Volume 4, Issue 7, July 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

NFR's: Definition

Harsimran Kaur
Assistant Professor
Depart of CSE
ASU, Haryana.

Abstract— The stakeholders realized that all Non-functional requirements are important and will get a place in the software architecture. In this paper we tried to identify different types of NFR's. Based on the literature survey we found that there are NFRs which have no definition yet. In this paper we tried to give informal definition to NFRs enlisted in [16].

Index Terms— Definition Attributes.

I. INTRODUCTION

Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements". The brilliant Greek computer scientist Diomidis Spinellis warned us about this conundrum in his book Code Quality "…a failure to satisfy non-functional requirements can be critical, even catastrophic …an insecure web-server or unreliable anti-lock brake system are worse than useless…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability or the absence of security vulnerabilities." Non-functional requirements are proved to be the reason of most modern applications failure. The continuing stream of business system disasters being reported in the press are rarely the result of a functional defect. Rather they are structural flaws in the system that we cluster under the catch-all phrase, 'non-functional requirements'. So these requirements needs to be stated from the beginning of the project as these are requirements. As Spinellis points out, non-functional requirements are a different phenomenon [7]. Not only will we have difficulty writing test cases to verify them, we cannot expect customers or analysts to even state them with any detail beyond simple descriptions of system behavior. In this paper we tried to list different NFR's and their description.

A single non-functional requirement may generate a number of related functional requirements. Non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed [16].

Dewi Mairiza et al. presents the result of an extensive and systematic analysis of the extant literature over three NFRs dimensions [16]: (1) definition and terminology; (2) types; and (3) relevant NFRs in various types of systems and application domains. The list consists of 114 NFR's in relation to quality as shown in Figure 1. Besides these NFR's there are other list too like Backup, Certification, Compliance, Configuration management, Dependency on other parties, Deployment, Documentation, Disaster recovery, Emotional factors, Environmental protection, Escrow, Exploitability, Resilience [16].

.

II. DEFINITION

In this paper we are defining all these 114 NFRs to understand their applicability. Some of the NFRs which are not defined yet in terms of software are defined on the basis of their application in other fields or with reference from the dictionary [16].

1. Accessibility can be viewed as the "ability to access" and benefit from some system or entity [58]. It is measured as

$$Accessibility_i = \sum_j Opportunities_j \times f(C_{ij})$$

where:

- i = index of origin zones
- j = index of destination zones
- $f(C_{ij})$ = function of generalized travel cost (so that nearer or less expensive places are weighted more than farther or more expensive places).

2. Accountability is answerability, blameworthiness, liability, and the expectation of account-giving. Accountability

© 2014, IJARCSSE All Rights Reserved

cannot exist without proper accounting practices; in other words, an absence of accounting means an absence of accountability [58].

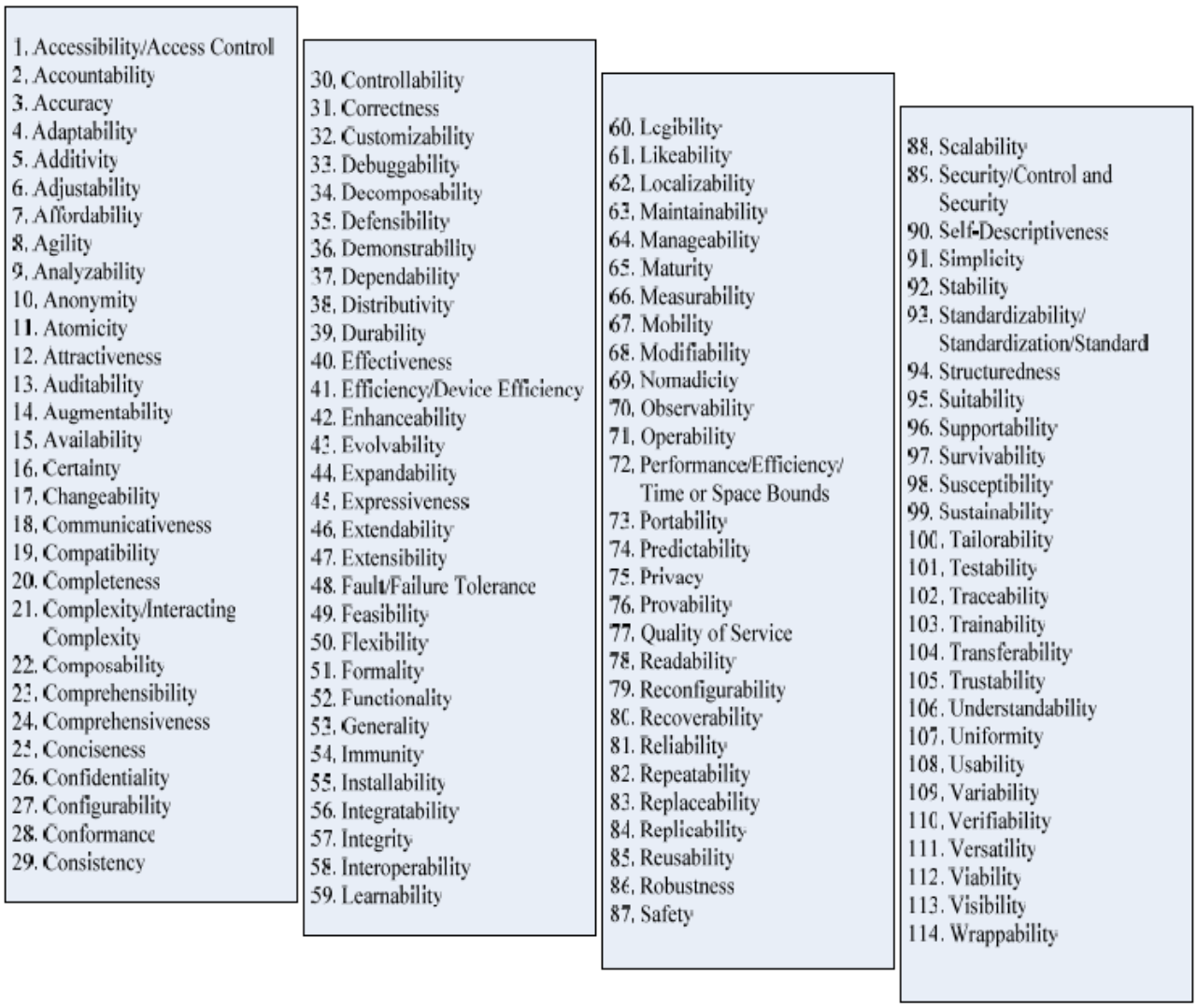

Figure 1: List of NFR's Types in relation to Quality [16]

3. The accuracy is the proportion of true results (both true positives and true negatives) in the population. It is a parameter of the test [57].

