TRACEABILITY AND VIEWPOINTS IN ENTERPRISE ARCHITECTURES

Dmytro Panfilenko, Roman Litvinov
DFKI IWi, Stuhlsatzenhausweg 3, Campus D3.2, D-66123 Saarbruecken, Germany

Dirk Werth, Peter Loos
DFKI IWi, Stuhlsatzenhausweg 3, Campus D3.2, D-66123 Saarbruecken, Germany

Keywords: Viewpoint, Traceability, System development, Enterprise architecture.

Abstract: In the times of increasing information volumes it is virtually impossible to harness the complexity and changes of the enterprise processes and requirements without taking into account the aid of the enterprise architectures, which are being supported by methodologies surveyed in this article. Software architectures for them serve as templates for system development processes and can describe the basic infrastructures for hardware, software and networks as well as their interrelations. Each involved participant fashions its own view on the final system, a developer or an architect alike, thus constituting rationale for the introduction of viewpoints at different abstraction levels provided in this article.

1 INTRODUCTION

Each viewpoint represents a model for certain stakeholders, as the capability for embracing the whole of the complexity is usually limited. Therefore it is rational to split a large concept into a series of relatively small views. Modelling is one of the acknowledged techniques for facilitating the understandability of the system under construction during development phase. The introduction of abstraction levels allows developers focusing on specific aspects of the system and communicating the information for specific notations and viewpoints of stakeholders.

The changes on the system caused by enterprise goal adjustment or obsolete technologies induce analysis of the dependencies between artefacts, where traceability plays important role during project phases of requirements definition, specification, and testing. Traceability of software artefacts has been identified as one of the important factors for supporting different activities during software system development processes. In general the goal of traceability is the improvement of the software systems quality, especially the analysis, integration and support for the induced changes on a system.

The term traceability has been introduced in the context of requirements traceability in the 1970-ies in order to avoid deviations between software behaviour and customer requirements. Software traceability is a crucial success factor in software engineering. During the evolution of projects a number of products, artefacts and relations between them emerge. One of the main tasks of traceability is to watch the changes in the relations between these artefacts and specification stated in the documentation. IEEE (1994) defined traceability as a capacity of a software for creation of a certain relation grade between artefacts, especially for the components having predecessor-successor or superordinate-subordinate relations.

This article is providing a survey on existing methodologies for viewpoints and traceability and is composed as follows: the related work section 2 makes a short overview of known scientific publications on viewpoints and traceability; traceability in concept of viewpoints in section 3 shows the classification approaches for viewpoints, traceability and its techniques; discussion and implications in section 4 present reasoning on the collected methodologies and finally section 5 makes a summary and outlines the future work.

Panfilenko D., Litvinov R., Werth D. and Loos P..
TRACEABILITY AND VIEWPOINTS IN ENTERPRISE ARCHITECTURES.
DOI: 10.5220/0003435301500156
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 150-156
ISBN: 978-989-8425-55-3

2 RELATED WORK

The term viewpoint is a topic of many scientific publications (Finkelstein et al., 1993; Kotonya and Sommerville, 1995; Leite et al., 1996; Sabetzadeh et al., 2010, and Steen et al., 2004). We return to detailed examining of this term in section 3.

Ramesh und Jarke, 2001 define this term as a characteristic of a system which uniquely binds the requirements with its sources and artefacts, i.e. requirement specification can include business requirements, user queries, rules, interface specification, source standards etc.

Projects involve a number of interest groups, e.g. sponsors, project managers, analysts, designers, programmers and end users. The task of traceability is to support the different interests, priorities and goals, which are difficult to provide at the same time (Ramesh and Edwards, 1993). Gotel and Finkestein, 1994 gave the following definition for traceability: The capability to describe and trace a requirement in both directions, forwards and backwards, from its specification to the development, usage and continuous improvement in each phase of the life-cycle. Edwards and Howell, 1991 wrote the definition as a guarantee for the support of the relations between requirement specification, design and the final implementation. As by Spanoudakis and Zisman, 2005 traceability is a capacity to trace the artefacts that emerge during system development in order to describe the system from different perspectives and levels of abstraction for all of the interest groups. Another interpretation of Wright, 1991 tells that with aid of requirement traceability a software developer can check whether the system fulfils the customer requirements and whether any unnecessary components to functionalities are present.

