Traceability framework for requirement artefacts

Traceability Framework for Requirement Artefacts

Foziah Gazzawe , Russell Lock, Christian Dawson

Department of Computer Science

Loughborough University, United Kingdom

F.Gazzawe@lboro.ac.uk, R.Lock@lboro.ac.uk , C.W.Dawson1@lboro.ac.uk

Abstract. In a bid to improve requirement traceability techniques, a framework is presented which aims to clarify the link between artefacts, stakeholders who deal with the software, SDLC models, and their stages. Identifying the links will improve traceability and in doing so support the software development lifecycle. This paper will discuss why a conceptual framework is a suitable choice for the clarification of the links found. It also discusses the design of this framework, including its features and process. The potential contribution of the framework and its usefulness are also explained. A description of why, to whom, and how this framework will be of benefit is provided. This study thus provides an important asset applicable to all sectors of software development.

Keywords: Requirement Traceability, Design Requirement Artefacts, Traceability Framework, Artefacts Link; Mapping the Requirement Artefacts.

.

I. Introduction

The main objective of this research is to build a framework for traceability to increase support for software developers. The creation of a traceability framework makes it easier to understand the relationship that exists between software design, implementation, and requirements. Requirements and architectural frameworks enable software developers to understand the links that exist between traceability and requirement artefacts [7]. It works by analyzing and reasoning information, and communicates between the different aspects through links [2].

The framework to be developed is aimed at supporting software development in smaller-medium sized organizations with poor traceability. Smaller organizations require more support due to their limited budgets, as purchasing a traceability tool would be of very high cost to them. Hence, this free and open-sourced tool will help minimize the risks, improve traceability, and deliver a high-quality project. The requirement artefacts software framework would be useful throughout all the SDLC stages. Clearer understanding of relationships aids with the understanding of:

- Requirement analysis
- Design
- Development and Implementation
- Testing
- Maintenance

The application of conceptual frameworks enables software engineers to understand the links that occur between artefacts and the iterations involved in any given design phase during software development processes. [9] asserts that a traceability framework for requirements identification is an important tool in identifying relationships that exist in software development processes and the components involved. It is a platform which identifies possible errors in traceability and requirement artefacts recognition and correcting them in time before the completion of the software development process [4]. Although this is advantageous, the tool lacks the ability of application, due to its limited scope and use boundaries. [11] showed that the dependencies between software development components and their subcomponents is usually outlined by traceability framework designs so that stakeholders are able to filter information that relationships are going to meet their desired needs in software development.

[12] is also of the view that for an effective traceability framework to be designed for the purpose of viewing traceability relationships, there are three main aspects that must be considered: The first is that there is need for automation of maintenance traceability relationships. Secondly, traceability relationships must be created based on their information needs. Third is that traceability must be designed in such a manner that users are able to view it within familiar and common tools. Moreover, the evolution of artefacts' relationships should be enabled in terms of evaluation via a comprehensive framework that is easy to understand [4].

According to [4], the use of open information retrieval and integration provides a good conceptual framework that enables the understanding of the relationships that exist between traceability and requirement artefacts during software development processes. A conceptual framework presents an information integration environment framework that is guided by an open information retrieval system [12]. Using an information retrieval approach, the links between artefacts and traceability are outlined, creating a model that exhibits relationships between multiple sources. The role of analyzing design frameworks for software development and traceability is to identify the missing links between artefacts and relevant stakeholders during the development and implementation processes, amongst other issues faced.

Thus, according to [20] traceability frameworks are designed to outline relationships between different software development components as well as overcoming the problem of heterogeneous artefacts that different tools provide [12]. The potential problem of inconsistency often found after manually checking items for consistency leads to the recommendation of mapping mechanisms to clearly relate different artefacts. This is where the traceability framework would be of benefit, as they will rule out inconsistency by linking the heterogeneous artefacts together.

