Journal of Software Engineering and Applications, 2019, 12, 249-265
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Toward the Design and Implementation of Traceability Engineering Tool Support

Subik Pokharel*, Hassan Reza

Department of Computer Science, University of North Dakota, Grand Forks, North Dakota, USA
Email: *subik.pokharel@und.edu, hassan.reza@engr.und.edu

How to cite this paper: Pokharel, S. and Reza, H. (2019) Toward the Design and Implementation of Traceability Engineering Tool Support. *Journal of Software Engineering and Applications*, **12**, 249-265.
https://doi.org/10.4236/jsea.2019.126015

Received: January 26, 2019
Accepted: June 27, 2019
Published: June 30, 2019

Copyright © 2019 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Requirements of a system keep on changing based on the need of stakeholders or the system developers, making requirement engineering an important aspect in software development. This develops a need for appropriate requirement change management. The importance of requirements traceability is defining relationships between the requirements and artefacts extracted by the stakeholder during the software development life-cycle and gives vital information to encourage software understanding. In this paper, we have concentrated on developing a tool for requirement traceability that can be used to extend the requirement elicitation and identification of system-wide qualities using the notion of quality attribute scenarios to capture the non-functional requirements. It allows us to link the functional and non-functional requirements of the system based on the quality attribute scenarios template proposed by the Carnegie Mellon Software Engineering Institute (SEI). Apart from this, the paper focuses on tracing the functional and non-functional requirements of the system using the concept of requirement traceability matrix.

Keywords

Requirement Engineering, Requirement Engineering Tool, Tool Support, Quality Attributes, Requirement Traceability, Software Engineering

1. Introduction

The essential proportion of accomplishment of a software system is how much it meets the goal for which it was developed [1]. Failure of a software system to meet the needs of its users and its environment after it has been developed can cause serious problems on both the development team and the stakeholders. The development team must maintain the system and on the stakeholder's side, it may cost time and money for rework. Therefore, the teams developing these

software products must perform a rigorous risk analysis to distinguish possibly dangerous conditions, and their contributing factors before launching, or during the development of the software system [2]. Therefore, to design and develop any software system, requirement engineering assumes an essential job as it portrays the functional and non-functional requirements of the system software [3]. As crucial component of a software system, requirement engineering is a cyclic process of finding the software requirements, by recognizing the stakeholders and their needs and archiving these needs in a form that is manageable for analysis, correspondence, and resulting execution [1] [3].

The process of requirement engineering revolves around five main activities [4] [5]. Domain understanding means getting a decent comprehension of the space in which the issue is established, and what the underlying foundations of the issue are. Requirement elicitation is the activity of finding competitor prerequisites and presumptions that will shape the software to-be, founded on the shortcomings of the present software as they rise up out of area understanding. Requirement evaluation and negotiation deal with making educated choices about issues raised during the elicitation procedure. Requirement specifications deal with the thorough displaying of prerequisites, to give formal definitions to different parts of the software. Finally, verification and validation are concerned with checking the prerequisites record for consistency, culmination and precision of the system. The challenges faced during the research by the requirement engineering are distinguishable from those faced by the software engineers since requirements remain principally in the problem space whereas other software requirements reside basically in the solution space [1] [6]. Therefore, the development team faces several inherent challenges. Stakeholders might be from various fields and may shift and strife in objectives contingent upon their points of view of the environment they perform their tasks [1].

The types of requirements we are concerned with are the functional and non-functional requirements. Functional requirements deal with the functionality of a system and specifies what the system should perform under specified conditions [7] [8]. Some of the examples of the functional requirements are business rules, authentication and authorization levels, external interfaces, administrative functions, etc. These requirements depend on the type of software users are interested in and the nature of the environment where the software is expected to be deployed [8]. The functional system requirements of a software should be able to describe the system services in detail. A non-functional requirement defines how the system performs certain functionality under specified conditions [9] [10]. An example of this kind of requirement can be, the reloading feature of a web-page which should be performed within some fraction of a second. Non-functional requirements are referred to as the requirements ending with the string -ility or -ity or -ness [11] [12]. These include usability, modifiability, traceability, scalability, security, robustness and so on [13] [14].

The work presented in this paper is the continuation of our previous work [3]

[15] [16] which talks about the development of a requirement engineering tool and captures the requirements of a system. The key contribution of this paper is to implement the traceability between the requirements that were introduced in our previous work. The tool developed can be used to stimulate the domain understanding, and requirement elicitation and specification process for system qualities [3]. We have utilized the quality attributes template proposed by the Carnegie Mellons SEI, which helps the system architects and designers for the development of the system. In the next sections, we will go in detail about the traceability of requirements, followed by some related works and finally some details about the developed requirement engineering tool and the traceability between the requirements that are linked with one another.

2. Background

Computer systems are utilized as a part of numerous critical applications where a failure can have great consequences. Creating deliberate strategies to relate the software quality attributes of a system to the architecture of the system gives a sound premise to settling on target choices. This enables decisions for plan tradeoffs and empowers designers to make sensibly exact expectations about a system's attributes that are clear from predisposition, and shrouded presumptions [9]. Quality attributes give a strategy to examining a systems architecture against various critical quality attributes, for example, availability, performance, testability, usability, security, scalability, and modifiability that are gained from mission or business objectives [5]. Quality attributes drive the design of a system architecture.

Traceability between the development requirements and artefacts play a major role in the development of a system, for example, system validation, change impact analysis, and regulation compliance. The importance of requirements traceability is defining and using relationships between the requirements and artefacts extracted by the stakeholder during the software development lifecycle and gives vital information to encourage software understanding [17]. Traceability can be characterized as how much a relationship can be built up between at least two items of the developed requirements, particularly items having an antecedent successor or ace subordinate relationship to each other [18]. Tracing of requirements for software/system development can be focused at various viewpoints, such as system/software verification and validation, change administration, and administrative consistency. The significance of traceability has been broadly perceived, and it is training recommended in numerous progression standards [19]. The research on the field of traceability has significantly centered around prerequisites traceability, going for concentrate how to portray and take after the life of a prerequisite, in both forward and in reverse directions.

Traceability of requirement is characterized as the capacity to portray and pursue the life of a requirement, in both a forward and reverse way [18] [20], for example, stakeholder's needs, building segments, requirements, or source code.

Traceability of requirement is viewed as essential for setting and keeping up consistency between heterogeneous models utilized all through the advancement life-cycle. Every now and again detailed advantages of traceability of requirement incorporate elements such as the assistance of correspondence, bolster for the combination of changes, the conservation of plan learning, quality confirmation, and the counteractive action of false impressions. Tracing of information or requirements promotes developed system understanding and helps designers in managing basic issues in system advancement and support. For instance, architects may be occupied with the inceptions of a necessity (e.g., the stakeholder's requirement) or the justification for a specific design decision. They may likewise need to know how precisely functional or non-functional necessities are figured out in the system, or if a usage totally understands a given arrangement of requirements [17].