The benefit of software traceability usage lies in improved verification und validation of the customer requirements, lower maintenance costs and better software quality estimation. Moreover, traceability analyses system understanding, impact, defect correction and communication between developer and customer. Despite all the advantages it is not easy to conduct the complete checking through all the phases of the software development (Ahn, 2007). The confluence of different factors through the distribution between different groups, the heterogeneity of artefacts and used tools, the quick property changes of the components represent a great challenge for traceability management. The artefacts are distributed between different groups and as such difficult to reach. Heterogeneity hinders the traceability throughout different formats and abstraction levels. Lack of interoperability and tool support complicates the representation of the relation links. Due to constant changes of the relations between components they quickly become obsolete. These factors contribute to the high costs for traceability support.

Further reading of the following literature not discussed in this article in detail could be helpful for more comprehensive understanding of the topic (Andrade et al., 2004; Antoniol et al., 2002; Cleland-Huang et al., 2003; Darke and Shanks, 1996; Dijkman et al., 2008; Hayes et al., 2006; Leite and Freeman, 1991).

3 TRACEABILITY AND VIEWPOINTS IN ENTERPRISE ARCHITECTURES

3.1 Viewpoints in Enterprise Architecture Frameworks

Software systems comprise of a number of complex components. As per Breitman et al., 1999 there are managerial, organizational and computational aspects of a system that involve different resources (human, software, hardware, specification etc). Leite, 1996, defines an approach for viewpoint categorization which classifies viewpoints on three orthogonal directions for opinion (participants involvement), services (automatic goal fulfilment with aid of the "Viewpoint-Oriented Requirement Definition" (VORD) method by Kotonya and Sommerville, 1995) and specification (exact collaboration of different components described by Nuseibeh et al., 1994; Sommerville, 2007; Steen et al., 2004). Also, the standardisation work for viewpoints has already been laid out in IEEE 1471, 2007.

3.2 Viewpoints for Traceability

Sometimes the stakeholders cannot see the whole complex representation of the system and therefore are not able to improve the process of testing, documentation and validation in order to deliver the higher quality product to the customers. In general the traceability is the capability to follow the requirements all the way down from the specification and development of the system to the implementation and usage of the system. The goal of this article is to survey the traceability in context of viewpoints. The latter represent a concept for

requirements representation of the various stakeholders with their own perspectives and roles in the development process. Each stakeholder is responsible for the requirements profile and notation, with aid of which the domain, strategy and processes with histories are defined. It is supposed that each stakeholder can create her requirements in different viewpoints with different techniques. Frameworks with multiple viewpoints as per Finkelstein et al., 1993 provide the concept for tool integration through specific methods.

The authors believe that the focus of the consistency support lies in two following aspects. Firstly, all of the relations between different requirement profiles must be uniquely expressed. Secondly, through the transformation between specifications there are robust techniques for maintaining the consistency of the relations. Further, the techniques are discussed that provide for consistency and traceability.

3.2.1 Increasing Need for Traceability

A number of stakeholders like project manager, designer, and end-users with their own goals and priorities need the traceability for the consistent requirements specification and implementation. In the investigation phase it is important to represent the relations to specification in order to understand the development and verification better. During design phase the changes have to be documented. Traceability provides at this stage the capability to follow the modifications and information for reasoning and decisions. During the test phase the traceability comes in handy for the testing scenarios definition.

Change management benefits from traceability introduction as during changes the whole process of propagation of changing requirements can be examined and urgent changes can be identified. Moreover, the impact and costs of the changes can be estimated at each level.

3.2.2 Traceability Types

During design traceability empowers the designer to keep the overview over the changes' impact. It is useful when there is a possibility to relate the design and reasoning with respect to which decisions and assumptions can be bound to which results. System development needs better understanding of requirements during following them back to their sources. Due to facilitation of the references between instances of the requirements profile and design specifications precise change costs could be calculated.

Leon, 2000 points out a series of advantages the traceability brings with it: the overall analysis is easier to conduct, the better design is the result of traceability concept usage, the source code refactoring is bound to less costs. It is possible to discover all the potential problems in advance with traceability usage. Ecklund et al., 1996 claims the ease of the estimation of the impact through changes with aid of traceability. Gotel und Finkelstein, 1994 introduce two aspects of traceability: Pre- und Post-Traceability. Pre-Traceability deals with the aspects of the requirements, which are not directly involved in the requirement specification and focuses on better understanding of the requirements. The Pre-Traceability connects the original domain requirements with the actual ones for the better actual system and software understanding. The domain requirements comprise of stakeholders, business rules and original documentation. Post-Traceability defines such aspects of requirements in the design documents and components that are already in the requirement specification. In this way it is ensured that all the requirements through design and implementation are fulfilled.