4. Adaptability is to be understood here as the ability of a system (e.g., a computer system) to adapt itself efficiently and fast to changed circumstances. An adaptive system is therefore an open system that is able to fit its behavior according to changes in its environment or in parts of the system itself. That is why a requirement to recognize the demand for change without any other factors involved can be expressed [19][40].

5. Additivity is the property to add in small amounts to something else to improve, strengthen, or otherwise alter it [58].

6. Adjustability : To settle or arrange; to free from differences or discrepancies. The term is sometimes used in the sense of pay, when used in reference to a liquidated claim. Determination of an amount to be paid to insured by insurer to cover loss or damage sustained [58].

7. Affordability: Conclusion drawn from the analysis of the 'life cycle cost' of a proposed acquisition, that the purchase is in accord with the resources and long term requirements ofthe acquirer[1].

8. Agility is defined by Roger Pressman as: Effective (rapid and adaptive) response to change, Effective communication among all stakeholders, Drawing the customer onto the team and organizing a team so that it is in control of the work performed [1].

9. A system's analyzability, in particular, is influenced by its decomposition into components. But into how many components should a system be decomposed to achieve optimal analyzability? And how should the elements of the

© 2014, IJARCSSE All Rights Reserved

system be distributed over those components? The ISO/9126 [56] standard for software quality defines analyzability, a sub-characteristic of maintainability, as: "the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified". Eric Bouwers et al. designed a metric called Component Balance to evaluate analyzability. $CB(S) = SB(|C|) \times CSU(C)$ The result of this function is a number in the range [0,1]. Higher values represent better component decomposition. System Breakdown (SB), which is designed to measure whether a system is decomposed into a reasonable number of components and Component Size Uniformity (CSU), which aims to capture whether the components are all reasonably sized [11].

10. Anonymity meaning "without a name" or "namelessness". Anonymous commercial transactions can protect the privacy of consumers. Tor is free software for enabling online anonymity. In recent years, a handful of anonymity metrics have been proposed that are either based on (i) the number participants in the given scenario, (ii) the probability distribution in an anonymous network regarding which participant is the sender / receiver, or (iii) a combination thereof [12][25].

11. Atomicity is one of the ACID transaction properties. In an atomic transaction, a series of database operations either all occur, or nothing occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright. In other words, atomicity means indivisibility and irreducibility. Atomicity allows the developer to selectively hide the complexity of concurrency by presenting simplified models or views of the system at certain stages of the development cycle. Serdar et al propose a LP Metric for concurrently accessed software components. The LP metric captures well common concurrency errors that lead to atomicity or refinement violations [8][34].

12. Attractiveness: Market size, projected growth, profitability, Fits with present businesses, Social, political, regulatory, environmental factors and Degree of risk / uncertainty contribute to system attractiveness. The sum of the weighted ratings provides a quantitative measure of the attractiveness relative to other industries and systems [58].

13. Auditability is the ease with which the interactions with a system by clients (users, other systems, donors, other organisations) can subsequently be inspected. Keeping audit logs of access to facilities, systems, and sub-systems may well be a legal requirement upon organisations managing data. It is an evaluation of the accuracy of figures reported for performance measure in the Budget or in the Service Efforts and Accomplishments report[1]

- Audited elements – what business elements will be audited?
- Audited fields – which data fields will be audited?
- Audit file characteristics - before image, after image, user and time stamp, etc

14. Augmentability: The software is augmentable is it is easy to add new functional capability or new data-it is easy to expand its capabilities without major redesign or modification in future [57].

15. Availability is the probability that the system is operational at any point in time [1][2][33][56]:

- Hours of operation – when is it available? Consider weekends, holidays, maintenance times, etc
- Locations of operation – where should it be available from, what are the connection requirements?
- Availability=UpTime/uptime+downtime

16. Certainty is having complete confidence that some decision is correct. When one is certain, all the evidence supports the decision made, and there is also confidence that if any further data were gathered, that data would support the decision too. The basis for gathering the evidence could be termed a model, and thus, one could say there is certainty about the model and the associated evidence, or data. It should be noted, however, that certainty does not preclude decisions being wrong [39][58].

17. Changeability: Change can be defined as the transition over time of a system to an altered state. If a system remains the same at time i and time i+1, then it has not changed.
A change event can be characterized with three elements: (1) the agent of change, (2) the
mechanism of change, and (3) the effect of change. Ross et al. quantify changeability by trade space exploration, which uses a parameterization of the system under consideration, can help to elucidate the broad comparison of the changeability of various system configurations [21].

18. Communicativeness: It means anything relating to communication. Measuring the effectiveness of communication activities is a critical component in the development of an effective communication strategy. The measurement process should include the following steps:

- Define measures of effectiveness and a measurement process to assess the results of various messages, channels,

© 2014, IJARCSSE All Rights Reserved

and delivery schedules
- Review measures of effectiveness and clarify if needed
- Implement measurement process
- Collect and analyze feedback data
- Assess success of communication activity
- Identify need for additional communication activities or improvements
- Incorporate feedback from measurements to continuously improve the communication process and activities throughout the transformation [2][29].

19. Compatibility: Compatibility decisions influence the success of a product. The answer to the compatibility dilemma depends on multiple factors: the size of the network, the number and the kinds of incumbents, the technology employed and so on [33].
- Compatibility with shared applications – What other systems does it need to talk to?
- Compatibility with 3rd party applications – What other systems does it have to live with amicably?
- Compatibility on different operating systems – What does it have to be able to run on?
- Compatibility on different platforms – What are the hardware platforms it needs to work on?

20. Completeness: Davis states that completeness is the most difficult of the specification attributes to define and incompleteness of specification is the most difficult violation to detect [52]. According to Boehm [2], to be considered complete, the requirements document must exhibit three fundamental characteristics: (1) No information is left unstated or "to be determined", (2) The information does not contain any undefined objects or entities, (3) No information is missing from this document. The first two properties imply a closure of the existing information and are typically referred to as internal completeness. The third property, however, concerns the external completeness of the document [36]. External completeness ensures that all of the information required for problem definition is found within the specification. This definition for external completeness clearly demonstrates why it is impossible to define and measure absolute completeness of specification because how could analysts know with certainty what is missing from the specification

21. Complexity: complexity is a major feature of computers software, and will increasingly be in the future. IEEE [56] defines software complexity as "the degree to which a system or component has a design or implementation that is difficult to understand and verify". Curtis [7] refines the concept of software complexity into algorithmic and psychological complexity. Algorithmic (or computational) complexity characterizes the run-time performance of an algorithm. Psychological complexity affects the performance of programmers trying to understand or modify a code module. Over the last decade many software complexity measures have been proposed by researchers [1][2][13][33].