During the maintenance of the system, traceability of requirement is likewise vital for examining the effect of new requirements or changes to existing ones. It regularly experiences the huge exertion and many-sided quality of making and looking after follows, in spite this fact, procedures for producing and approving traceability of requirement are accessible. This outcome in invalid or inadequate trace data which cannot bolster engineers or architects in certifiable issues.

3. Related Works

Visure quality analyzer launched by Visual Requirements S.L. is one of the approaches to handle requirement engineering. This tool permits the user to define, measure, improve and manage the quality of each requirement, along with entire requirement specifications [21]. To generate the complete requirements for a project, the tool uses a user-customized process-meta model. This model captures all the processes required during the development stage in a diagrammatical fashion and links the components required for the design to one another.

Another tool is inteGREAT (Modern Requirements 4TFS), whose model attempts to "provide all partners with a typical perspective of prerequisites, prompting more exact, steady, and brought together fruition of projects over time." [22]. inteGREAT's answer gives determinability of necessities through "various prerequisites measurements" and traceability of necessity trait history, displaying of utilization cases, and reusability of past data sources. inteGREAT works for the most part through Microsoft Office items, enabling clients to produce necessities and exchange them to the inteGREAT stage. Some companies like Bright Green Projects, Leap SE, PACE, etc. provide services with different tools that middle around boundlessly unique inclinations and product configuration details [23].

Reza *et al.* [24] talk about a non-functional requirement tool utilizing the scenario-based approach. Based on the different styles, tactics, and quality of the system requirements, this tool allows its users to decide the architectural style for a system. The main limitation of this tool is that it only captures the requirement

but lacks the traceability between the requirements.

Quality attributes talked about in planning stages will, if effectively met, reinforce software development by giving designed system requirements to enhance the code lucidness, dependability, and more. Non-functional requirements incorporate modularity, security, adaptation to internal failure, and that's only the tip of the iceberg. These parameters don't really give usefulness to the developed system or software, yet careful quality in their advancement can manage the way the system may be changed, or even how it may respond in case of a breakdown.

The importance of non-functional requirements and criticality of the quality of their solutions increases while considering different software. Inability to effectively execute software that is both profoundly accessible and fault tolerant cannot be permitted in these situations. Inability to actualize safety efforts in software administering flying vessels can abandon it open to attacks or external control. An absence of exactness in the source-code can along these lines influence the accuracy of software designed to explore airship or shuttle, causing the flight direction of the vessel to be changed suddenly, similarly as with the Ariane 5 [25].

Egyed *et al.* [26] have presented a tool support technique using a video-on-demand system which facilitates requirement tracing by creating trace data automatically, and later show that it can be used in various engineering fields to solve requirement traceability problems. The key contribution of the work is that it decreases the huge exertion and unpredictability of procuring traces via automatically getting trace information from a little arrangement of clear speculated traces. This prompts more complete trace and the maximum capacity of requirement traceability can be exploited.

Pohl *et al.* [27] present an environment which enables the requirement pre-traceability during the development phase of the system. The work presented here is the continuation of their previous work, where in the current work they have re-implemented the work and address the scalability problems faced when running in the real application. **Figure 1** shows the main contribution of their work. In **Figure 1**, we can see the actors involved during the origin of the requirements for a system. This phase is called the requirement pre-traceability as not all requirements generated by the actors are involved in the system. Once the requirements are generated, the next step is the design and implementation of the requirements. Since this step is carried out by the development team, some requirements may be ignored due to conflicts or some may be added to the system. So, the requirement post-traceability is carried out in this step.

Cuddeback *et al.* [28] have introduced an experimental framework where they study human interactions with decision support systems. The work focuses on a group of people making decisions over a support software and was carried out at different universities. The participants analyzed the requirement traceability matrix for a Java code formatted program and based on the result, the authors

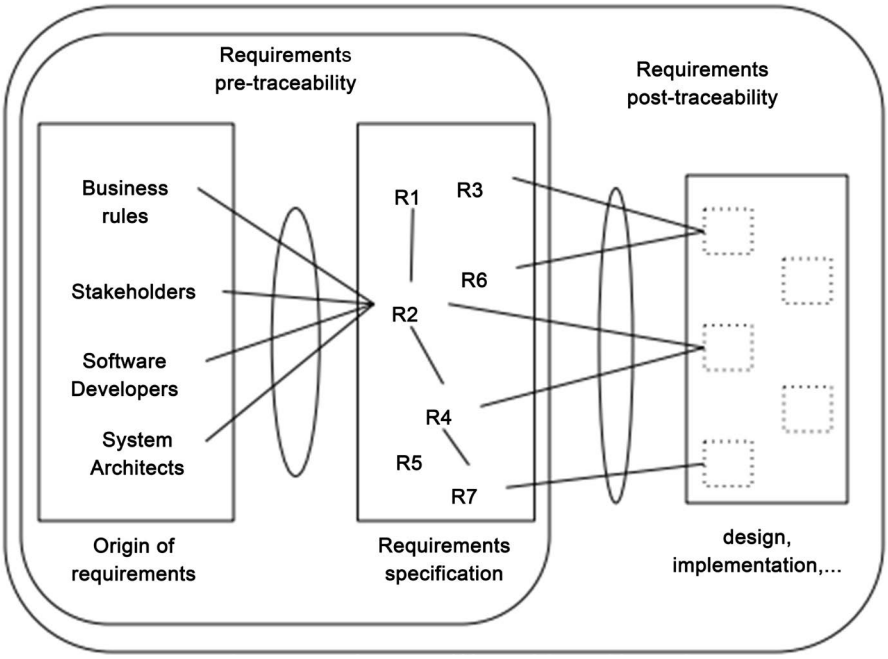

Figure 1. Generation of requirements during a system development [27].

presented the computation for accuracy of human interaction on automated requirement traceability. The automated tracing tool used is RETRO (REquirements TRacing On-target) with some modification in the back-end.

Gregoriades *et al.* [29] represent a tool (SRA) to support non-functional requirement in complex sociotechnical system. The tool proposed uses scenario-based testing to enhance the dependability and operational execution requirements for a system. The author of the paper [30] showed the motivations and concerns among broad scale circumstances, and a method named LSS, which uses robotized and semi-robotized systems to the portrayal, support and correspondence, with the usage of far-reaching computation circumstances in the field of requirement engineering.

Bashir *et al.* [31] give a survey of the existing techniques in the current domain of traceability. They evaluated the current traceability techniques and found that the existing techniques are inadequate and may cause problems while managing the changes in the system requirements. In the paper, the authors categorized the current techniques into three classes based on the utility of the techniques and argued that they can be combined and used for removing the shortcomings of one another to yield the highest benefits from requirement traceability. The first class incorporates the techniques with the system level scope, the second incorporates the software level scope and the third is a weak class that incorporates the software level scope.

