; repeatability; replicability; self-descriptiveness; simplicity; standardizability; structuredness; supportability; susceptibility; sustainability; tailorability; traceability; trainability; transferability; trustability; uniformity; variability; verifiability; versatility; viability; visibility; wrappability

Figure 4 - NFRs Definition and Attributes

3.2 Types

Our investigation on the types of NFRs in literature resulted in identifying 252 types of NFRs. Generally these NFRs consist of quality attributes (e.g. maintainability, performance, and reliability); development constraints (e.g. timing, cost, and development personnel); external interfaces requirements (e.g. user interface & human factors, look & feel, and system interfacing); business rules (e.g. production life span), and others (e.g. cultural, political, and environmental). Among these 252 types, 114 types correspond to the NFRs definitions that have been discussed specifically in relation to "the quality". The list of these 114 NFRs types is presented in Figure 3.

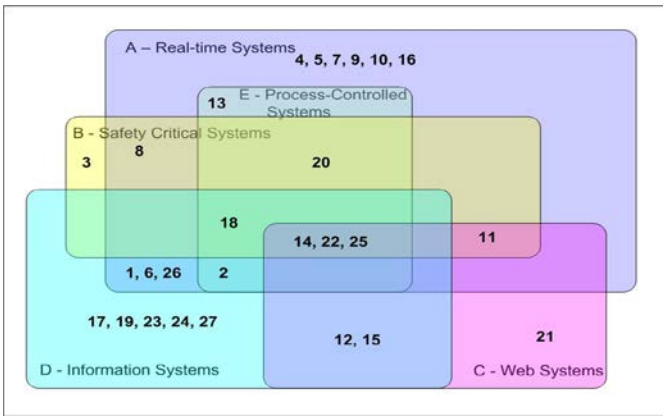
Further investigation to the NFRs types list shows that 23 types of NFRs (20.18%) have definition and attributes, 30 types (26.32%) only have definition, and the rest 61 types (53.50%) were introduced without definition or attributes. The detail list of NFRs in each of this classification is illustrated in Figure 4.

Furthermore, the result of frequency analysis indicates that performance (88.68%); reliability (67.92%); usability (62.26%); security (60.38%); and maintainability (54.72%) are the top five of the most frequent types of NFRs listed in the NFRs catalogue[2]. The detail definition and attributes of these top five NFRs are presented in Table 2. These definitions and attributes are decomposed by integrating several definitions and NFRs attributes based on general complementary description stated in the scholarly literatures. The investigation also shows that some types of NFRs are also recognized as the attribute of the other NFRs. For example, integrity, availability, and confidentiality are those NFRs which also become the attributes of security. Therefore, in one place those three NFRs are considered as NFRs while in another place they are also considered as the attributes of the other NFRs.

[2] We refer to these NFRs as the most commonly considered NFRs.

3.3 NFRs, Types of Systems and Application Domains

In this section, mapping between each type of system and its relevant NFRs as well as between each application domain and its relevant NFRs are presented.



Legend:

1 Accuracy	10 Installability	19 Reusability
2 Availability	11 Integrity	20 Safety
3 Communicativeness	12 Interoperability	21 Scalability
4 Compatibility	13 Maintainability	22 Security
5 Completeness	14 Performance	23 Standardizability
6 Confidentiality	15 Privacy	24 Traceability
7 Conformance	16 Portability	25 Usability
8 Dependability	17 Provability	26 Verifiability
9 Extensibility	18 Reliability	27 Viability

Figure 5 - Type of Systems and Relevant NFRs

3.3.1 NFRs and Types of Systems

From the investigation, five types of systems with their relevant NFRs were identified. These are real-time systems; safety-critical systems; web systems; information systems; and process controlled systems. Mapping between each type of system and its relevant NFRs is illustrated in Figure 5.

As shown in Venn diagram (Figure 5), three types of NFRs: performance, security, and usability are NFRs that are considered in all five types of systems, while reliability is a type of NFRs considered in four types of systems (real-time systems, safety-critical systems, information systems, and process-controlled systems). It indicates that the former three NFRs are the most common NFRs in each type of software being developed.

3.3.2 NFRs and Application Domains

By adopting a well-known software application domain taxonomy from the Digital's Industry Taxonomy [30, 31], in this investigation eight different application domains were considered: banking and finance; education; energy resources; government and military; insurance; medical/health care; telecommunication services; and transportation. Mapping between each software application domain and its relevant NFRs is presented in Table 3.

Further analysis of Table 3 shows that performance and usability requirements are considered in almost all application domains (seven out of eight domains); security is considered in six domains; confidentiality is considered in five domains; and accuracy and reliability are considered in four domains. Therefore, findings from section 3.3.1 and 3.3.2 indicate that performance and usability are the most commonly considered NFRs in various types of systems and application domains.