22. Composability is a system design principle that deals with the inter-relationships of components. A highly composable system provides recombinant components that can be selected and assembled in various combinations to satisfy specific user requirements. In information systems, the essential features that make a component composable are self contained and stateless. The ability for a user to access data or access models is an important factor when considering composability metrics. If the user does not have visibility into a repository of models, the aggregation of models becomes problematic. Composability contending with the alignment of issues on the modeling level. Hyun Jung La et al. proposed two metrics: Inter-CaaS Composability(ICC)and CaaS-to-Application Composability(CAC) where Caas is Component as a Service.[3]

23. Comprehensiveness is user satisfaction key and must be quantified. In terms of web search engines comprehensiveness refers to the usefulness of the search results. In turn, a page can be useful if it is informative and/or novel, i.e., if the user actually learns something about the answer to the input query. One way to measure the relative comprehensiveness of a page is the dwell time distribution, where the dwell time is the time spent on the page by users. Pages abandoned too quickly may not present much satisfactory information [14][15].

24. Conciseness is one of the important factors for quality of interfaces. It means brevity and completeness. The entire underlying premise of "Elements of Style" [57].

25. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. Confidentiality metric measures the impact on confidentiality of a successfully exploited vulnerability. The possible values for this metric are None, Partial and complete. Increased confidentiality impact increases the vulnerability score[37][39].

26. Configurability is a unified, unintrusive, assume-nothing configuration system. It lets you keep the configuration for

© 2014, IJARCSSE All Rights Reserved

multiple objects in a single config file, load the file when it's convenient for you, and distribute the configuration when you're ready, sending it everywhere it needs to go with a single action. Configurability is one of the keystones to the success of any SaaS (Software as a Service) software [3].

27. Conformance is the Certification or confirmation that a good, service, or conduct meets the requirements of legislation, accepted practices, prescribed rules and regulations, specified standards, or terms of a contract. the conformance of components is a key quality criterion in determining their reusability. In spite of this, current industry practices of acquiring components are largely based on domain experts' experience and knowledge rather than a quantitative measurement of conformance. Woo Choi proposes Family Functional Conformance (FFC) = (number of represented functional features in component specification and correctly realized functional features in components) /(number of functional features in FRS) and Requirement Functional Conformance (RFC) (number of represented functional features in component specification and correctly realized functional features in components)/(number of functional features in requirement). He also proposes metrics for Standard conformance, Reference Model Conformance and Variability Conformance [28].

28. Consistency: Consistency is a fundamental feature in the software metrics. In order to improve consistency several organizations have developed revisions of the standard methods, that is, modifications to some aspects of the method which have an influence on the consistency, however, without modifying its application domain [39].

29. Controllability: A linear system is said to be completely controllable if, for all initial times t_0 and all initial states $\mathbf{x}(t_0)$, there exists some input function (or sequence for discrete systems) that drives the state vector to any final state $\mathbf{x}(t_1)$ at some finite time $t_1 > t_0$. Controllability determines the work it takes to set up and run test cases and the extent to which individual functions and features of the system under test (SUT) can be made to respond to test cases [46].

- What do we have to do to run our test cases?
- How hard (expensive) is it to do this?
- Does the SUT make it impractical to run some kinds of tests?
- Given a testing goal, do we know enough to produce an adequate test suite?
- How much tooling can we afford?

30. Correctness: Steve McConnell defines it as the degree to which a system is free from defects in its specification, design, and implementation. proving correctness has an important impact on program verification. It is well known that proving correctness and testing complete each other. There are some aspects (performance, for example) that can be verified only by testing [39].

31. Customizability is the ability for software to be changed by the user or programmer. Customizability indicates the built-in capability for supporting the customization and configuration of a component's internal functional features. Properties are used for component customization and configuration. Therefore, components must allow their property values to be changed programmatically or through some type of visual interface for their customization. The number of write methods provided by a component strongly affects the customizability of the component [58].

32. Debuggability: We need to have something which would improve our ability to debug code model issues. As we are lucking "dynamic" testing of editor in many cases this type of issues is reported by users. Frequently they are difficult to reproduce. So we need something that would help us to understand what was wrong without reproducing the issue [1].

33. Decomposability: A corollary of the decomposability requirement is division of labor: once you have decomposed a system into subsystems you should be able to distribute work on these subsystems among different people or groups. This is a difficult goal since it limits the dependencies that may exist between the subsystems: The most obvious example of a method meant to satisfy the decomposability criterion is top-down design. This method directs designers to start with a most abstract description of the system's function, and then to refine this view through successive steps decomposing each subsystem at each step into a small number of simpler subsystems, until all the remaining elements are of a sufficiently low level of abstraction to allow direct implementation [2].

34. Defensibility: For an exam program to have legal defensibility there must be evidence as to the test's quality that would stand up in a court challenge. You will need to be able to provide evidence that sound, professionally recommended guidelines were followed throughout the design, development, and maintenance of the software. Studies should also be conducted to investigate and confirm that the test has reasonable degrees of validity, reliability, and fairness. Among the most important elements that courts look for are a well-conducted job analysis and strong content validity (that is, the items need to have a high degree of "job relatedness"). Finally, good documentation of the design,

© 2014, IJARCSSE All Rights Reserved

development, and analysis of the software should be collected and maintained [58].

35. Demonstrability measures at each stage of the software development and deployment life cycle. There are demonstrable measures of quality from a business perspective. These measures are hard metrics that can be measured, trended and correlated to business performance like service availability and performance [2].

36. Dependability defined as the amount of reliance that can be placed on the service delivered by the system, a reliance that can also be justified to the user. It is a concept normally used in general and non-quantitive terms to describe computing systems and other systems, Jonsson has taken a step towards a quantitative understanding of it. This is done by merging aspects like reliability, performability, safety and security or parts thereof into a more general quality and defining a measure for it [2][15].

37. Distributivity: In abstract algebra and logic, distributivity is a property of binary operations that generalizes the distributive law from elementary algebra. In propositional logic, distribution refers to two valid rules of replacement. The rules allow one to reformulate conjunctions and disjunctions within logical proofs. A generalized notion of distributivity is proposed by Lucas Champollion, and formalized as a parametrized higher-order property called stratified reference: a predicate that holds of a certain entity or event is required to also hold of its parts along a certain dimension and down to a certain granularity [15].

38. Durability: In database systems, durability is the ACID property which guarantees that transactions that have committed will survive permanently. Durability can be achieved by flushing the transaction's log records to non-volatile storage before acknowledging commitment [58].

39. Effectiveness: The degrees to which objectives are achieved and the extent to which targeted problems are solved. In contrast to efficiency, effectiveness is determined without reference to costs. Most software and IT organizations have great difficulty measuring organizational efficiency and effectiveness, despite a bewildering array of metrics that have been proposed and occasionally used. However, a basic-yet-powerful set of metrics that gets to the heart of these issues does exist, and at the same time facilitates the application of Six Sigma. Tools such as a defect cost scorecard, defect containment effectiveness (DCE) and total containment effectiveness (TCE) metrics can be applied to manage differential effectiveness across phase of origin and detection. Measurement effectiveness is quantified by examining the extent to which the measurement process goals and objectives are met; and the extent to which managers utilize measurement information during decision-making [17].