Apart from these two aspects of traceability Davis, 1990 classifies traceability in the following four types: "forward-from traceability", "backward-to traceability", "forward-to traceability", "backward-from traceability". For further traceability classification approaches and illustrations please refer to Gotel, Finkelstein, 1994 and Emrich et al., 2010.

3.2.3 Traceability Techniques

In practice there are numerous techniques for traceability support being used as traceability matrix, hypertext, templates etc. They differ in variety of information that can be followed, in the number of relations between artefacts and in amount of traces being supported. Some forms of these techniques can be differentiated through usage of certain languages, models and methods for the development.

ARM. Modelling of software architecture requires as per Tang und Han, 2005 amongst other things the fulfilment of such premises as traceability, verifiability and completeness. For that the authors developed the "Architecture Rationale Method" (ARM). This method examines the qualitative and quantitative principles for the choice of the appropriate architecture design. "Architecture Rationale" is an artefact for storing the requirements and design principles. Qualitative rationale is described in textual format and chooses the design. Quantitative rationale concerns the costs, advantages

and risks of the chosen design. ARM is using the top-down method with two techniques: "requirement refinement" and "architecture decomposition".

Value Based Traceability Technique (VBRT). Other traceability techniques impose no importance on which requirements are essential and which are of lower value. The goal of VBRT is the identification of the "traces", which one can see as more prior and meaningful as the others (Heindl and Biffl, 2005). VBRT method comprises of 5 iterative steps: requirements definition for the requirements specification; requirements prioritization for estimation of the value, the risks and results of all stakeholders; requirements packaging for requirements clusters; requirements linking for the relations definition between artefacts and requirements here; evaluation for different goals, e.g. change impact estimation.

Feature-Oriented Requirement Tracing (FORT). Requirements tracing registers also the logical relation between individual requirements and further system elements. Certain artefacts as requirements specification, design documentation, source code and test cases are being created during software development. It is hard to determine which parts have been changed during software engineering process. The FORT method supports the links management through requirements prioritization with respect to costs and results (Ahn, 2007).

Pre-RS Tracing. The requirements are being checked against their sources that have primarily unstructured information. Traceability between requirements and their sources as per Ravichandar et al., 2007 is the great challenge for consistency support. This method is based on „capability engineering", which is the process for creating a change tolerant system under functional abstraction consideration known as capabilities, and comprises three phases: problem-, transition- und solution space.

Event-based Traceability (EBT). This method has been suggested by Cleland et al., 2002. The main reason for the development of this method is the thorough maintenance of the traceability relations. The authors define relation for traceability as "publisher - subscriber" type, which registers artefacts by the respective requirement. Each time the changes are being done a message to this event is being published which notifies all dependent artefacts. With aid of this method one can fulfil functional and non-functional requirements.

Information Retrieval (IR). This method can be used for automatic traceability links generation, which comprises Vector Space Model (VSM), Probabilistic Models and Latent Semantic Indexing (LSI) methods. IR is based on a similarity comparison and probabilistic values of two artefacts. Blaauboer et al., 2007 define three steps in this process: the pre-processing; the analysis, indexing, creation of the presentations and archiving; the analysis of the incoming artefacts with aid of classifying algorithm.

Rule-based Approach. This method has been suggested by Spanoudakis et al., 2004. The idea of this approach is the automatic traceability links generation with usage of a series of rules. All of the rules and all of the document types are presented in XML format. This method comprises the four essential steps: grammatical tagging, XML conversion, generation of requirements-to-object-model relations, and generation of requirements-to-requirements relations.

Hyper-Text based Approach (HB). The most traceability studies count it to optional activity that consumes a lot of resources and brings little benefit. The reasons for that depend not only on the developer side, but also on the project managers. The reason for that is the creation of multiple artefacts in different viewpoints during software system development. Artefacts represent as per Maletic et al., 2005 the system models on different abstraction levels (specifications, design, requirement models etc.).

Traceability Matrix (TM). This method is being used in the industry for definition of the relations between specification and other artefact types, to which also belong design, code module and test cases. This method envisions the manual relation creation between artefacts. In the practice the usage of traceability matrix is constrained to the critical non-functional requirements. Usually the requirements are being brought up on the rows and artefacts on the columns of the matrix, in which the marks being set on each of the cells when the according requirement relates to the artefact.