It is important to recognize that an artefact can come in different types and forms such as design, technical, and business requirement artefacts. The perspective proposed would be to distinguish between technical implementations and business requirements, and recognize the sub-categories of user oriented components and technical components for business requirements, and design and implementation for technical implementations. The requirement artefacts are divided into two types based on the order of the SDLC phases commencing with Business Requirements.

The purpose of these is to adhere to the business aspects of the project, and they are further divided into User Oriented Components and Technical components based on shared characteristics to help distinguish between them. According to [23], the User Oriented Components can have use cases or stories, as an example, and the Technical Components can have system requirements. The capture of Business Requirements may take place during the Requirement Gathering (Specification Document) phase and depends on it. Then a second type of artefacts also exists, namely Technical Software Implementation, which serves to satisfy the design aspects of the software. It is further divided into the Design and Implementation components.

The next section will briefly review some of the main concepts and definitions as gathered from secondary research in order to understand the purpose of the framework. This will link into the discussion of the framework and its design, such as its architecture and features, and make it more comprehensible. The following section will then explain the methodology used in this research, and then the findings will be displayed. The paper ends with a discussion of the findings as well as a look into the evaluation carried out.

II. Literature Review

Traceability can be said to be a process used in a software development project by stakeholders to identify the relationship that exists between software artefacts. The association between these artefacts and the way they depend on each other is understood through the process of software development [15]. The ability to trace the evolution of an artefact or requirement, in both forward and backward directions after the description of its links to its life cycle, can be defined as software artefact traceability [22].

There is a massive, inseparable link between requirements and traceability as traceability is exhibited by requirements. Requirements are the specific needs of a project that can be used to address an existing issue of the problem facing a software development project. The requirements can hardly be identified by the developers without a set process to detect those needs, known as traceability. Traceability involves a series of traces of tools used to fulfil the properties that are desired [6]. The process of linking requirements to the issues affecting a project forms the relationship between requirements and traceability, the two are treated separately and then linked by commonly desired factors [14].

Requirement traceability is a process which is necessary in software development but with numerous setbacks and challenges in its implementation. In cases of unchecked scope creep in requirement traceability, projects tend to get an inevitable downstream in quality, cost and time management. The elicitation of software requirements helps the development team identify the challenges and problems that may block the quality and effective completion of software. There are challenges related to link identification, designs, manual versus technological traceability, tool implementation and identification and the process used [3]. Identification of challenges and problems facing traceability is essential in filling the gaps of poor traceability processes as well as ensuring that future processes are improved in quality, time and cost managements.

There are several traceability tools that already exist, such as Rational Dynamic Object-Oriented Requirement Systems (DOORS), it optimizes requirement communication, verification and collaboration in the supply chain and software management of an organization [2]. There's also Requirements Tracing On-target (RETRO), which is known to employ the Information Retrieval (IR) methods in the tracing of requirements and software maintenance. The CRADLE Traceability Tool is another example, it makes the documentation process simple, defines product features, and is able to handle data modules. However, the use of these existing tools is limited. One of the major limitations includes the ability to automatically define relationships between different RAs. Finding the links between the relationships manually is an iterative process which often requires a certain level of expertise, and not enough clarity is provided by the existing tools. Thus, current traceability tools do not provide a clear guideline across different stages within the SDLC. Another limitation is that most traceability tools do not take into account the risk assessment based on missing relationships between RAs, nor how it impacts the end-user product. Therefore, the developed tool within this study attempts to overcome these limitations, and pave the way for a practical approach that incorporates traceability within smaller-medium sized organizations.

III. RESEARCH METHODOLOGY

This research, which is guided by the philosophy of Interpretivism, employs the qualitative methods of case study and interviews as a strategy to obtain the information required and help develop a framework for modelling requirements traceability. The data used is cross-sectional, and prior to the primary research, a comparison was made of existing techniques to help in developing the new framework and evaluating it.