40. Efficiency in general, describes the extent to which size, time, effort or cost is well used for the intended task or purpose. It is often used with the specific purpose of relaying the capability of a specific application of effort to produce a specific outcome effectively with a minimum amount or quantity of waste, expense, or unnecessary effort [33].

41. Enhanceability: The degree of ease with which a system can be enhanced [57].

42. Evolvability is defined as the capacity of a system for adaptive evolution. In software engineering area evolvability is the property of a software to be easily updated to fulfill new requirements. Most of the research on software evolution is focused on analyzing the properties of evolution on the source code level. The majority of these studies try to capture the properties of software evolution by analyzing the changes on the source code in release cycles of software systems. The research on this field has mainly considered size (number of modules) as a principal measure for evolvability [4]. Currently, ISO/IEC 9126 standard [56] derives metrics for evolvability based on the goal-question-metric (GQM) [4]. The steps that are taken during an evolution request act as the goal (e.g., analyze the current system).

43. Expandability: Computer system characteristic that is the ability to increase capability while retaining or increasing response time and throughput performance. Such a system can accommodate additions to its capacity and capabilities. On hardware, it is adding or increasing or improving hard disks, memory, or video boards .On software, it is supporting more network users, allowing a greater number of 'hits' from website visitors, or getting faster number crunching [33]

44. Expressiveness means the ability to say only what you want done rather than how you want it done: Among the things that contribute to expressiveness are:

- A lack of required syntactic sugar
- First-class functions
- Garbage collection

© 2014, IJARCSSE All Rights Reserved

- Dynamic typing
- The language core not being slavishly minimalistic
- Good functionality in the standard library

To some degree, the expressiveness of any language can be increased by shoving as much "how to do it" off into subroutines/objects as possible so that most of the remaining code is "what to do." The amount of "how to do it" code needed in the most abstract code is one measure of a language's expressiveness: The more the code looks like pseudocode, the more expressive it is of the programmer's intent [58].

45. Extendability: This principle dictates that our platforms' architecture must be fully functional for its intended environment as well as easily extendable to other environments for which it can be of value [13].

46. Extensibility is a system design principle where the implementation takes future growth into consideration. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change – typically enhancements – while minimizing impact to existing system functions. Extensibility can also mean that a software system's behavior is modifiable at run time, without recompiling or changing the original source code. For example, a software system may have a public Application Programming Interface that allows its behavior to be extended or modified by people who don't have access to the original source code. The extra functionality can be provided through either internally or externally coded extensions. Edson proposes the Extensibility metric and validate them empirically in [13].

47. Fault-tolerance or graceful degradation is the property that enables a system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naïvely-designed system in which even a small failure can cause total breakdown. Kannan and Parker define a metric for calculating the usefulness of fault-tolerance towards system performance in multi-robot teams [26][55].

48. Feasibility: A feasibility study is important because it drives the development of your project proposal, which can be presented to senior management to gain their commitment to the project and to obtain project funding. Furthermore, during the creation of the feasibility study one will often identify risks associated with the project, providing valuable input into the risk management efforts. Feasibility study should indicate[1][2]:

1. The implementation alternatives(not quantified yet)
2. The economic feasibility(quantifiable)
3. The technical feasibility(not quantifiable)
4. The operational feasibility(not quantifiable)
5. The political feasibility(not quantifiable)

49. Flexibility has been recognized as a desirable quality of software since the earliest days of software engineering. Classic and contemporary software design literature suggest that particular implementations are more flexible than others but stop short of suggesting objective criteria for quantifying such claims. Eden and Mens introduce the notion of evolution complexity and demonstrate how it can be used to measure and compare the flexibility of programming paradigms (object-oriented vs. procedural programs), architectural styles (Shared Data, Pipes and Filters, and Abstract Data Type) design patterns (Visitor and the Abstract Factory) [21].

50. Formality: The rigor of a software development project is achieved by setting rules into the process. There are various degrees of rigor. The highest level of rigor is formality – a situation where software systems can be verified by mathematical laws. Automated testing and error removals are possible. Much research has gone into applying formality in software development. A branch of Software Engineering research known as Formal Methods has been well debated and discussed in numerous Software Engineering conferences. However, practical application of formal methods has been limited [1][5].

51. Functionality is the sum or any aspect of what a product, such as a software application or computing device, can do for a user. A product's functionality is used by marketers to identify product features and enables a user to have a set of capabilities. Functionality may or may not be easy to use. Software functionality expressed in user requirements is a key element for the measurement and planning of the software process, the measurement of functionality requires a

© 2014, IJARCSSE All Rights Reserved

deep understanding of the object measured [1][2][33].

52. Generality can be characterized as a state or quality of being not limited to one particular case. Generalization, as an inductive process, collects information about a number of particulars and presents it in a single description. According to Navrat it is a transformation that moves entity e to a more general e' in such a way that all the features and dimensions of e' remains present in e, but at least some of the features or dimensions are less constrained in e'; Gfeatures (e) = e' Where, e' is e with some features/dimensions less constrained.

53. Immunity: Immunity-aware programming refers to programming techniques which improve the tolerance of transient errors in the program counter or other modules of a program that would otherwise lead to failure. Transient errors are typically caused by single event upsets, insufficient power, or by strong electromagnetic signals transmitted by some other "source" device [58].

54. Installability: The ability of a software component or system to be installed on a defined target platform allowing it to be run as required. Installation includes both a new installation and an upgrade. To be installable, a package must meet many criteria. Some are related directly to installation, while others cover the reconfiguration and use of the package after installation [33].

55. Integratability: Whether distributed applications A1, A2, ..., An, as their data transaction data belonging to disjunctive sets. Applications run on independent servers S1, S2, ..., Sn. Direct integration requires building a set of applications A1, A2, ..., An selected having regard to the following:

- homogeneity of the functional structure;
- security and confidentiality of data at a similar level of confidence;
- the level of quality reliability of software is equally sensitive;
- the user interface is simple easy to use;
- Integrator has an important mission, to verify that the necessary conditions are reached
- to achieve effective integration with high degree of utility.

We have to thoroughly test the following:

- use of quality indicators of integrating software;
- generating sets of test data;
- clear evidence of differences in the test [58].

56. Integrity[33]

- Fault trapping (I/O) – how to handle electronic interface failures, etc
- Bad data trapping - data imports, flag-and-continue or stop the import policies, etc
- Data integrity – referential integrity in database tables and interfaces
- Image compression and decompression standards

57. Interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer. Interoperability becomes a quality of increasing importance for information technology products as the concept that "The network is the computer" becomes a reality. For this reason, the term is widely used in product marketing descriptions. Products achieve interoperability with other products using either or both of two approaches:

- By adhering to published interface standards
- By making use of a "broker" of services that can convert one product's interface into another product's interface "on the fly"
- A good example of the first approach is the set of standards that have been developed for the World Wide Web. These standards include TCP/IP, Hypertext Transfer Protocol, and HTML. The second kind of interoperability approach is exemplified by the Common Object Request Broker Architecture (CORBA) and its Object Request Broker (ORB).