4. Quality Tool and Its Capabilities

To implement a requirement for a system, multiple people from different fields may be involved. These people may be the stakeholders, developers, engineers,

architects, and so on, making it a difficult job as all the people involved are from multidisciplinary fields. The stakeholders set the functional requirement of the system but do not understand the approach to solve the problem. The members of the development team have knowledge about the tools used to develop and the techniques used for developing the system. Since we see a gap between the stakeholders and the development team, the author is needed to communicate between them, who creates a statement between them. Since the requirements of a system are changing from time to time as required by the stakeholders, there is a need for it to be tracked so that it does not affect the overall system by any means. To encourage the effective interchanges among the distinctive stakeholders, we require a tool which can monitor all functional and non-functional requirements with their connection to one another.

In our previous work [3], we developed a model-based requirement engineering tool that captured the functional and non-functional requirements of the system. The major work presented in the tool was capturing the non-functional requirements of the system using the quality attribute scenario template which was originally proposed by the Carnegie Mellon Software Engineering Institute (SEI) [5]. The capabilities of the tool presented in [3], was capturing the functional and non-functional requirements of the system along with the linking between them. The limitation of the tool [3] is that it just captures the requirements and if along the development phase of a system, if any of the requirements are changed, the user of the tool must go through all the requirements of the system and look for any major changes. In this paper, we add on to the previous work and introduce the traceability features. With this feature, if there are any changes to the requirements of a system, it gives the user a notification on the things they might want to investigate or make changes to.

The users of the software requirement tool developed maybe a member from the development team or the system architect. It allows the users to perform create, read, update and delete (CRUD) operations for the functional requirements, nonfunctional requirements and the system constraints. It also allows the users to link the requirements with one another. The links can be between the functional and non-functional or functional to functional or non-functional to non-functional requirements, which is shown in the **Figure 2**. This relation might vary from one-to-one up to many-to-many depending on the type of

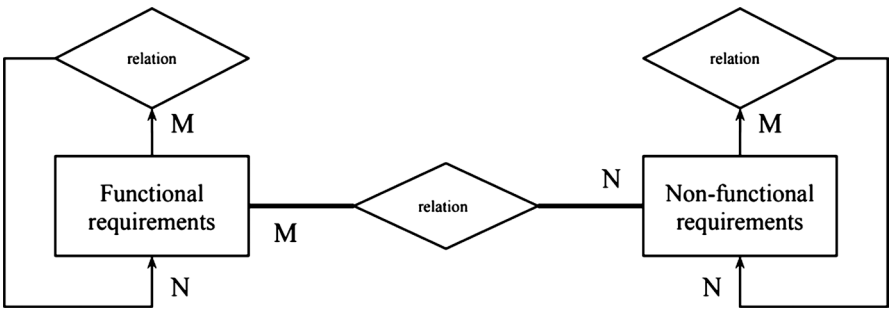

Figure 2. Relationship between the requirements in the system.

requirements. The system constraints for a project consist of the software requirements and the hardware requirements.

The detail of the schema definition developed from the entity relationship model is shown in Figure 3. It shows the different classes composed inside the tool. The class project contains attributes regarding the requirements captured into the tool. The class functional requirement captures the functional requirements of the project and similarly the class nonfunctional requirement captures the non-functional requirement of the project. To capture the non-functional requirements, we have used the quality attribute scenarios template proposed by SEI. As any requirements implemented into the system must be related to one of the projects, therefore both functional and non-functional requirement table stores a foreign key of the project id.

As discussed, and shown in Figure 2, the requirements may be linked with one another or to itself. To keep track of these links, we have implemented three relation classes. These classes store the ids of the requirements that are linked with one another along with the project id that they belong to or they are linked in. The details of how we capture the requirements in the tool are described below.

Figure 4 shows the window for creating project. It allows the user to insert the name of the project along with its start date, end date, and description of the