In this research, the inductive approach was followed, starting off with observation through the literature review and testing with the case study, and leading to finding the research gap. One of the objectives set include the development of a traceability tool to aid developers with evaluating requirement artefacts, their relationships and level of risk, in line with satisfying the research's aim. This research contains a mixture of quantitative and qualitative data. One of the main contributions of the

research is the experimental tool, which is a quantitative measure. However, this research also provides a literature review, case study, interview, and an evaluation of the experimental tool, which are all qualitative contributions.

IV. FRAMEWORK DESIGN AND IMPLEMENTATION

The categorization of artefacts, as mentioned, is very important to the basis of this framework. An example of how an artefact is linked is with Technical Software Implementation, which serves to satisfy the design aspects of the software. It is further divided into the Design and Implementation components. The Design part has different tasks and models under it that may be used. A mock-up, as an example, is a prototype that enables testing and therefore obtains feedback from users. After this is conducted, Implementation then comes into play, where code is developed. All of these are different artefacts in the system, which link to each other as explained.

A. Framework Type

There are different types of frameworks that can be developed; conceptual and theoretical. According to [13], a theoretical framework is where one theory is implemented in order to clarify an ambiguous matter. A conceptual framework, on the other hand, is more suited for this research because it uses concepts from different theories to find an explanation for a particular subject. For instance, when different concepts are gathered in this research about artefacts then are connected to each other to make relations.

There are several ways of which a conceptual framework can be built, such as content analysis methods and a grounded theory method. According to [17], the content analysis method depends on setting a theory in place, as a hypothesis, before carrying out the testing for its validation. It also uses quantitative analysis, which is not suitable for this research, as conclusions need to be made from data collected.

Opposingly, the Grounded theory process starts with defining different categories by collecting information and then finding the links between them [5]. It also follows a qualitative analysis method, which is best for defining relationships. As the framework in this research aims to define the relationships between different artefacts and concepts in a software system, the Grounded theory is the method most fit for this purpose. In addition, [10] stated how this theory is most commonly used due to its effectiveness.

The aim of this paper is to present a framework which can support software developers in managing traceability. The aforementioned relationships in the ontology were created based on the analysis of secondary research to identify their types, factors that influence them, and common properties. An instance of these relationships can be seen in Figure 1 below.