Goal-Centric Traceability (GCT). Traceability of NFR like security, performance and reliability is hard to reach. Cleland-Huang, 2005 developed a holistic method GCT for NFR. This method uses „Softgoal Interdependency Graph" (SIG) for NFR model description. It comprises four phases: goal modelling, impact detection, goal analysis and decision making (Cleland-Huang, 2005). The goals in SIG are being distributed in sub-goals for better requirements tracing with respect to stakeholder needs: goal modelling for data gathering, specification and design of the system; impact detection for definition of the traceability links and a set of potentially related SIG elements; goal analysis

for contribution re-analysis and goal re-evaluation; decision making for examination of the conducted analysis and identification of the influences of the changes with respect to NFR goals.

3.2.4 Reasons for Introduction

This section will elicitate the motives for introducing traceability for the system development.

Rationale and Origins. Originally traceability has been used for the development of the security features in the critical systems. Ramesh and Jarke, 2001 give an overview of the areas traceability can be used for. The relations between customer needs and requirements can be defined manually, whereas the automated techniques guarantee the complete tracing. The analysis and monitoring can identify the lost relations of the informal customer requirements, for example the creation of the traceability links of the customer requirements to the existing design.

Non-functional Requirements. Through traceability even the non-functional requirements can be bound to model elements and source code. In addition traceability provides for identification of contradictory requirements. It is always complicated because of scalability concerns to derive all the requirements, whereas the automatic methods for dependency creation can be critical, as it is not clear whether the requirements are consistent or not. Traceability in this context eases the communication between involved stakeholders.

Verification of Requirements. Software developer should check whether the original requirements are completely realized during the development process.

Determination of the Lost Requirements. It can happen in practice that one of the customer requirements by mistake or an oversight is not being accounted for or lost. If some requirements for scenario definition are missing, it can mean that one of the developers forgot to take these into account.

Determination of "change impact". In reality it is usual to see the changing requirements. These changes can lead to changes in other requirements that have to be taken into account.

Understanding of the Intensity of a Dependency. In this context the intensity is a number of joint classes or methods two artefacts have in between. It is clear that different traceability links depend on each other and one has to clarify to which percentage they are dependent.

Definition of Essential Artefacts. These should be handled with a special care, as they can constrain many other artefacts. Complexity of one artefact can depend on the number of constraining artefact, which traceability can determine.

Comparison of the Granularity of Requirements. In the early phases of the project the requirements gathering is relatively generic and is getting more complex in the later phases. Through traceability it is relatively easy to determine which requirements are generic and which are not.

Detection of Inconsistencies. Through creation of dependency links between requirements it is possible to identify the inconsistencies between different artefacts and to conduct the following activities as specification, detailed documentation of design and testing.

Responsibility. The existence of relations between artefacts during design, implementation and verification can support the understanding of the customer requirements. Moreover it is possible to conduct the regular checking for the test cases association with customer requirements.

Change Management. The thorough documentation of requirements and other artefacts support the change management. It is simple to follow the relations between design elements and the according position in the source code and thus to define the changes needed.

Verification and Re-engineering of Software. Systems. An advantage of traceability lies in the complete documentation of the relations between artefacts, which allows for checking the accordance of the requirements with the implemented system.

Storing the Information. Usually a participant in the project possesses only a constrained view at the big picture of the project development, but a very specific one, containing information endangered in case she lefts, but which could be preserved through traceability feature. Domges und Pohl, 1998 pointed this benefit of storing the information for the system.

4 DISCUSSION

Traceability term comes from the domain of requirements engineering and describes the relationships between requirement artefacts. In the literature there are different kinds of traceability: among other horizontal and vertical. Horizontal traceability analyses the relationships among requirements, whereas vertical traceability explores how requirements are used in consequent phases of the software development process. Other newer

methodologies can define traceability in a different way, in a sense that it analyses the traceability among all the artefacts.

In the context of the viewpoints, traceability is the main property among different artefacts that allows for analyses according to changes. The artefacts can be of different types, as in commonly known software engineering understanding of traceability. Moreover, it enables incorporation of additional information from the context, such as associated roles, persons, or help documents, etc. Thus, traceability is the key enabler for the system development for enterprise architectures.

Traceability provides the technique to analyse all changes that have to be performed, when a certain event occurs. However, it does not make any assumptions about the costs.

Costs models can either be explicitly assigned to certain transformation tasks with the traceability support and tooling or can be gathered by monitoring the actions of users during system development. Consequently, the costs in terms of assets, time and quality can be estimated for (a set of) transformation tasks.

In case of design decisions making this can be a helpful assistance. If two different transformation tasks would both resolve a conflict, cost estimation could help to decide, which task performing is more efficient.