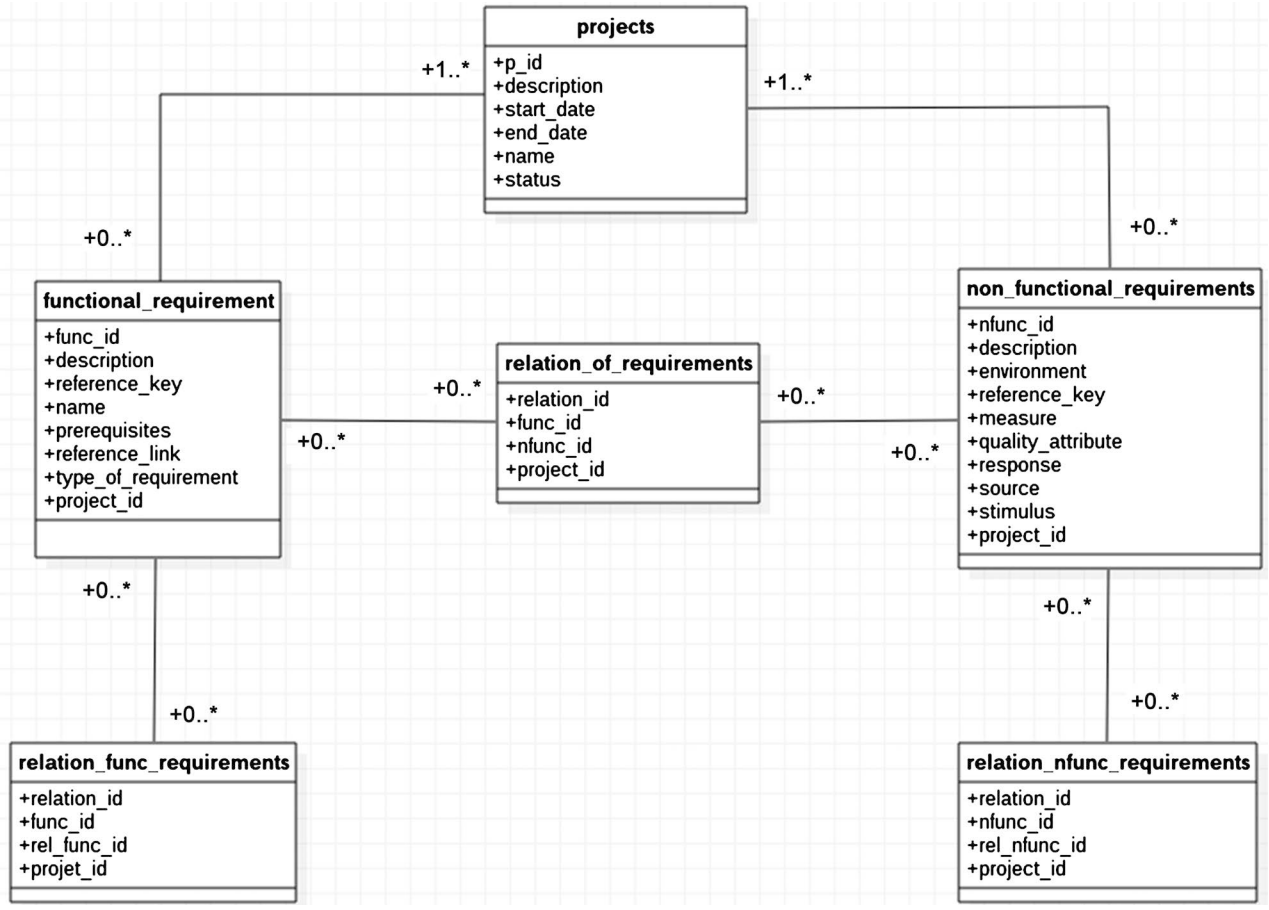

Figure 3. Schema definition of the database.

project. After inserting these details, the user can then enter the functional requirements, the non-functional requirements and the system constraints present in the left side of the window.

We have used the quality attribute scenarios to add the non-functional requirements for a system. The user can either enter or update the quality attributes such as availability, security, modifiability, performance, testability, and usability for a project using this template, which is shown in Figure 5. The source captures if the

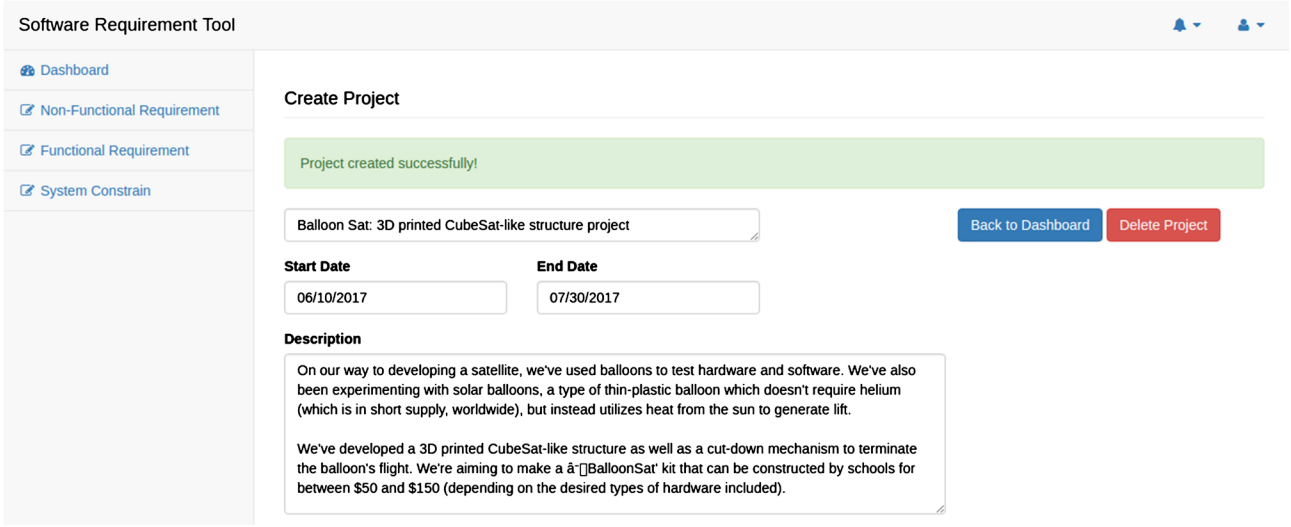

Figure 4. Template for creating and managing projects.

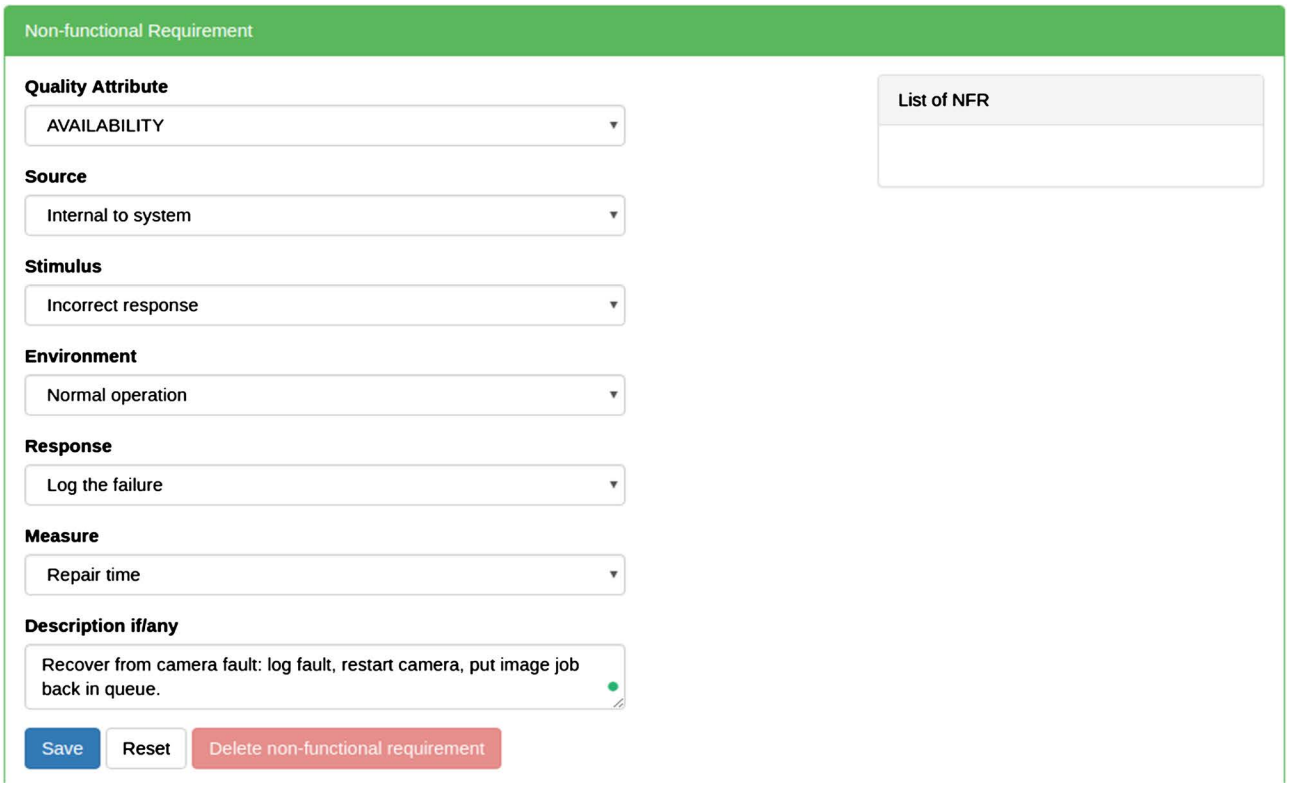

Figure 5. Template for creating and updating non-functional requirement.

requirement is internal or external to the system. The user can then enter the other descriptions as per the project referring to the quality attribute scenario template description [2]. The names for the different quality attributes are generated uniquely by the system and can be edited or deleted at any time by the user. If there is a relation between the other requirements, the user is notified about it and the user must check the integrity of data and remove its relationship with other requirements before removing it completely from the system.