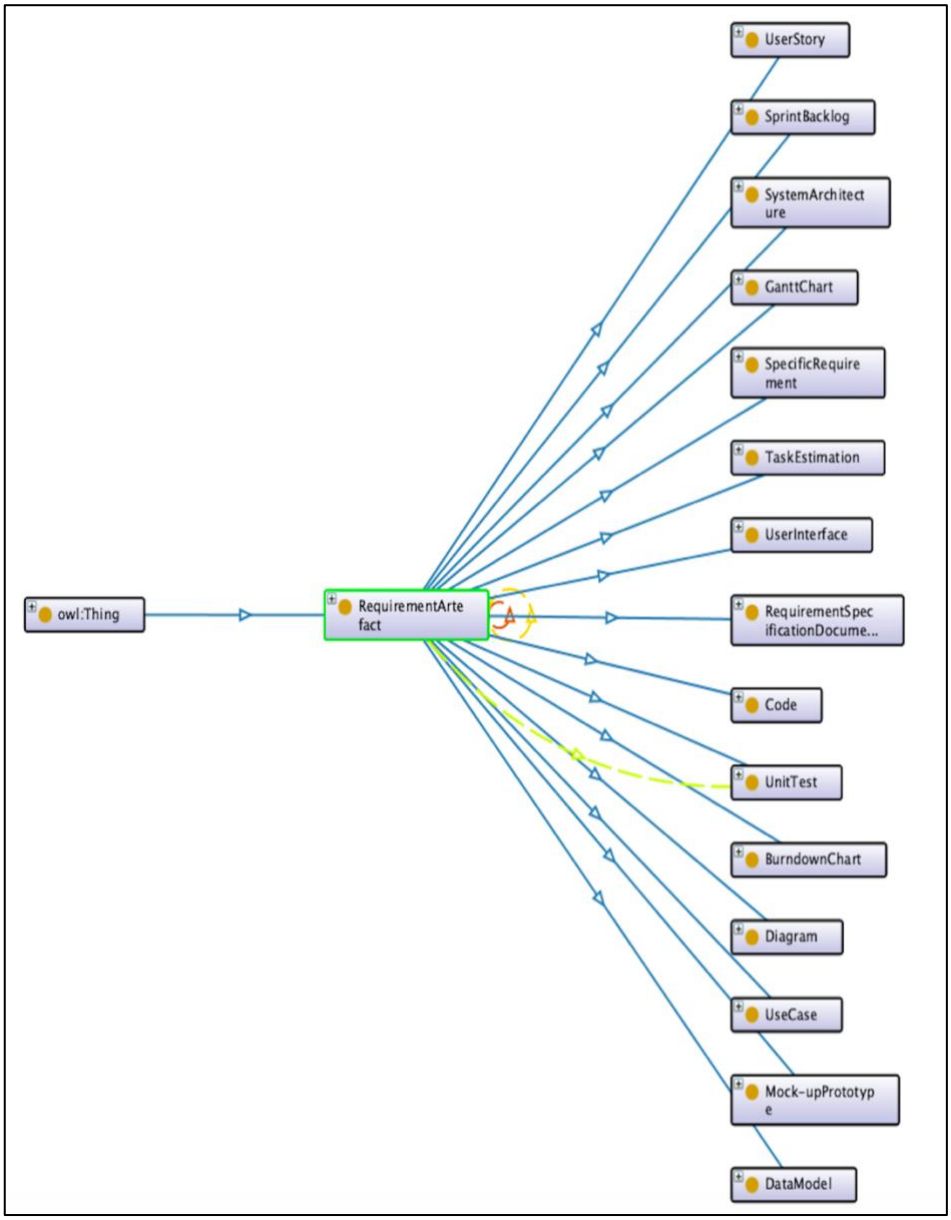

Figure 1: Example of Requirement Artefacts and their Subclasses

The figure above shows the main classes and subclasses in the ontology. The requirement artefact class models fifteen types of requirement artefacts. The requirement artefacts are modelled as subclasses in the ontology, for instance, User Story, Data Model, and User Interface.

B. Framework Features

A feature can be defined as a characteristic that is advantageous to the user, it is one of the most critical components of a framework, according to [21]. The main features of the missing links framework are as follows:

- Defining roles for people involved in each SDLC stage
- Categorisation of requirement artefacts
- Defining links between entities in the system
- Risk management through characteristics in the system
- Ability to customise the provided entities to allow for adaptation
- Providing guidelines for users to follow when tracing the artefacts relationships

The framework presented can be used to assign functions to people who play a key role in each stage of the SDLC to enable the adaptability of the framework, as well as it being able to set dependencies between the artefacts. This allows the categorization and linking of artefacts to be devised, providing the basis of the framework. Another essential feature in a conceptual framework is its ability to show the different types of links between entities and also to allocate requirement artefacts within each stage of SDLC. The aforementioned features include considering the different attributes and characteristics of each artefact to find the links between them, allowing the creation of more valid relations. Moreover, the framework has the ability to determine and manage risk and identify characteristics that can improve the system framework and also displays warnings when there are unmapped entities (requirement artefacts) to help prevent issues. Furthermore, an additional key feature of the framework is the ability to reuse the already provided entities already provided by customizing them according to the project. This allows for the framework to be more adaptive and useful in different contexts.

C. *Framework Development*

As the framework is developed, in order to aid with understanding, it is suggested that each artefact in each of the models in the ontology is broken down, decoupling them. The ontology was developed to store data and put it to use in the framework. This would help the developer to establish between the constituent parts of the waterfall model and the constituent agile artefacts model, which would make defining the missing links easier for the software developer, thereby linking them.