The approach selected for the quantification of interoperability was the development of a set of measures of performance (MOPs) and measures of effectiveness (MOEs). The MOPs/MOEs were integrated with a candidate set of components, which were used to partition the totality of interoperability into measurable entities [59].

58. Learnability: In software testing learnability, according to ISO/IEC 9126, is the capability of a software product to enable the user to learn how to use it. Learnability may be considered as an aspect of dusability, and is of major concern in the design of complex software applications. Nielsen also defines learnability as easy first time use but lists

© 2014, IJARCSSE All Rights Reserved

learnability as a sub component of the construct of usability[56].

59. Legibility is the degree to which glyphs (individual characters) in text are understandable or recognizable based on appearance.
"The legibility of a typeface is related to the characteristics inherent in its design ... which relate to the ability to distinguish one letter from the other. " Legibility includes factors such as "x-height, character shapes, stroke contrast, the size of its counters, serifs or lack thereof, and weight." Legibility is different from readability which refers to entire words, sentences, and paragraphs. Readability is influenced by line length, primary and secondary leading, justification, typestyle, kerning, tracking, point size, etc [58].

60. Likeability: Capability of the software product to be liked by the user. Educational software games aim at increasing the students' motivation and engagement while they learn. However, if software games are targeted to school classrooms they have to be usable and likeable by all students. Issue is discussed in paper by Maria and George [24].

61. Localizability often referred to a process that began after an application developer compiled the source files in the original language. Another team then began the process of reworking the source files for use in another language [57].

62. Maintainability

- Conformance to architecture standards – What are the standards it needs to conform to or have exclusions from?
- Conformance to design standards – What design standards must be adhered to or exclusions created?
- Conformance to coding standards – What coding standards must be adhered to or exclusions created?[33]

63. Manageability: How efficiently and easily a software system can be monitored and maintained to keep the system performing, secure, and running smoothly. Software manageability is one of the most overlooked features of a software system during the construction phase of many software development projects. While the largest part of a software system's life cycle is spent in managing and administrating the system, most software hasn't been developed with the specific needs of a system administrator in mind. The reasons for the omission of a manageability solution in most software products are twofold. Firstly, software manageability is hardly ever considered during the requirements analysis and design of a system. Stakeholders for the manageability of a system, such as the system administrators, are seldom involved in the system definition. A second reason for the lack of well manageable applications is a relative lack of knowledge of technical standards that deal with manageability. For applications developed on the Java platform, the JMX specification is the standard of choice for adding manageability features onto an existing system [21] .

64. Maturity: A metric in IEEE 982.1-1988 M = number of modules in current version, A = number of added modules in current version, C = number of changed modules in current version, D = number of deleted modules in current version compared to the previous version. SMI = (M - (A + C + D)) / M

More perspicuously, SMI = 1 - N/M, where M is the total number of modules in the current version of the system and N is the number of modules added, changed or deleted between the previous version and this one. SMI can be a measurement of product stability, when SMI approaches 1.0 the product is stable. When correlated with the time it takes to complete a version of the software, you have an indication of the maintenance effort needed. The maturity that the CMM for software is speaking about is process maturity. SMI is a metric for product maturity. A mature product is not necessarily a stable product. And this is an index of stability [56][58].

65. Measurability: There are a variety of reasons for measuring the software development process and product. Measurement is a mechanism for creating a corporate memory and an aid in answering a variety of questions associated with any software development. It helps support project planning, (e.g., how much will a new project cost?); allows us to determine the strengths and weaknesses of the current process and product, (e.g., are certain types of errors commonplace?); provides a rationale for adopting/refining techniques, (e.g., what techniques will minimize current problems?); allows us to assess the impact of techniques, (e.g., does functional testing minimize certain error classes?); evaluate the quality of the process/product, (e.g., what is the reliability of the product after delivery?) and the functionality and user friendliness (e.g., to determine if the system is easy to use and does what the user wants it to do.)

66. Mobility represents a total meltdown of all the stability assumptions (explicit or implicit)
associated with distributed computing. The network structure is no longer fixed, nodes may come and go, processes may move among nodes, and even programs (the code executed by processes) may evolve and change structure. Roman et al discusses research issues mobility poses on SE [45].

67. Modifiability: Belady explains the problem of software modifiability is two-fold: that of modifying existing systems

© 2014, IJARCSSE All Rights Reserved

and of designing new systems which are easier to modify [21].

68. Nomadicity: There are a number of compelling reasons why nomadicity is of interest. For example, nomadicity is clearly a newly emerging technology that users are already surrounded with. Indeed, this author judges it to be a paradigm shift in the way computing will be done in the future. Information technology trends are moving in this direction. Nomadic computing and communications is a multidisciplinary and multi-institutional effort. It has a huge potential for improved capability and convenience for the user. At the same time, it presents at least as huge a problem in interoperability at many levels. Leonard discuss the vision of nomadicity, its technical challenges, and approaches to the dresolution of these challenges [45].

69. Observability: From Merriam Webster dictionary, observability is defined as capable of being observed. Observed itself is defined as to watch carefully especially with attention to details or behavior for the purpose of arriving at a judgment [57]. [60] define observability factors of software engineering by making literature research on prior works and to assess the OSS engineering processes using the observability factors to show that this approach is working and can improve the software quality.

70. Operability is the ability to keep an equipment, a system or a whole industrial installation in a safe and reliable functioning condition, according to pre-defined operational requirements. In a computing systems environment with multiple systems this includes the ability of products, systems and business processes to work together to accomplish a common task such as finding and returning availability of inventory for flight. It is closely related to reliability, supportability and maintainability [58].

71. Performance is measured by [33]:

- Response times - application loading, screen open and refresh times, etc
- Processing times – functions, calculations, imports, exports
- Query and Reporting times – initial loads and subsequent loads [24]

72. Portability refers to the ability of a software unit to be ported (to a given environment). A program is portable if and to the degree that the cost of porting is less than the cost of redevelopment. The principal role of metrics in relation to portability issues is to help characterize the costs and benefits of incorporating portability in a software design, or of porting an existing software unit [51].

73. Predictability can be considered as the fundamental property of any process. It determines how accurately the outcome of the following that process in a project can be predicted before the project is completed

74. Privacy: Measuring privacy risk in online social networks is a challenging task. One of the fundamental difficulties is quantifying the amount of information revealed unintentionally [18]. Tools like PrivAware are available to measure the privacy.

75. Provability is the quality or state of being provable [57].

76. Quality of Service parameters are a key factor in the roll-out of new technology. ETSI works on several QoS specifications and has been particularly active in Interoperability events on speech quality. QoS parameters are increasing in importance as networks become interconnected and a large number of operators and providers interact to deliver communications [38].