Figure 6 shows the window for the functional requirements, where the user is prompt to enter the name, description, references on file that are related to this requirement, prerequisite before implementing it, and the relation with non-functional requirements. The functional requirement can be linked to another functional requirement. Also, in case of functional requirements, the process of edit and delete is the same as that of the non-functional requirements discussed and shown above.

Next the user can capture the system constraints, which is also an important aspect of a project along with the functional and non-functional requirements. The window for capturing the system constraints is shown in Figure 7, where the user enters the hardware, software and network requirement of the project. These are not related to any of the requirements but are linked to only projects, as the system constraints for a project is fixed and is independent to the requirements

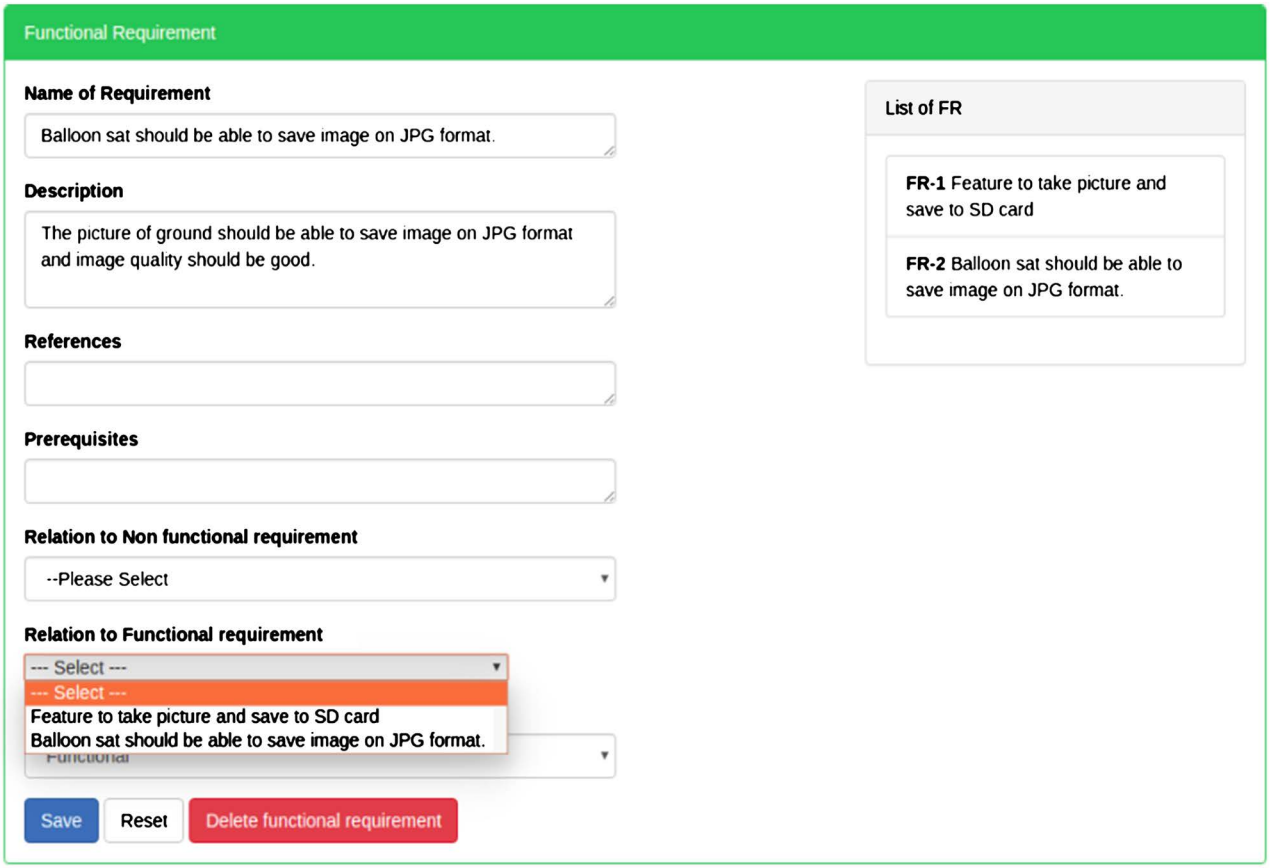

Figure 6. Template for creating and updating functional requirement.

changes within a project.

The user can view the list of ongoing and completed projects from the dashboard of the tool, shown in Figure 8. This window allows the user to view, edit and delete the projects. On clicking the view button, the user will be able to

