1. Using Automated Tests for Communicating and Verifying Non-functional Requirements

Summary:
Non-functional requirements are often communicated and verified by writing guidelines as a result it is costly. When the developer is working on a software, his focus is on a single module which we can say one functional requirement at a time. So, in between time he has no time to remember about the non-functional requirement. The easy way to communicate and verify non-functional requirement is to add them to the tool-chain as automated tests and checkers. This shows the experience of implementing and observing the use of automatic verification of the non-functional requirements by using the tool-chain feedback in a number of cases. As a result, productivity is increased and number of non-compliant nonfunctional requirements is lower using the tool-chain feedback instead of using guidelines and reports.

2. Automated Testing of functional Requirements

Summary:
An open-source toolkit offers a substitute to current approaches (testing suites) by being lighter and flexible through reusing libraries and executables found in common operating systems and environments enables automated testing of functional requirements. The toolkit provides developers and project owners and managers with reports about the system under development. Significant risk reductions are experienced in projects where the toolkit prescribed in research paper has been used.

3. Software Testing for Web-Applications Non-Functional Requirements.

Summary:
The main issues of web applications are load and performance testing. To ensure their performance ability and load testing, web applications should be able to answer a user's request in appropriate time and solve other user’s issues simultaneously. It should not have broken links and must be able to navigate to the whole website from the homepage.

4. A Framework for Non-Functional Testing Process of Mobile Applications

Summary:
There is a considerable growth in the acquisition of mobile devices with different characteristics; under this, it is very important to emphasize the phases of the mobile application testing process. Proposes a framework for non-functional testing process where the non-functional requirements that should be considered for mobile application tests, test case scenarios (TCS) and tools to test and execute these TCS. They allow to control and ensure the quality of the product. This results as a cost-benefit analysis according to the types of test execution, the validation and execution of the proposed framework and errors are identified before production.

5. Non-Functional Requirements Analysis Modeling for Software Product Lines

Summary:
Non-functional requirements (NFRs) are usually handled ad-hoc and during testing when it is late and costly to fix problems. Due to the importance of NFRs, study about the problem of modeling NFRs for Software Product Lines (SPL) is done. This paper is surveying the software engineering literature, searching for a methodical method to break down and plan NFR, from the points of view of the idea of shared trait and fluctuation of SPL and the qualities of NFR. Purposed solution is extension of PLUS to all NFRs categories and a reasoning engine to provide unified framework for analysis modeling.

6. Requirements Based Testing of Software

Summary:
Testing is the method that assures that the developed system conforms to the specification and contain no errors, by producing confidence into the system. This presents the Requirements Based Testing (RBT) technique for testing the Usability by considering that requirements are well written. The main emphasis is to test the usability and realibity. A template has been proposed for testing usability, which uses quantitative approach.

7. Automated Test Case Generation to Validate Software Requirements

Summary:

In practice, the validation of the functional and non-functional requirements is not given the same importance. Several techniques are presented that enable cost effective validation of non-functional requirements. Approach is targeting validation of performance properties. Cost effectiveness of performance validation by automatically generating load tests and subsequently introduced a compositional load test generation technique that targeted more complex software systems. Robustness and cost effectiveness of such validation is improved by using exception-handling constructs and by amplifying existing tests to exhaustively test every possible pattern in which exceptions can be raised respectively.

8. Non-Functional Requirements Research: Survey

Summary:

The survey has been presented on interesting ongoing work in the field of non-functional requirements, its testing and automation. Through this figure out the approaches and methods that are suggested in literature to deal with these issues. NFR needs to look after from the very start because many numbers of models are available for FR but not for NFRs. New variable can be introduced for NFRs from scratch. Paper also mentioned testing issues in the light of NFR. On solution, for the testing issue is to have aspect-oriented techniques (capture issues under verification).

9. Testing Non-Functional Requirements with Aspects

Summary:

Testing for non-functional requirements in an industrial software system is a unique challenge. In order to try a different approach, this study will be using aspect orientated techniques to test for non-functional requirements with a case study of an industry example. It is seen that due to modularity of the code and heavy instrumentation of SUT aspect orientation is an innovative way to accomplish this task.