77. Readability: Aggarwal [61]claims that source code readability and documentation readability are both critical to the maintainability of a project. The readability of a program is related to its maintainability, and is thus a critical factor in overall software quality. Paper reviews and construct an automated readability measure and show that it can be 80% effective, and better than a human on average, at predicting readability judgments [22].

78. Reconfigurability is defined as the ability to repeatedly change and rearrange the components of a system in a cost-effective way. This concept is illustrated through its application in computing, automated assembly and robotics. Setchi shows shown that there are common research issues in reconfigurable computing, robotics and manufacturing such as system-module-component interfaces, design methodologies, modularity, tools and tool suites development, strategic analysis and business modelling, training, and support [3].

79. Recoverability can be measured as [33]:

- Recovery process – how do recoveries work, what is the process?
- Recovery time scales – how quickly should a recovery take to perform?

© 2014, IJARCSSE All Rights Reserved

- Backup frequencies – how often is the transaction data, set-up data, and system (code) backed-up?
- Backup generations - what are the requirements for restoring to previous instance(s)?

80. Reliability -Software reliability is the probability that the software system will function properly without failure over a certain time period. Realiability metrics include:

Probability Of Failure On Demand (POFOD) is likelihood that a transaction request will fail. Rate Of Occurrence Of Failures (ROCOF) corresponds to the failure intensity. Mean Time To Failure (MTTF) is the average time between consecutive system failures. Availability is the likelihood that the system will be working at a given time. Mean Time To Recovery: if broken, how much time is available to get the system back up again [46] [50].

81. Repeatability or test-retest reliability is the variation in measurements taken by a single person or instrument on the same item and under the same conditions. A less-than-perfect test-retest reliability causes test-retest variability [56].

82. Replaceability is the capability of the product to be used in place of another specified product for the same purpose in the same environment [56].

83. Replicability is regrettably not a priority for Software Engineering researchers and, moreover,not afforded by many published studies. Louridas and Gousios raises the issue of the dangers of inadequate reproducibility [5].

84. Reusability is the degree to which a thing can be reused. To achieve significant payoffs a reuse program must be systematic. As organizations implement systematic software reuse programs to improve productivity and quality, they must be able to measure their progress and identify the most effective reuse strategies. This is done with reuse metrics and models which are explained in [3][14][36] [43]

85. Robustness is just one of the many facets of software development that make reading, modifying, updating and managing code easier. The Rule of Robustness [ii] states that robustness results from transparency and simplicity [10][21].

86. Safety: Software cannot by itself cause death or injury, software failure modes can give rise to system-level hazards. A software failure can result in an effect propagation path that traverses the host system and human operational system to cause hazards for users and the environment. Emphasis is placed on accuracy of the specifications of the required software behavior, on establishing that delivered software meets the specifications and that any side effects involved are acceptable. Safety Measurement map is issued in the paper by PSM Safety and security TWG[14].

87. Scalability is how system performance grows with adding more resources or, alternatively, how the resource utilization grows with increasing load. The dimensions of scalability include performance economics, physical size, addressing, software independence, communication ability, technology independence, and optionality. All must be considered in the design if an architecture is to be successful over a significant time period [49].

88. Security/privacy are often measured in terms of Information Assurance (IA)
and on the basis of :

- Login requirements - access levels, CRUD levels
- Password requirements - length, special characters, expiry, recycling policies
- Inactivity timeouts – durations, actions[18].

89. Self Descriptiveness: Software is an artifact hence it cannot have the kind of self that humans have. Concept is referred in paper with reference to semantic database search [19].

90. Simplicity: The design must be simple, both in implementation and interface. It is more important for the implementation to be simple than the interface. Simplicity is the most important consideration in a design[2]

91. Stability: One of the most desirable qualities attributes, yet hardest to achieve, is stability. A stable basis provides a foundation for building quality software. According to the vision behind introducing software stability, a stable model should be a must for the modeling of any software system [37].

92. Standardizability: A definition or format that has been approved by a recognized standards organization or is accepted as a de facto standard by the industry. Standards exist for programming languages, operating systems, data formats,

© 2014, IJARCSSE All Rights Reserved

communications protocols, and electrical interfaces.

93. Structuredness: A well-known property of process models is that of block-structuredness, meaning that for every node with multiple outgoing arcs (a split) there is a corresponding node with multiple incoming arcs (a join) such that the subgraph between the split and the join forms a single-entry-single exit (SESE) region. The key finding is that structuredness is not an absolute desideratum vis-a-vis for process model understandability [8].

94. Suitability: It is the quality or state of being especially suitable or fitting Research proposes a model that captures the evolution of the quality of a software product, and provides reliable forecasts of the end quality of the software being developed in terms of product suitability. It presents a Bayesian network for software quality [35].

95. Supportability is the capability of supporting a software system over its whole product life. This implies satisfying any necessary needs or requirements, but also the provision of Software Support covers the whole software life-cycle once it enters into service. In particular, it covers the following key aspects associated to the software: Operation, Logistics Management and Modification.

96. Survivability: A system that can repair itself or degrade gracefully to preserve as much critical functionality as possible in the face of attacks and failures is called a survivable system. This report summarizes an architecture and software mechanisms to make software applications based on the popular Object Services Architecture (e.g., OMGs CORBA or Java/RMI/Jini) model far more survivable than is currently possible, while at the same time maintaining the flexibility and ease of construction that characterizes OSA-based applications [42].

97. Susceptibility of a material or substance describes its response to an applied field. Moving to denser semiconductor technologies at lower voltages will cause an increase in transient errors. Alan etal. investigate the trends in transient errors and the susceptibility of operating systems and applications to them, and introduce ideas regarding software transient error recoverability [41].

98. Sustainability aspects can be brought to bear both during the development and use of software systems. The sustainability of software is affected by both technical and non-technical issues. Technical issues tend to focus on how reusable the software is - i.e. its potential for adaptation - while non-technical issues include how a project is governed and funded. Many approaches to measuring the sustainability of software rely on a set of qualitative measures that examine the implementation of the software and its management processes. However, qualitative measures are problematic, as they allow personal opinion to influence the results [19].

99. Tailorability refers to capability of an artifact (e.g. a pedagogical text or a tool) of being adapted to a given purpose or function. Software tailorability is achieved in terms of component software. Concept is explained in paper in context of distributed software application [44].

100. Testability can be defined by applying the concepts of observability and controllability to software. Observability refers to the ease of determining if specified inputs affect the outputs; controllability refers to the ease of producing a specified output from a specified input. Observability and controllability properties are already used for assessing the testability of hardware components. A metric for testability is also proposed in paper [6].

101. Traceability: Many facets of a software project can be traced; the term traceability is used to refer to requirements traceability throughout. Paper attempts to address this need by studying the factors that make traceability important and discusses the challenges facing traceability practices in industry [43].