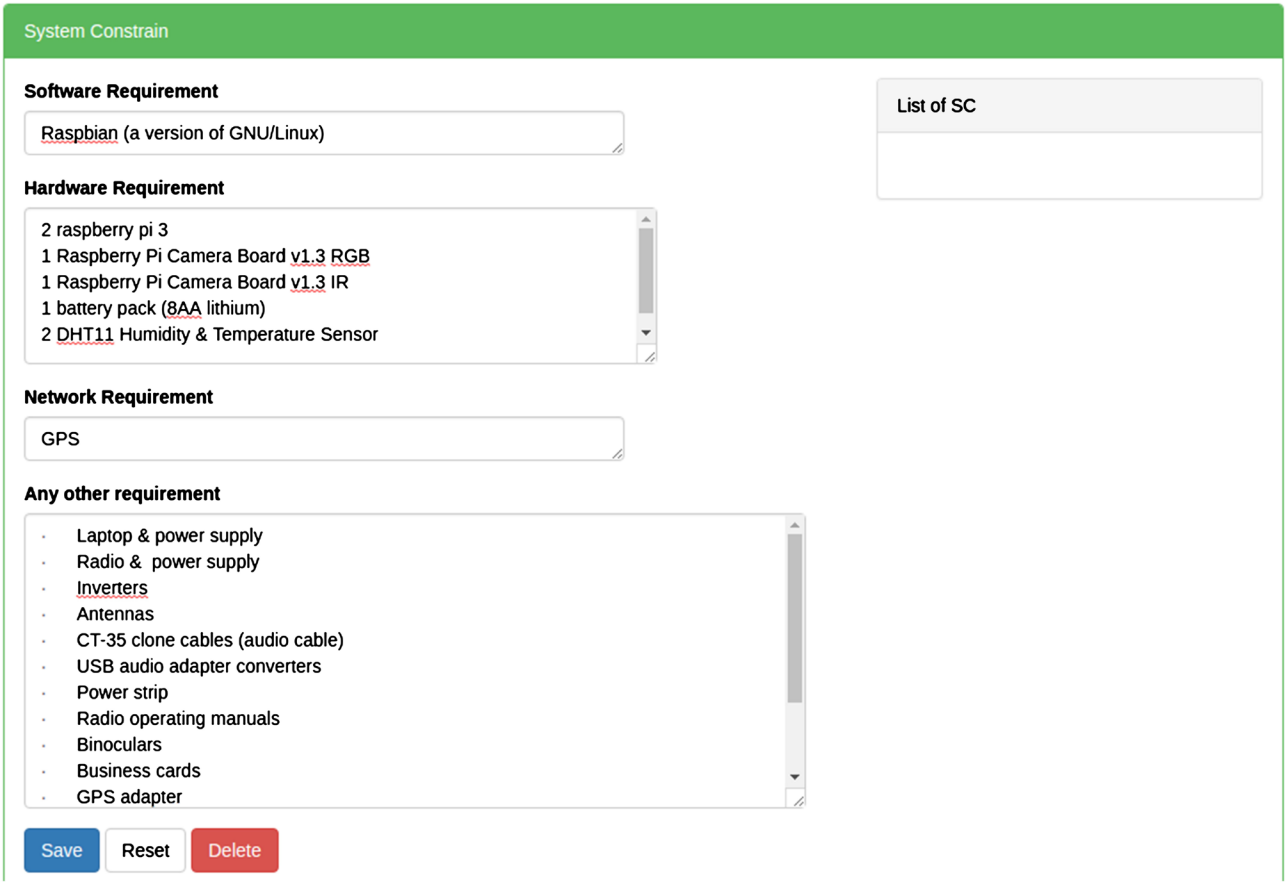

Figure 7. Template for creating and updating system constraints.

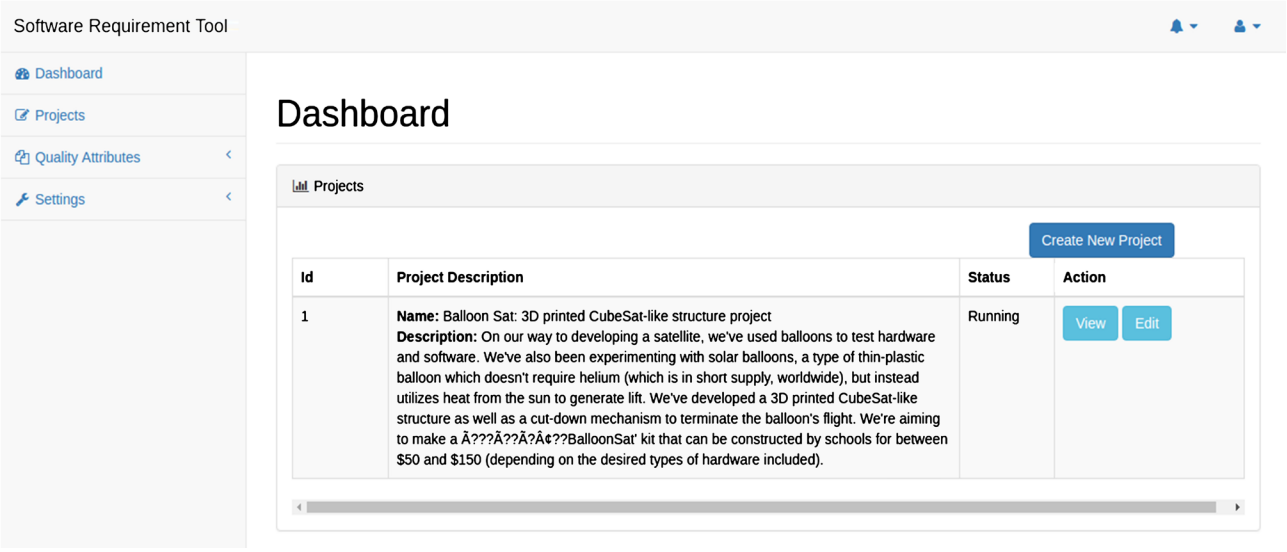

Figure 8. Dashboard for software requirement tool.

find the details of the project along with the list of functional requirements, non-functional requirements and system constraints associated with it. Also, the user can click on the individual requirements to find the details about them. When the user clicks the delete button, the project is deleted with the confirm button.

If the requirements inside the project are linked to some other projects, then the user must delete the relationship before completely deleting the project. The other feature is the edit, where the user can edit the project and the requirements inside them. When the user makes changes to any of the requirements and saves it, the tool notifies the user about the other requirements that they might want to change before proceeding as the other requirements are connected to each other. This is an important task as it captures the traceability between the requirements as a change in a requirement may affect others due to the dependability between them. To trace the dependability between the requirements when they are changed, we use the requirement traceability matrix. It helps us to keep track of the progress as well as ensures that each requirement is tested thoroughly and helps in determining the changes to be made in the requirements along the way.

Figure 9 shows a requirement traceability matrix of a project at an instance of change, which generated by scanning the database and finding the links between the requirements inside that is being changed. The requirements labeled FR (1⋯n) represent the functional requirements for the project, the requirements labeled NFR (1⋯m) represent the non-functional requirements for the project, and the symbol "x" represents the links between the requirements inside the project. This requirement traceability matrix is generated due to the change in one of the requirements inside a project. Once this is generated within the system, the tool then notifies the user about the relationship between the requirements. The user then has an option to either ignore the changes or make additional changes to the linked requirements. This process continues until the changes are made to the linked requirements or till, they are ignored. Once the dependencies are resolved, then the changes to the project are committed.

Figure 10 shows the alert dialogue box shown to the user when they decide to change any existing requirements in an existing project. The alert window shows the requirements (*i.e.* functional and non-functional) that are currently linked to the requirement being changed. The tool allows the users to either ignore

	FR #1	FR #2	⋯	FR #N	NFR #1	⋯	NFR #M
Link #1		x			x		
Link #2	x				x		x
Link #3			x		x	x	
Link #4			x				x
Link #5			x	x		x	

Figure 9. An instance of a requirement traceability matrix (RTM).