10. A systematic review of search-based testing for Functional system properties

Summary:
Search based technique is application of search techniques to generate software tests. It identifies the size and complexity of the software. This technique shows the metaheuristic search technique for non-functional execution of time, security, safety, unitability and quality of service. It is mainly used for solving engineering problems to solve competing constraints.

11. Model Based Testing for Non-Functional Requirements

Summary:
Model Based Testing (MBT) is a new-age test automation technique with the capability of generating test cases by using model developed from the analysis of the abstract behavior of the System. This helps to model non-functional requirements. Various MBT tools will help one of the Ericsson's (telecommunication-equipment provider company) testing divisions to select the best tool for adapting to its existing test environment.

12. Agile Team Members Perceptions on Non-functional Testing

Summary:
Non-functional requirements are a challenge to incorporate and test. In agile development, it is required to cater these requirements. Due to the nature of these, identifying factors that influence the testing in performance and security aspects will be empirically sought out by interviewing IT professionals. This will enable to get a sense of which factors to look out for and how to handle them.

13. Rethinking the Notion of Non-Functional Requirements

Summary:
Non-Functional Requirements Testing enhances quality characteristics. These requirements can't be treated likely. New methods of Constraints and Object oriented exist, so that design decisions are captured in goal graph and reused by other models. Testing starts from requirement elicitation to design and code phase to lower down risk failure.

14. A Framework for Non-functional Testing of Compilers

Summary:
Compiler users apply different optimizations to generate efficient code with respect to non-functional. Due to the huge number of optimizations provided by modern compilers, finding the best optimization way is challenging. This paper proposes NOTICE (tool), a component-based framework for non-functional testing of compilers through the monitoring of generated code in a controlled environment. Evaluation is obtained by verifying the optimizations performed by the GCC compiler and this tool can automatically construct optimization levels that represent optimal trade-offs between multiple non-functional properties.

15. A Testability Analysis Framework for requirements

Summary:
Testability is a quality attribute that evaluates the effectiveness and efficiency of testing. This based on an in-depth analysis of available testability definitions, testability frameworks and work testability of non-functional properties. Testability issues investigated in the context of robustness are observability, controllability, automation, and testing effort. Following identifies the test sensitivity characteristics: component state, component interaction, resource limitations and availability, which determine whether testing interferes with the state of the running system in an unacceptable way as well as the test isolation techniques state separation, interaction separation, resource monitoring and scheduling that provide countermeasures for its test sensitivity.

16. Targeted Mutation: Efficient Mutation Analysis for Testing Non-Functional Properties

Summary:
Mutation analysis is computationally expensive so focuses on mutation effort to those parts of the code where a change can make a difference with respect to the targeted non-functional property. program slicing enables direct mutations to the parts of the code that are likely to have the strongest influence on execution time. This information can be used to increase the fault detection reflectiveness of the test suites. Test suites are run against the mutants to determine the percentage of mutants the tests will detect which refers to the mutation adequacy score that is a coverage criterion.

17. Automated verification of compliance of non-functional requirements on mobile applications through metamorphic testing

Summary:
Mobile application testing imposes several challenges and peculiarities, such as limited power, interface adaptation, and user's privacy. The current state of MT technique is being conducted to identify the main problems faced to establish testing strategies for the context of NFRs. Through Systematic mapping the process of data analysis, and some information related to security can be observed. This aims to automatize the verification of compliance of NFRs on mobile applications through MT. Neglecting such requirements is costly because difficult to fix the system if it has already been implemented.

18. Using Automated Tests for Communicating and Verifying Non-functional Requirements

Summary:
In software development much of the work is done in a tool-chain that contains tools like text editors, compilers, linkers, static analysis tools, automatic test frameworks. The architectural requirements often control the ability of the software to be reused and maintained in a cost-efficient way. When a developer is writing code, they often focus on implementing one functional requirement at the time. This requirement can be verified using automated tests and is often located in an isolated part of the software. The tool-chain analyzes and transforms of the code in many different ways and the tools can give fast feedback to the developer about the progress. This means that the tool-chain can be used to prevent mistakes from the developer.