102. Trainability is ability to be trained [57].

103. Transferability is one-to-one act of selling an ownership interest without having to acquire permission from other parties. For example The GPL by itself includes unrestricted transferability, but under defined strict terms which seem to suit the needs well. the trustability approach can be used to determine strategies in which methods are combined for maximum effectiveness. It can be used to determine the minimum amount of resources needed to guarantee a required degree of trustability, and the maximum trustability that is achievable with a given amount of resources [58].

104. Trustability: A program p has trustability T if we are at least T confident that p is free of faults. Trustability measurement depends on detectability. The detectability of a method is the conditional probability that it will detect faults [50].

105. Understandability: When programmers try to reuse a software system developed by other programmers, the

© 2014, IJARCSSE All Rights Reserved

difficulty of understanding the system limits reuses [1]. It is not easy to measure software understandability because understanding is an internal process of humans. Lin proposes "integral of understandability" as a model for measuring software understandability which can extract first the best value of software understandability from factors with higher weight.

106. Uniformity means Conforming to one principle, standard, or rule. Software development should be consistent. [2]

107. Usability in the context of creating software represents an approach that puts the user, rather than the system, at the center of the process. This philosophy, called user-centered design, incorporates user concerns and advocacy from the beginning of the design process and dictates that the needs of the user should be foremost in any design decisions. Usability evaluation is a core component of user-centered systems design and an essential competency for Human Factors professionals working in the software domain.eg user interface usability is measured by [33][43]:

- Look and feel standards - screen element density, layout and flow, colors, UI metaphors, keyboard shortcuts
- Internationalization / localization requirements – languages, spellings, keyboards, paper sizes, etc

108. Variability is the ability of a software system or artifact to be changed, customized or configured for use in a particular context. Variability is captured through defining the required features of the software and defining the various variations that these features could hold. WISE uses its expertise in information systems in providing a knowledge management based solution to the software variability problem [13].

109. Verifiability is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. Verifiability depends on modularity, self-description and simplicity.

110. Versatility is defined as a set of properties (such as variability, reuse, dynamism and usability) that allows the customization, extension and compression of middleware [31].

111. Viability is the ability of a thing (a living organism, an artificial system, an idea, etc.) to maintain itself or recover its potentialities [58].

112. Visibility means the ability to measure progress or status against goals. In software engineering, one encounters the visibility principle mainly in the form of process visibility, and then mainly in the form of schedule visibility: ability to judge the state of development against a project schedule. Quality process visibility also applies to measuring achieved (or predicted) quality against quality goals. The principle of visibility involves setting goals that can be assessed as well as devising methods to assess their realization. Visibility is closely related to observability, the ability to extract useful information from a software artifact [1].

113. Wrappability: It means enclose (an object) completely in any flexible, thin material such as fabric or paper [58].

III. CONCLUSION

Mairiza et al. investigated that 114 NFRs only 23 types of NFRs have definition and attributes, 30 types only have definition and 61 types were introduced without definition or attributes [16].

With reference to Glinz [32] classification in his paper we would categorize NFRs (which are without attributes) on the basis of their representation. On the basis of representation the NFRs can be Operational, Quantitative, Qualitative and Declarative. In future work we will show the definition and quantification of enhancebility of software (NFR). According to [16] Enhancebility is one of the NFR having no definition and attributes defined. Enhancebility is one of the important attribute along with availability, reliability to design self adaptive software [19].

We will quantify it with the help of Predicate calculus which facilitates goal refinement, helps to identify conflicts and design tradeoffs.

REFERENCES

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

[2] Ian Sommerville 2006.6th edition Ed. Addison Wesley,

[3] La, Hyun Jung; Her, Jin Sun; Kim, Soo Dong, "Framework for evaluating reusability of Component-as-a-Service (CaaS)," Principles of Engineering Service-Oriented Systems (PESOS), 2013 ICSE Workshop on , vol., no., pp.41,44, 26-26 May 2013.

[4] Joel Lehman and Kenneth O. Stanley. Evolvability is inevitable: Increasing evolvability without the pressure to adapt. PLoS ONE, 8(4):e62186, April 2013.

© 2014, IJARCSSE All Rights Reserved

[5] Panos Louridas and Georgios Gousios. 2012. A note on rigour and replicability SIGSOFT Softw. Eng. Notes, 37(5):1–4.

[6] Pratima Singh and Anil Kumar Tripathi, "ISSUES IN TESTING OF SOFTWARE WITH NFR", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.4, July 2012.

[7] B.curtis blog " non functional requirements be here " on Consortium for IT Software Quality 2012

[8] Dumas, M., La Rosa, M., Mendling, J., Mäesalu, R., Reijers, H.A., Semenenko, N.: Understanding Business Process Models: The Costs and Benefits of Structuredness. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 31–46. Springer, Heidelberg (2012)

[9] S. Tasiran, M. E. Keremoglu, and K. Muslu, "Location pairs: a test coverage metric for shared-memory concurrent programs," Empirical Software Engineering (ESE), vol. 17, no. 3, 2012.

[10] Huanjing Wang; Khoshgoftaar, T.M.; Wald, R., "Measuring robustness of Feature Selection techniques on software engineering datasets," Information Reuse and Integration (IRI), 2011 IEEE International Conference on , vol., no., pp.309,314, 3-5 Aug. 2011.

[11] Eric Bouwers, José Pedro Correia, Arie van Deursen, Joost Visser, "Quantifying the Analyzability of Software Architectures," wicsa, pp.83-92, 2011 Ninth Working IEEE/IFIP Conference on Software Architecture, 2011

[12] Addicam Sanjay et al white paper on Metrics for Anonymous Video Analytics on Digital Signage for intel corporation 2011.

[13] Oliveira Junior, E. A., Gimenes, I. M. S., Maldonado, J. C.Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures. In: Proceedings of the Fourth Brazilian Symposium on Software

[14] Ali dasdan "Web Search Engine Metrics" WWW 2010, ACM, April 26–30, 2010, Raleigh, North Carolina, USA Components, Architectures, and Reuse, pp. 31-40. (2010)

[15] Champollion, L. (2010). Parts of a whole: Distributivity as a bridge between aspect and measurement. PhD Thesis, University of Pennsylvania.
2010

[16] Dewi Mairiza et al. , An investigation into the notion of non-functional requirements, Proceedings of SAC'10, Pages 311-317, ACM New York, NY, USA ©2010

[17] Peter Baxter Measurement Process Effectiveness, white paper distributive management 2010

[18] J. Becker and H. Chen. Measuring privacy risk in online social networks. In Web 2.0 Security and Privacy Workshop, 2009.