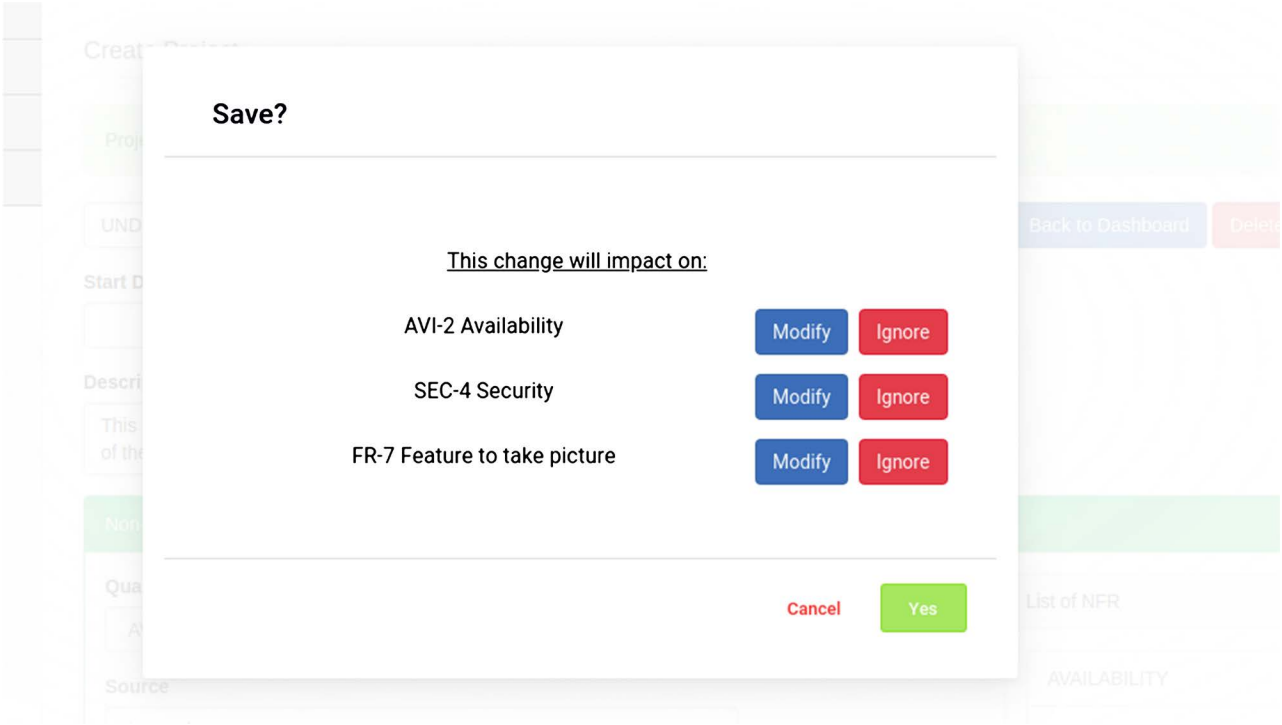

Figure 10. Notification on changing or updating a requirement.

or modify the linked requirements. If the user decides to ignore the requirements, that implies that the current change does not impact on that specific requirement and removes the link from the traceability matrix shown in **Figure 9**.

If the user decides to modify the linked requirement, the user is directed to a window based on the requirement (shown in **Figure 5** and **Figure 6**) and can make changes on them. Once the requirement is changed and the user clicks save button, the user is redirected to the alert window (*i.e.* **Figure 10**) where the ignored and modified requirements are removed. The user is only allowed to save the changes made on a requirement after they either ignore or modify all the linked requirements. By doing so, we achieve the traceability of the requirements in the developed tool.

The requirement engineering tool developed, helps the different stakeholders and the development team to team up on each progression of the project, and helps to not only capture the requirements but also to keep track of all the changes that are made or will be made into the system. Looking after relationship between quality properties and requirements of a project is dreary occupation for the author. This tool not only allows the user to capture the requirements of the system but also allows establishing a link between the functional and non-functional requirements allowing the members involved a detailed picture of the requirements.

5. Discussion

Requirement Engineering is one of the critical and challenging fields of study as

it keeps on changing as per the stakeholders and the development team throughout the system development process. Failure of a system that has been deployed and not meets its requirement needs is a hassle for both the stakeholders and the developers involved. As discussed in the previous section, the developed tool captures the requirement for a system and based on it, the user can make modification to it. The tool allows the user to capture both functional and non-functional requirement of a system before starting the development phase. To develop this tool, we have used JAVA, PHP, JavaScript, and HTML as the front end and MySQL as the relational database and can be used in any operating system environment.

The appropriate users for the tool would be knowledgeable of knowing the complete process of requirement engineering. The tool not only focus on the software developed in the market but also the safety critical systems, complex spacecrafts, and autonomous systems which on failure may even hamper people lives. The traceability of the tool helps the user to know the effects that might cause due to an addition or modification of a requirement in the system, which in turn notifies the user beforehand or during the development phase.

6. Conclusion and Future Work

In this paper, we have presented the process of designing and implementation of a requirement engineering tool which has the capabilities to capture the requirements, link the requirements to one another and let the user make changes to the requirements based on their demand. We have used the database approach to store the requirements and based on that, applied the concept of traceability matrix for capturing the traces between the requirements. The developed tool tries to bridge the problem of continuously changing software requirements. The major applicability of this tool is in the fields of software engineering, software architects and for the unmanned aircraft systems, where a minor change in a system can reflect a major impact on the overall project and lives of the people.

There are different approaches currently present in this field whereas this tool provides the user with the feature to maintain a one-to-one relationship between the requirements. In future, to improve this tool we can use the concepts of artificial intelligence such as machine learning and deep learning for making it easier for the users to handle and track the traceability of the requirements with detection of failure mechanisms.

Acknowledgements

We would like to thank Sanjaya Pandey of the University of North Dakota and Graciela Vargas Roque of the University of California at Davis for helpful suggestions and comments during the development of the tool. Furthermore, we would also like to acknowledge all the members at the University of North Dakota who were and are engaged in the area related to the work presented in this paper.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Nuseibeh, B. and Easterbrook, S. (2000) Requirements Engineering: A Roadmap. *ICSE*'00 *Proceedings of the Conference on the Future of Software Engineering*, Limerick, Ireland, 4-11 June 2000, 35-46. https://doi.org/10.1145/336512.336523

[2] Mader, P., Jones, P.L., Zhang, Y. and Cleland-Huang, J. (2013) Strategic Traceability for Safety-Critical Projects. *IEEE Software*, **30**, 58-66. https://doi.org/10.1109/MS.2013.60

[3] Pandey, S., Pokharel, S. and Hassan, R. (2018) Towards Cyber-Physical Requirement Engineering Elicitation Tool Support. 2018 *World Automation Congress* (*WAC*), Stevenson, WA, 3-6 June 2018, 1-5. https://doi.org/10.23919/WAC.2018.8430399

[4] Ward, J. and Daniel, E. (2006) Benefits Management: Delivering Value from IS & IT Investments. John Wiley & Sons, Chichester.