Some requirement artefacts aid with explaining the functions, building, and the design of the software, others are involved with the development and maintenance of the system itself. With the framework identifying the relations, it enables the developers to determine the importance of each requirement hence prioritising their implementation [3]. The guidelines the framework provides are derived from the

datasets of the ontology and assist the user in achieving the tasks appropriately and minimising the errors that might be encountered. Also, they have a role in ensuring the types of relationships between different entities are understood, and thereby the correct dependency properties are chosen. In addition, because the framework already provides the relationships between artefacts, the traceability process is quicker and less complicated. Not to mention, the artefacts determined at the start are turned into the deliverables of the system, achieving software development. As to the maintenance of the artefacts, the role of an artefact needs to be determined beforehand in order to be maintained, for example, practical artefacts need to be more heavily maintained [4]. The framework makes this easier by already identifying the roles of the artefacts.

V. FINDING AND RESULTS

As mentioned previously, the relationships in the ontology which were devised from secondary research are the basis of the framework. Hence, the process of the framework depends on retrieving information from the ontology. This was done based on the ontological analysis using Protégé software, as it stores all the relationships between the artefacts of the system, which are ready to be retrieved through the use of the framework. The Protégé program was chosen because it was seen as best fit to depict this framework as it has many advantages such as it being an open source, its extensibility to plug-ins, and its ability to describe ontologies explicitly, defining where individuals belong and what the class hierarchies are [18].

An example of how the framework process works is if a user wanted to find out the risks within a given configuration of requirement artefacts. They would select the entities they require, the system would then highlight the potential risks based on the relationships existing in the ontology. This would benefit the user by saving time and minimizing difficulties in case the risk was to take place with them being prepared. In order to delve into the actual process and understand deeply how it works, figure 2 provides a more detailed view.

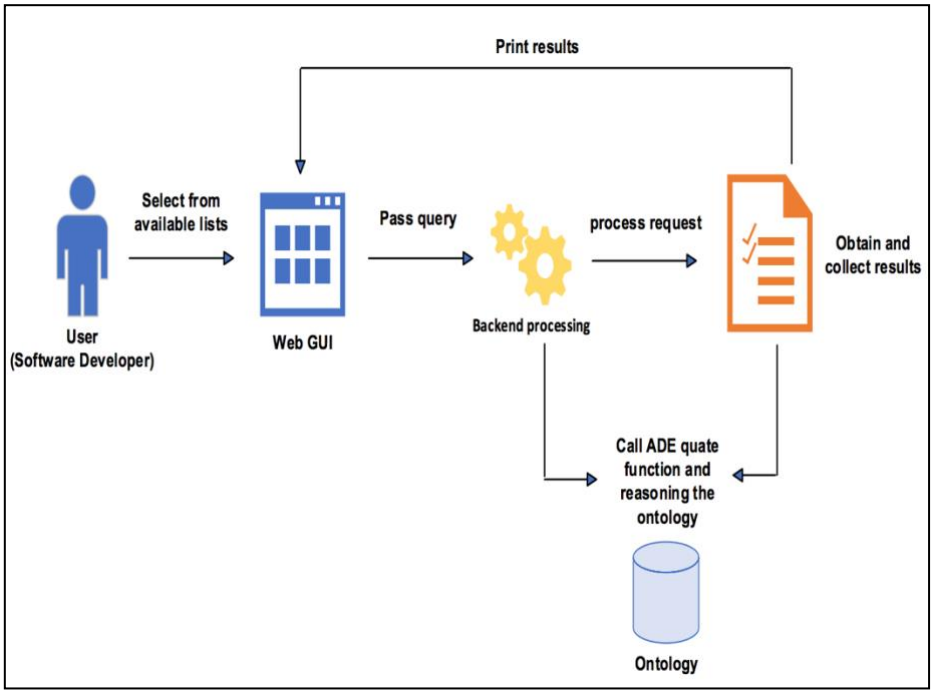

Figure 2: Framework Process

Figure 2 shows the stages of the process of the framework. The user would start by choosing from a drop-down list of expected questions in the Web GUI. The system then continues by reasoning the ontology in order to collect the results and send them back to the user.