System development with aid of traceability between the viewpoints allows for the seamless management of artefacts of a software architecture throughout the entire software and services development lifecycle. It empowers developers and software engineers to keep track of all the changes occurring in the context of a chosen architecture with respect to the system under development or the service cloud being modelled within this system. Furthermore, it helps understanding the system development process much better through the extensive traceability support for participants. It aids with identifying appropriate contact points for problems and enables easy-to-use cost estimations for transformation tasks. Overall, this makes decisions more transparent for the user.

5 CONCLUSIONS AND FUTURE WORK

In this work we provided an overview for existing viewpoint and traceability classifications and techniques. The task during development of a complex software system consists of the back-tracing of the changes being made in order to maintain their value for the enterprise. The decision concerning design is usually being made under unclear conditions, because the consequences of different alternatives cannot be determined exactly a-priori. To this extent the traceability helps making the right decision, minimizing the risks of inconsistencies and providing analysis for change impact despite the opinion that it only costs resources and time in vain.

In order to support the sustainability of the traceability subject to such factors as the project or enterprise specificity, we classified various methods for its realization. In practice there is a series of tools providing the realization frameworks for traceability concept, whereas manual and automatic methods are supported.

Future work will comprise research and development project work for viewpoint based enterprise architectures with traceability functionality including all kinds of forward, backward, vertical and horizontal traceability, impact analysis and recommendations.

REFERENCES

Ahn, S.; Chong, K.: Requirements Change Management on Feature-Oriented Requirements Tracing. In: *Lecture Notes in Computer Science*, 2007, pp. 296-307.

Andrade, J., Ares, J., Garcia, R., Pazos, J., Rodriguez, S., Silva, A., A methodological framework for viewpoint-oriented conceptual modeling, *IEEE Transactions on Software Engineering,* Volume 30, Issue 5, 2004, pp. 282-294.

Antoniol, G.; Canfora, G.; Casazza, G.; De Lucia, A.; Merlo, E.: Recovering Traceability Links between Code and Documentation, *IEEE Transactions on Software Engineering,* v.28 n.10, October 2002, pp.970-983.

Blaauboer, F.; Sikkel, K.; Aydin, M. N.: "Deciding to Adopt Requirements Traceability in Practice", *Springer Lecture Notes in Computer Science*, v. 4495/2007, pp 294-308.

Breitman, K. K.; Leite, J.C.S.D.P.; Finkelstein, A.: The world's a stage: a survey on requirements engineering using a real-life case study. In *Proceedings of J. Braz*. Comp. Soc.. 1999, 13-37.

Cleland-Huang, J.; Chang, C. K.; Sethi, G.; Javvaji, K.; Hu, H.; Xia, J.: Automating speculative queries through event-based requirements traceability. In: *Proceeding of the IEEE Joint RE*, 2002, p. 289.

Cleland-Huang, J.; Chang, C. K.; Christensen, M.: Event-Based Traceability for Managing Evolutionary Change, *IEEE Transactions on Software Engineering*, v.29 n.9, September 2003, pp.796-810.

Cleland-Huang, J.: Toward improved traceability of non-functional requirements. In: *Proc. of TEFSE '05, Automated Software Engineering*, 2005, p. 14-19.

Cleland-Huang, J., Settimi, R.; BenKhadra, O.; Berezhanskaya, E.; Christina, S.: Goal-centric traceability for managing non-functional requirements. In: *International Conference on Software Engineering*, 2005, pp. 362-371.

Darke, P.; Shanks, G.: Stakeholder, Viewpoints in *Requirements Definition: A Framework for Understanding Viewpoint Development Approaches, Requirements Eng.,* v. 1, pp. 88-105, 1996.

Davis, A.: Software requirements: analysis and specification. In: *Systems and Software Requirements Engineering, IEEE Computer Society Press* (1990).

Dijkman, R. M.; Quartel, Dick A.C.; van Sinderen, M. J.: *Consistency in multi-viewpoint design of enterprise information systems,* IST,v.50, n.7-8,2008,pp.737-752.

Domges, R.; Klaus, P.: Adapting traceability environments to project-specific needs. In*: Commun*. ACM (1998), v.41, n.12, pp. 54-62.

Ecklund, E. F.; Delcambre, L. M. L.; Freiling, M. J.: Change cases: use cases that identify future requirements. In: *Proceedings of the eleventh annual conference on Object-oriented programming systems, languages, and applications* (1996), v.31, n.10, pp. 342-358.