[19] Mazeiar Salehie , Ladan Tahvildari, Self-adaptive software: Landscape and research challenges, ACM Transactions on Autonomous and Adaptive Systems (TAAS), v.4 n.2, p.1-42, May 2009 [doi>10.1145/1516533.1516538]

[20] A Kannenberg, H Saiedian Why Software Requirements Traceability Remains a Challenge The Journal of Defense Software Engineering 01/2009; 22:14-19.

[21] Ross AM, Rhodes DH, Hastings DE (2008) Defining changeability: reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. Syst Eng 11(3):246–262

[22] Raymond P.L. Buse , Westley R. Weimer, A metric for software readability, Proceedings of the 2008 international symposium on Software testing and analysis, July 20-24, 2008, Seattle, WA, USA

[23] Gruia-Catalin et al. Software Engineering for Mobility: A Roadmap"The Future of Software Engineering" , Anthony Finkelstein (Ed.), ACM Press 2000

[24] Maria Virvou on usability and likieability of virtual reality games for education: The case of VR-engagae.in jornal of computer and education Volume 50 Issue 1, January, 2008 Pages 154-178

[25] C. Andersson and R. Lundin, "On the fundamentals of anonymity metrics," in The Future of Identity in the Information Society, ser. IFIP International Federation for Information Processing. Springer Science & Business Media, 2008.

[26] B. Kannan and L. E. Parker. Fault-tolerance based metrics for evaluating system performance in multi-robot teams. In Proceedings of Performance Metrics for Intelligent Systems Workshop, 2006.

[27] Da--wei,, E..: The Software complexity model and metrics for object--oriented,, IEEE International Workshop on Anti--counterfeiting,, Security,Identification,, 464--469,, 2007..

[28] Woo Choi "A Metric-based Approach to Measure Conformance in Components" Pacific Science Review, vol.9, no.1, 2007, pp.67~71.

[29] Tasiran, S., et al.: A novel test coverage metric for concurrently-accessed software components. In: Grieskamp, W., Weise, C. (eds.) FATES 2005. LNCS, vol. 3997, pp. 62–71. Springer, Heidelberg (2006)

[30] J. Davies, C. Crichton, E. Crichton, D. Neilson, Ib H. Sørensen Formality, evolution, and model-driven software engineering ,in: Alexandre Mota, Arnaldo Moura (Eds.), Proceedings of SBMF 2004, ENTCS (2005)

[31] Silva Filho RS, Redmiles DF (2005) A survey on versatility for publish/subscribe infrastructures. Technical Report UCI-ISR-05-8, (Institute for Software Research), Irvine, CA, pp. 1–77.

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

[33] Steve Easterbrook ppt on NFR University of Toronto 2004-2005

[34] Jörg Kienzle On Atomicity and Software Development Journal of Universal Computer Science, vol. 11, no. 5 (2005), 687-702

© 2014, IJARCSSE All Rights Reserved

[35] Beaver, J.M.; Schiavone, G.A.; Berrios, J.S., "Predicting software suitability using a Bayesian belief network," Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on , vol., no., pp.7 pp.,, 15-17 Dec. 2005

[36] Washizaki, H.; Yamamoto, H.; Fukazawa, Y., "A metrics suite for measuring reusability of its software components "Proceedings. Ninth International , vol., no., pp.211,223, 3-5 Sept. 2003

[37] Ahmed Mahdy and Mohamed E. Fayad,: A Software Stability Model Pattern,. DOI: http:// www.hillside.net/plop/plop2002/final/StabilityPatternPLoP02Correct.pdf

[38] M. Wirsing and A. Knapp. View consistency in software development. In Proceedings of the 9th International Workshop on Radical Innovations of Software and Systems Engineering in the Future (RISSEF), Venice, Italy, October 2002.

[39] Didar Zowghi and Vincenzo Gervasi. The Three Cs of Requirements: Consistency, Completeness, and Correctness. In Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ'02), 2002.

[40] Nary Subramanian , Lawrence Chung, Software architecture adaptability: an NFR approach, Proceedings of the 4th International Workshop on Principles of Software Evolution, September 10-11, 2001, Vienna, Austria [doi>10.1145/602461.602470]

[41] Alan Messer et al "Susceptibility of Modern Systems and Software to Soft Errors" HP Laboratories Palo Alto in march 2001

[42] Wells, D.; Ford, S.; Langworthy, D.; Wells, N., "Software survivability," DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings , vol.2, no., pp.241,255 vol.2, 2000

[43] Totter, A. And Stary, C. Tailorability and usability engineering: a roadmap to convergence. User Interfaces for All: Proceedings of the 5th ERCIM, Dagstuhl, Germany. 1999

[44] D. Theotokis et al., "Distributed Information Systems Tailorability: a Component Approach," Distributed Computing Systems, Proceedings 7th IEEE Workshop on Future Trends, 1999, pp. 95-101.

[45] Leonard Kleinrock, Nomadicity: anytime, anywhere in a disconnected world, Mobile Networks and Applications, v.1 n.4, p.351-357, Dec. 1996

[46] W. Frakes and C. Terry,Software Reuse: Metrics and Models, ACM Computing Surveys, vol. 28, pp. 415-435, 1996.

[47] Nikolay Mehandjiev , Leonardo Bottaci, User-Enhanceability for Organizational Information Systems through Visual Programming, Proceedings of the 8th International Conference on Advances Information System Engineering, p.432-456, May 20-24, 1996

[48] Lawrence Chung , Brian A. Nixon, Dealing with non-functional requirements: three experimental studies of a process-oriented approach, Proceedings of the 17th international conference on Software engineering, p.25-37, April 24-28, 1995, Seattle, Washington, United States

[49] Gustavson, D.B., "The many dimensions of scalability," Compcon Spring '94, Digest of Papers. , vol., no., pp.60,63, Feb. 28 1994-March 4 1994doi: 10.1109/CMPCON.1994.282944

[50] Howden, W.E.; Yudong Huang, "Software trustability," Software Reliability Engineering, 1994. Proceedings., 5th International Symposium on , vol., no., pp.143,151, 1994.

[51] Mooney, J.D. Issues in the Specification and Measurement of Software Portability. Technical Report TR 93-6, Dept. of Statistics and Computer Science,West Virginia University, Morganto

[52] Davis AM., "Software Requirements: Analysis and Specification", Prentice Hall, second edition, (1993).

[53] Jefferey M. Vaos "Factors that affect software Testability " NASA Langley Reasearch center,1991.

[54] N. E. Fenton Software Metrics, A Rigorous Approach, 1991 :Chapman & Hall

[55] T. J. Shimeall and N. G. Leveson "An empirical comparison of software fault-tolerance and fault elimination", IEEE Trans. Software Eng., pp.173 -183 1991

[56] *IEEE* Std 610.12-1990

[57] Dictionary www.techterms.com

[58] www.wikipedia.org

[59] Sun.java.com.

[60] Rogers, E. Diffusion of Innovations. Free Press, New York, 1983

© 2014, IJARCSSE All Rights Reserved