A. Framework and Queries

To understand how the framework functions in more depth, the process is explained in this section:

1) To start, the developer chooses from a list of questions such as if the requirement artefacts are complete, if the relationships are correct, and if the current relationships contain any risk, etc. In addition to this, the user can choose from different requirement artefacts and stakeholder list then select a question that is related to stakeholder.
2) Next, for example, the relationship between a stakeholder and requirement artefacts or a stakeholder role, etc is chosen. The user then then presses the button of simulation and the system collects the results from the ontology.

3) If there is a relationship, the system shows the result of the user's choices and prints it as graph or text as appropriate.
4) If there is no relationship or link, then the system will give a suggestion to the user. For example, it shows a message saying there is no direct link between your choices please change them, or the system may suggest specific requirement artefacts that have a link. The system would then return to the earlier window (lists page) and the user selects again and submits the choices.
5) The results are received and printed.

VI. DISCUSSION

The framework developed in this research is intended to benefit both software developers and designers by identifying the links between artefacts, people in control of the software, and SDLC models, improving traceability. The framework would aid the developers in implementing traceability and thereby enhancing the quality of software developed. The framework aims to support the definition and classification of traceability and artefacts through an approach that enables semantic traceability to identify the missing link between them.

This framework also provides guidelines for the developers to follow during tracing the relationships between artefacts in each stage of the software lifecycle, verifying and validating them. These guidelines aim to help software designers by providing them with necessary information. They are derived from the datasets of the ontology and assist the user in achieving the tasks appropriately and minimizing the errors that might be encountered. Also, they have a role in ensuring the types of relationships between different entities are understood, and thereby the correct dependency properties are chosen.

In the ontology used, one of the most important features used is a reasoner. As mentioned by [11], one of a reasoner's contribution is testing to see if a class is a subclass of another. This feature allows to find any possible inconsistencies by ensuring that every class has instances relating to its conditions. If a class doesn't have instances, it is considered inconsistent. Another function of a reasoner is that it can compute the class hierarchy so it doesn't need to be done manually. According to [1] , a reasoner's use is vital because not only does it ensure there are no logical contradictions in the ontology, it can also use the information it has to infer more knowledge to add to its database. The restrictions placed on the classes or the properties it has is what allows the relations in an ontology to be inferred, as opposed to simply having a hierarchy of entities. The reasoner also aids with the overall building and maintenance of an ontology by performing these tasks.

In addition, because the framework already provides the relationships between artefacts, the traceability process is more efficient and less complicated than without the use of the framework. As to the maintenance of the artefacts, the role of an artefact needs to be determined beforehand in order to be maintained, for example,

practical artefacts need to be more heavily maintained [8]. The framework makes this easier by already identifying the roles of the artefacts.

VII. CONCLUSION AND FUTURE WORK

In this paper, a conceptual framework was proposed to aid software developers in solving traceability problems and easing the overall process of it. This was done through the use of an ontology that stores the relations between the artefacts of the system, i.e. the people, SDLC, and requirement artefacts. The proposed framework is thus fundamental in gathering information about software development requirements and in specifying the links that exist in each artefact in relation to the software and other artefacts.

As to the evaluation carried out in this research, it was done through testing the tool developed on software developers in a target company. Interviews were carried out and the outputs thematically analysed to highlight themes found. One of the important themes found outlines the difficulties encountered while using the tool, such as trouble with understanding the format and detailing of the output as well as with the sequence of the stages in the tool. However, with that in mind, the participants also identified a number of positives about the tool, which included capturing different stakeholders' roles and responsibilities, as well as the value of the provided guidelines, information, and the clear defining of the process.

References

[1] Abburu, S., 2012. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, *57*(17).

[2] Angelopoulos, K., Souza, V.E.S. and Pimentel, J., 2013, May. Requirements and architectural approaches to adaptive software systems: A comparative study. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on* (pp. 23-32). IEEE.

[3] Bashir, M.F. and Qadir, M.A., 2006, December. Traceability techniques: A critical study. In *Multitopic Conference, 2006. INMIC'06. IEEE* (pp. 265-268). IEEE.