19. Performance Testing based on Test-Driven Development of Mobile Applications

Summary:
Due to tough schedule for the development of mobile applications performance testing is mostly done at the end of the development, which has shown many problems. The importance of testing is emphasized in TDD (Test driven development) and the automated test framework is supported for efficient software development with unit tests. This paper purpose a methods of performance testing based on test-driven development with regard to non-functional factors as well as functionality of software during the software development process by advancing performance testing to the development

stage and introduce a testing tool that assists performance testing on software development phase. It provides automation of test case generation and test execution at unit test level.

20. Autonomous Compliance Monitoring of Non-Functional Properties

Summary:

Defining non-functional requirements can end up in a major undertaking consuming significant resources. Even after defining, they do not reflect back real-world usage. Differences are due to workload, hardware, or utilized third-party libraries. In order to solve these problems, they had proposed a fully automatic compliance monitoring solution for non-functional properties. The proposed system mines stable behavioral patterns of the system and automatically extracts assertions that can be used to detect deviations of expected non-functional behavior. They especially focus on non-functional properties that require runtime observation, e.g. execution time, performance, throughput.

21. A safety-aware, System-Based Approach to teaching Software Testing

Summary:

This paper describes an approach to teach software testing that addresses these two limitations (abstracting system from socio-technical system create false impression second practicing at small level creates problems) by adopting a system engineering paradigm, with an additional strong emphasis on safety. It consists of two modules. In the core module, students build fundamental testing knowledge in a software engineering context, and put this for both of developer and tester. In addition, in second project module, students get hands-on experience in engineering, and in particular testing, of microcontroller-based, safety-critical systems, for advising, guest lectures and project steering. They will get to know about the importance of NFR's and functional requirements

22. Requirement-Centric Reactive Testing for safety-Related Automotive Software

Summary:

This paper is about requirement centric reactive testing for safety related automotive software. Which is one of non-functional requirement type. They

proposed a test reactivity taxonomy for embedded automotive software that a tester can use to align the process of test design with the requirements specification and verification of the project at hand. This proposed test reactivity taxonomy can be used to explore, identify and design the right types of test reactivity, which target a specific functional or nonfunctional requirement or an application-specific situation. The taxonomy can also be used to document software-testing decisions and enable discussions between the relevant stakeholders.

23. Functional Requirements in Business Process Modeling

Summary:
This paper purposes how two new artifacts may be applied to model the constraints associated with a business process. This is the operating condition to denote a business process constraint and the control case to define controlling criteria to mitigate risk associated with an operational condition. Modeling constraints in this way provides an opportunity to capture these characteristics of business process early in the systems development life cycle. A key preliminary step in determining requirements is to identify the type and categories of FRs. The operating condition is the first (high-level) step to identify such business requirements.

24. The Interplay of Design and Runtime Traceability for Non-Functional Requirements

Summary:
Non-functional Requirements (NFRs) have a significant impact upon the architectural design and drive critical trade-offs. From a traceability perspective, it is thus necessary to trace individual NFRs into the design and their associated design rationales. When runtime data indicates that a mismatch has occurred that adversely affects system performance and create bottlenecks at various points. This paper had explored five different types of NFRs across the design and runtime phases of the development process. Their approach is illustrated by examples from the Dronology System for Fault Tolerance, Security, Usability, Performance, and other critical qualities.

25. A Testability Analysis Framework for requirements

Testability is a quality attribute that evaluates the effectiveness and efficiency of testing. This based on an in-depth analysis of available testability definitions, testability frameworks and work testability of non-functional properties. Testability issues investigated in the context of robustness are observability, controllability, automation, and testing effort. Following identifies the test sensitivity characteristics: component state, component interaction, resource limitations and availability, which determine whether testing interferes with the state of the running system in an unacceptable way as well as the test isolation techniques state separation, interaction separation, resource monitoring and scheduling that provide countermeasures for its test sensitivity.