Table 2 - The Most Commonly Considered NFRs

NFRs	Definition	Attributes
Performance	requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions	response time, space, capacity, latency, throughput, computation, execution speed, transit delay, workload, resource utilization, memory usage, accuracy, efficiency compliance, modes, delay, miss rates, data loss, concurrent transaction processing
Reliability	requirements that specify the capability of software product to operates without failure and maintains a specified level of performance when used under specified normal conditions during a given time period	completeness, accuracy, consistency, availability, integrity, correctness, maturity, fault tolerance, recoverability, reliability, compliance, failure rate/critical failure
Usability	requirements that specify the end-user-interactions with the system and the effort required to learn, operate, prepare input, and interpret the output of the system	learnability, understandability, operability, attractiveness, usability compliance, ease of use, human engineering, user friendliness, memorability, efficiency, user productivity, usefulness, likeability, user reaction time
Security	requirements that concern about preventing unauthorized access to the system, programs, and data	confidentiality, integrity, availability, access control, authentication
Maintainability	requirements that describe the capability of the software product to be modified that may include correcting a defect or make an improvement or change in the software	testability, understandability, modifiability, analyzability, changeability, stability, maintainability compliance

Table 3 - Application Domains and Relevant NFRs

Application Domain	Relevant NFRs
Banking and Finance	accuracy, confidentiality, performance, security, usability
Education	interoperability, performance, reliability, scalability, security, usability
Energy Resources	availability, performance, reliability, safety, usability
Government and Military	accuracy, confidentiality, performance, privacy, provability, reusability, security, standardizability, usability, verifiability, viability
Insurance	accuracy, confidentiality, integrity, interoperability, security, usability
Medical/Health Care	communicativeness, confidentiality, integrity, performance, privacy, reliability, safety, security, traceability, usability
Telecommuni-cation Services	compatibility, conformance, dependability, installability, maintainability, performance, portability, reliability, usability
Transportation	accuracy, availability, compatibility, completeness, confidentiality, dependability, integrity, performance, safety, security, verifiability

4. DISCUSSION AND CONCLUSION

This paper presents the results of a systematic investigation of three essential dimensions of NFRs: (1) definition and terminology; (2) types; and (3) NFRs in various types of systems and application domains. Two different perspectives of how software engineering community considers the notion NFRs have been identified. Other similar terms to represent NFRs in each perspective have also been discussed. By conducting an extensive literature review, 252 types of NFRs have been identified where 114 of them are NFRs that have been discussed specifically in relation to the quality of the system. Among them, performance, reliability, usability, security, and maintainability are five of the most frequent NFRs listed in the NFRs catalogue. Mapping between NFRs and various types of systems as well as between NFRs and the application domains have also been presented as the paper's original contribution. From this study, performance, security, and usability are the most common NFRs considered in all five types of systems (real time systems, safety critical systems, web systems, information systems, and process-controlled systems) while performance and usability requirements are two NFRs that are considered in almost all application domains (seven out of eight application domains).

It is expected that findings presented in this paper would contribute to the software engineering research community in three ways: (1) to improve the understanding about the notion of NFRs; (2) to motivate the software engineering community to reach a consensus about several NFRs dimensions (e.g. definition, scope, terminology, types and granularity level of NFRs types and attributes, and the taxonomy of NFRs); and (3) the top five most considered NFRs presented in this paper (performance, reliability, usability, security, and maintainability) are expected to inform and motivate the research community to perform in-depth studies about these NFRs. Furthermore, these findings would benefit software developers in three ways. (1) The comprehensive list of NFRs types will let developers know what types of NFRs are there for the system being developed. (2) The matrix of relevant NFRs is expected to help developers to identify the important NFRs for their particular system being developed. Therefore, developers would be able to discover which NFRs should get attention in the project they are working on, depending on the type of system and/or the system application domain. For example, in the development of an embedded system, the catalogue of NFRs types as well as the matrix of relevant NFRs will help developers in identifying which NFRs need to be included in the software requirements specification. This matrix can act as a checklist which software developers can use to ensure that the system specification is complete with respect to the NFRs coverage. (3) This matrix can help the elicitation process by making sure that in the elicitation activity, those relevant NFRs have been discussed with the system stakeholders.

This study is conducted as part of a long term project of investigating conflicts among NFRs. The ultimate goal is to develop a framework to effectively identify and manage potential conflicts among them. Findings in this investigation will provide valuable insight into the mostly cited and investigated NFRs in the literature. The next step in this overall research project is to select those NFRs that are known to be frequently in conflict. Therefore the insight gained from the findings presented in this paper will assist in the selection of which NFRs to investigate for further research.

This study has two constraints: (1) the potential overlaps that exist among definitions and attributes of each NFRs were not investigated; (2) this study does not have the intention to create a structural hierarchy of NFRs types. These constraints will be considered for future research.

ACKNOWLEDGMENTS

We would like to thank The International Schlumberger Foundation for funding this research through Faculty for the Future Award Program.

REFERENCES

[1] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Massachusetts: Kluwer Academic Publishers, 2000.