[4] Bourque, P. and Fairley,RE., 2014. *Guide to the software engineering body of knowledge (SWEBOK), version 3.0*. IEEE Computer Society Press.

[5] Charmaz, K., 2006. Constructing grounded theory: A practical guide through qualitative analysis. Sage.

[6] Denney, E, and Fischer, B., 2005. Software Certification and Software Certificate Management Systems. Proceedings of ASE Workshop on Software Certificate Management SCM 05, pp. 1-5, Nov. 2005, Long Beach, CA. doi:10.1109/ase.2009.71.

[7] Elamin, R. and Osman, R., 2017, July. Towards Requirements Reuse by Implementing Traceability in Agile Development. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual* (Vol. 2, pp. 431-436). IEEE.

[8] Ghazi, P. and Glinz, M., 2017. Challenges of working with artifacts in requirements engineering and software engineering. *Requirements Engineering*, *22*(3), pp.359-385.

[9] Glenn A. Stout. 2001. Requirements traceability and the effect on the system development lifecycle (SDLC). In *Systems Development Process Research Paper* (pp. 3-17).

[10] Harris, I., 2003. What does "The discovery of grounded theory" have to say to medical education?. *Advances in health sciences education*, *8*(1), pp.49-61.

[11] Horridge, M., Simon. J, Georgina M., Alan R., Robert S., and Chris W., "A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 3." *The university of Manchester* 107 (2011).

[12] Helming J., Maximilian K., and Helmut N., 2009. Towards traceability from project management to system models. 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering. doi:10.1109/tefse.2009.5069576.

[13] Imenda, S., 2014. Is there a conceptual difference between theoretical and conceptual frameworks?. *Journal of Social Sciences*, *38*(2), pp.185-195.

[14] Jarke M., Rwth A., Aachen, and Germany., 1998. Requirements tracing. Communications of the ACM, vol. 41, no. 12, pp. 32-36. doi:10.1145/290133.290145.

[15] Kannenberg A. and Hossein S., 2009. Why software requirements traceability remains a challenge. CrossTalk the Journal of Defense Software Engineering, 22(5), pp.14-19.

[16] Kollin Z., 2017. *Dropdown alternatives for better (mobile) forms*. [online] Medium. Available at: https://medium.com/@kollinz/dropdown-alternatives-for-better-mobile-forms-53e40d641b53 [Accessed 25 Sep. 2018].

[17] Neuendorf, K.A., 2016. *The content analysis guidebook*. Sage.

[18] Noy, N.F. and McGuinness, D.L., 2001. Ontology development 101: A guide to creating your first ontology.

[19] Pohl Kl., 2010. *The Requirements Engineering Framework*. [online] Springer.com. Available at: https://www.springer.com/cda/content/document/cda_downloaddocument/9783642125775-c1.pdf?SGWID=0-0-45-972441-p173995858 [Accessed 8 Sep. 2018].

[20] Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U.A. and Kamran, K., 2012. Requirements traceability: a systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering*, *22*(03), pp.385-433.

[21] Trieloff L., 2014. *Feature Function Benefit vs. Feature Advantage Benefit.* [online] Medium. Available at:

https://medium.com/@trieloff/feature-function-benefit-vs-feature-advantage-benefit-4d7f29d5a70b [Accessed 12 Feb. 2018].

[22] Flynt, J.P. and Salem, O., 2004. Software Engineering for Game Developers Software Engineering Series. Course Technology PTR.

[23] O. Liskin, "How artifacts support and impede requirements communication," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2015, pp. 132–147.

[24] B. A. Sanchez, "Context-aware traceability across heterogeneous modelling environments," in *21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS-Companion 2018*, 2018.

[25] J. L. Brewer and K. C. Dittman, *Methods of IT project management.* 3 ,rd ed. Purdue University Press, 2018.

[26] P. Ghazi and M. Glinz, "Challenges of working with artifacts in requirements engineering and software engineering," *Requir. Eng.*, 2017.