[5] Barbacci, M.R., Ellison, R., Lattanze, A.J., Stafford, J.A., Weinstock, C.B. and Wood, W.G. (2002) Quality Attribute Workshops. Technical Report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. https://doi.org/10.21236/ADA405790

[6] Cheng, B.H.C. and Atlee, J.M. (2007) Research Directions in Requirements Engineering. 2007 *Future of Software Engineering*, Minneapolis, MN, 23-25 May 2007, 285-303. https://doi.org/10.1109/FOSE.2007.17

[7] Kotonya, G. and Sommerville, I. (1998) Requirements Engineering: Processes and Techniques. Wiley Publishing, Hoboken, NJ.

[8] Van Lamsweerde, A. (2009) Requirements Engineering: From System Goals to UMl Models to Software. John Wiley & Sons, Chichester.

[9] Barbacci, M., Klein, M.H., Longstaff, T.A. and Weinstock, C.B. (1995) Quality Attributes. Technical Report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. https://doi.org/10.21236/ADA307888

[10] Chung, L., Nixon, B.A., Yu, E. and Mylopoulos, J. (2012) Nonfunctional Requirements in Software Engineering. Springer Science & Business Media, Berlin.

[11] Chung, L. and do Prado Leite, J.C.S. (2009) On Non-Functional Requirements in Software Engineering. In: Borgida, A.T., Chaudhri, V., Giorgini, P. and Yu, E., Eds., *Conceptual Modeling*: *Foundations and applications*, Springer, Berlin, 363-379. https://doi.org/10.1007/978-3-642-02463-4_19

[12] Tang, A. and Van Vliet, H. (2009) Modeling Constraints Improves Software Architecture Design Reasoning. 2009 *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, Cambridge, UK, 14-17 September 2009, 253-256. https://doi.org/10.1109/WICSA.2009.5290813

[13] Bass, L., Clements, P. and Kazman, R. (2003) Software Architecture in Practice. Addison-Wesley Professional, Boston, MA.

[14] O'Brien, L., Merson, P. and Bass, L. (2007) Quality Attributes for Service-Oriented Architectures. *Proceedings of the International Workshop on Systems Development in SOA Environments* (*SDSOA*'07: *ICSE Workshops* 2007), Minneapolis, MN, 20-26 May 2007, 3. https://doi.org/10.1109/SDSOA.2007.10

[15] Reza, H., Korvald, C., Straub, J., Hubber, J., Alexander, N. and Chawla, A. (2016) Toward Requirements Engineering of Cyber-Physical Systems: Modeling CubeSat. 2016 *IEEE Aerospace Conference*, Big Sky, MT, 5-12 March 2016, 1-13. https://doi.org/10.1109/AERO.2016.7500897

[16] Reza, H., Sehgal, R., Straub, J. and Alexander, N. (2017) Toward Model-Based Requirement Engineering Tool Support. 2017 *IEEE Aerospace Conference*, Big Sky, MT, 4-11 March 2017, 1-10. https://doi.org/10.1109/AERO.2017.7943647

[17] Egyed, A. and Grunbacher, P. (2005) Supporting Software Understanding with Automated Requirements Traceability. *International Journal of Software Engineering and Knowledge Engineering*, **15**, 783-810. https://doi.org/10.1142/S0218194005002464

[18] Nair, S., De La Vara, J.L. and Sen, S. (2013) A Review of Traceability Research at the Requirements Engineering Conference[re@21]. 2013 21*st IEEE International Requirements Engineering Conference* (*RE*), Rio de Janeiro, 15-19 July 2013, 222-229. https://doi.org/10.1109/RE.2013.6636722

[19] Gotel, O., Cleland-Huang, J., Hayes, J.H., Zisman, A., Egyed, A., Grunbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J. and Mader, P. (2012) Traceability Fundamentals. In: Cleland-Huang, J., Gotel, O. and Zisman, A., Eds., *Software and Systems Traceability*, Springer, Berlin, 3-22. https://doi.org/10.1007/978-1-4471-2239-5_1

[20] Gotel, O.C.Z and Finkelstein, C.W. (1994) An Analysis of the Requirements Traceability Problem. *Proceedings of IEEE International Conference on Requirements Engineering*, Colorado Springs, CO, 18-22 April 1994, 94-101.

[21] (2018) Your Requirements Engineering Tool. Requirements Management Software. Online. http://www.visuresolutions.com/requirementsengineering.

[22] (2018) Modern Requirements. Requirements Definition and Management. Online. http://www.modernrequirements.com/integreat/.

[23] de Gea, J.M.C., Nicolás, J., Fernández Alemán, J.L., Toval, A., Ebert, C. and Vizcaíno, A. (2012) Requirements Engineering Tools: Capabilities, Survey and Assessment. *Information and Software Technology*, **54**, 1142-1157. https://doi.org/10.1016/j.infsof.2012.04.005

[24] Reza, H., Jurgens, D., White, J., Anderson, J. and Peterson, J. (2005) An Architectural Design Selection Tool Based on Design Tactics, Scenarios and Nonfunctional Requirements. 2005 *IEEE International Conference on Electro Information Technology*, Lincoln, NE, 22-25 May 2005, 6.

[25] dos Santos Romani, M.A., Lahoz, C.H.N. and Yano, E.T. (2010) Identifying Dependability Requirements for Space Software Systems. *Journal of Aerospace Technology and Management*, **2**, 287-300. https://doi.org/10.5028/jatm.2010.02037810

[26] Egyed, A. and Grunbacher, P. (2002) Automating Requirements Traceability: Beyond the Record & Replay Paradigm. *Proceedings of the* 17*th IEEE International Conference on Automated Software Engineering*, Edinburgh, UK, 23-27 September 2002, 163-171.

[27] Pohl, K. (1996) PRO-ART: Enabling Requirements Pre-Traceability. *Proceedings of the Second International Conference on Requirements Engineering*, Colorado Springs, CO, 15-18 April 1996, 76-84.

[28] David, C., Dekhtyar, A. and Hayes, J. (2010) Automated Requirements Traceability: The Study of Human Analysts. *2010 18th IEEE International Requirements Engineering Conference*, Sydney, 27 September-1 October 2010, 231-240. https://doi.org/10.1109/RE.2010.35

[29] Gregoriades, A. and Sutcliffe, A. (2005) Scenario-Based Assessment of Nonfunctional Requirements. *IEEE Transactions on Software Engineering*, **31**, 392-409. https://doi.org/10.1109/TSE.2005.59

[30] Hall, R.J. (2008) A Method and Tools for Large Scale Scenarios. *Automated Software Engineering*, **15**, 113-148. https://doi.org/10.1007/s10515-008-0026-8

[31] Bashir, M.F. and Qadir, M.A. (2006) Traceability Techniques: A Critical Study. *2006 IEEE International Multitopic Conference*, Islamabad, Pakistan, 23-24 December, 265-268.