Edwards, M.; Howell, S.: *A methodology for systems requirements specification and traceability for large real-time complex systems* (1991).

Emrich, A.; Panfilenko, D.; Weber, S. (2010): MDA Organization Platform: A Holistic Approach for the Management of Model-Driven Architectures. In *Proc. of the 4th MDA4ServiceCloud'10 Workshop, co-located with the 6th ECMFA 2010*, Paris, France, June 15, 2010.

Finkelstein, A.; Easterbrook, S. and Kramer, J. and Nuseibeh, B.: Requirements Engineering Through Viewpoints. In: *DRA Colloquium on Analysis of Requirements for Software Intensive Systems*, 1993, pp. 18-26.

Gotel, O.; Finkelstein, A.: An Analysis of the requirements traceability problem. In: *Proceedings of the First International Conference on requirements engineering* (1994), pp. 94-98.

Hayes, J. H.; Dekhtyar, A.; Sundaram, S.K.: Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods, *IEEE Transactions on Software Engineering*, v.32 n.1, pp.4-19, 01/2006.

Heindl, M.; Biffl, S.: A Case Study on Value-based Requirements Tracing. In: *Proceedings of the 10th European software engineering (2005)*, pp. 60-69.

IEEE Std 1471-2000 ISO/IEC *Standard for Systems and Software Engineering – Recommended Practice for Architectural Description of Software-Intensive Systems*, 2007.

Kotonya, G.; Sommerville, I.: *Requirements Engineering with Viewpoints, Technical Report CSEG/10/1995*, CSEG Computing Department, University of Lancaster.

Leite, J. C. S. P.; Freeman, P. A.: Requirements Validation Through Viewpoint Resolution, *IEEE Transactions on Software Engineering*, vol. 17, no. 12, pp. 1253-1269, Dec. 1991.

Leite, J. C. S. D.: Viewpoints on Viewpoints, *Joint Proceedings of the SIGSOFT'96 Workshops*, The Association for Computing Machinery (ACM) 1996, pp. 285-288.

Leon, M.: Staying on Track. In: *Intelligent Enterprise* 2000, pp. 54-57.

Maletic, J. I.; Collard, M. L.; Simoes, B.: An XML Based Approach to Support the Evolution of Model-to-Model Traceability Links. In: *Automated Software Engineering (*2005), pp. 67-72.

Nuseibeh, B.; Kramer, J.; Finkelstein, A.: A framework for expressing the relationships between multiple views in requirements specifications, *IEEE Transactions on Software Engineering,* 20(10) 1994, pp. 760-773.

Ramesh, B.; Edwards M.: Issues in the development of a requirements traceability model. In: *Proceeding of the International Conference on Requirements Engineering* (1993), pp.256-259.

Ramesh, B.; Jarke, M.: Towards reference models for requirements traceability. In: *IEEE Transactions on Software Engineering*, Vol.27, No.1 2001, pp. 58-63.

Ravichandar, R.; Arthur, J. D.; Pérez-Quiñones, M.: Pre-Requirement Specification Traceability: Bridging the Complexity Gap through Capabilities. In: *TEFSE/GCT* (2007).

Sabetzadeh, M.; Finkelstein, A.; Goedicke, M.: "Viewpoints," in Encyclopedia of Software Engineering, P. Laplante, Ed. New York: Taylor and Francis, 2010.

Spanoudakis, G., Zisman, A.; Pérez-Miñana, E.; Krause, P..: Rule-based Generation of Requirements Traceability Relations. In: *The Journal of Systems and Software (07/2004*), v.72, Issue 2, pp. 105-127.

Spanoudakis, G.; Zisman, A.: Software Traceability: A Roadmap. In: *Handbook in Software Engineering and Knowledge Engineering*, Word Scientific Publishing Vol. 3 (2005), pp.395-428.

Steen, M. W. A.; Akehurst, D. H.; ter Doerst, H. W. L. l Lankhorst, M.M.: Supporting Viewpoint-Oriented Enterprise Architecture. *Proc. 8th IEEE International Enterprise Distributed Object Computing Conference* (EDOC'04), Monterey,California, September, pp. 201-211.

Tang, A.; Han, J.: Architecture Razionalization: A Methodology for Architecture Verifiability, Traceability and Completeness. In: *ECBS,* 2005, pp.135-144.

Wright, S.: Requirements Traceability –What? Why? And How? In: *IEE Colloqium, Computing and Control Division*, Digest Number 1991/180, pp. 1/1-1/2.