[2] D. Firesmith, "Using quality models to engineer quality requirements," *Journal of Object Technology*, vol. 2, pp. 67-75, 2003.

[3] C. Ebert, "Putting requirement management into praxis: dealing with nonfunctional requirements," *Information and Software Technology*, vol. 40, pp. 175-185, 1998.

[4] R. T. Mittermeir, N. Roussopoulos, R. T. Yeh, and P. A. Ng, *Modern software engineering, foundations and*

current perspectives. New York, NY, USA: Van Nostrand Reinhold Co, 1989.

[5] G. Kotonya and I. Sommerville, *Non-functional requirements*, 1998.

[6] R. N. Charette, *Applications strategies for risk analysis*. New York: McGraw-Hill, 1990.

[7] K. E. Wiegers, *Software requirements*, 2nd ed. Washington: Microsoft Press, 2003.

[8] I. Sommerville, *Software Engineering*, 7 ed. Essex, England: Pearson Education Limited, 2004.

[9] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality Attributes," CMU/SEI-95-TR-021 ESC-TR-95-021 1995.

[10] K. K. Breitman, J. C. S. Prado Leite, and A. Finkelstein, "The world's a stage: a survey on requirements engineering using a real-life case study," *Journal of the Brazilian Computer Society*, vol. 6, pp. 1-57, 1999.

[11] A. Finkelstein and J. Dowell, "A comedy of errors: the London ambulance service case study," in *Eigth International Workshop Software Specification and Design*, 1996, pp. 2-5.

[12] D. R. Lindstrom, "Five ways to destroy a development project," *IEEE Software*, vol. 10, pp. 55-58, 1993.

[13] B. Boehm and H. In, "Identifying quality-requirements conflict," *IEEE Software*, vol. 13, pp. 25-35, 1996.

[14] N. G. Leveson and C. S. Turner, "An investigation of the Therac-25 accidents," *IEEE Computer*, vol. 26, pp. 18-41, 1993.

[15] H. In, "Conflict identification and resolution for software attribute requirements," in *Faculty of the Graduate School* vol. Doctor of Philosophy: University of Southern California, 1998.

[16] D. J. Grimshaw and G. W. Draper, "Non-functional requirements analysis: deficiencies in structured methods," *Information and Software Technology*, vol. 43, pp. 629-634, 2001.

[17] N. Heumesser, A. Trendowicz, D. Kerkow, H. Gross, and L. Loomans, "Essential and requisites for the management of evolution - requirements and incremental validation," Information Technology for European Advancement, ITEA-EMPRESS consortium 2003.

[18] N. Yusop, D. Zowghi, and D. Lowe, "The impacts of non-functional requirements in web system projects," *International Journal of Value Chain Management* vol. 2, pp. 18-32, 2008.

[19] G.-C. Roman, "A taxonomy of current issues in requirements engineering," *Computer*, vol. 18, pp. 14 - 23, 1985.

[20] J. Cleland-Huang, R. Settimi, O. B. Khadra, E. Berezhanskaya, and S. Cristina, "Goal-centric traceability for managing non-functional requirements," in *ICSE 2005* St. Louis, Missouri, USA: ACM, 2005.

[21] L. M. Cysneiros and J. C. S. do Prado Leite, "Nonfunctional requirements: from elicitation to conceptual models," *IEEE Transaction on Software Engineering*, vol. 30, pp. 328-350, 2004.

[22] S. Lauesen, *Software requirements: styles and techniques*: Addison-Wesley, 2002.

[23] B. Paech and D. Kerkow, "Non-functional requirements engineering - quality is essential," in *10th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2004, pp. 27-40.

[24] M. Glinz, "Rethinking the notion of non-functional requirements," in *Third World Congress for Software Quality*, Munich, Germany, 2005, pp. 55-64.

[25] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference (RE '07)*, 2007, pp. 21-26.

[26] K. Krippendorff, *Content analysis: and introduction to its methodology*, Second ed. Thousand Oaks, USA: Sage Publications, Inc., 2004.

[27] R. P. Weber, *Basic content analysis*: Sage Publications, Inc., 1989.

[28] S. Stemler, "An overview of content analysis," *Practical Assessment, Research & Evaluation*, vol. 7, 2001.

[29] K. A. Neuendorf, *The content analysis guidebook*, First ed.: Sage Publications, Inc., 2001.

[30] D. E. Corporation, *VAX VMS Software Source Book*: Maynard, Mass, 1991.

[31] R. L. Glass and I. Vessey, "Contemporary application-domain taxonomies," *IEEE Software*, vol. 12, pp. 63-76, 1995.

[32] D. Firesmith, "Security use cases," *Journal of Object Technology*, vol. 2, pp. 53-64, 2003.

[33] D. Firesmith, "Engineering safety requirements, safety constraints, and safety-critical requirements," *Journal of Object Technology*, vol. 3, pp. 27-42, 2004.

[34] D. Firesmith, "Specifying reusable security requirements," *Journal of Object Technology*, vol. 3, pp. 61-75, 2004.