s c c h
software competence center
hagenberg

Towards Automated Traceability Assessment through Augmented Lifecycle Space

Miklós Biró, József Klespitz, Johannes Gmeiner, Christa Illibauer, Levente Kovács

Abstract. The assessment and improvement of the satisfaction of traceability requirements during the development of software in general and of safety-critical software in particular is demanding and costly. The special requirements are reflected in software process related general and industry specific standards and the popular agile approaches as well. It is imminent, for practical and logical reasons, that there is a need for the automation of the assessment of the completeness and consistency of traceability as far as possible. In addition to highlighting experienced weaknesses of current either homogeneous or heterogeneous tool environments intending to support lifecycle traceability, this paper outlines new solutions and suggests the exploitation of emerging technologies for the automation of traceability assessment and improvement.

Keywords: application lifecycle management, process assessment, process improvement, open services for lifecycle collaboration, tools integration, heterogeneous tools environment, existence, non-existence

This is the preprint version of the paper whose final published version appeared as:

- Biró, M., Klespitz, J., Gmeiner, J., Illibauer, C., & Kovács, L. (2016). Towards automated traceability assessment through augmented lifecycle space. In C. Kreiner, R. V. O'Connor, A. Poth, & R. Messnarz (Eds.) Systems, Software and Services Process Improvement, vol. 633 of Communications in Computer and Information Science, (pp. 94–105). Springer International Publishing. URL http://dx.doi.org/10.1007/978-3-319-44817-6_8

COMET
Competence Centers for Excellent Technologies
Funded by the Austrian government, the State of Upper Austria, the Johannes Kepler University Linz

1 Introduction

Our research is motivated by the needs of embedded software development for active medical devices which are naturally safety-critical. Safety-critical systems have so high risk of causing harm that this risk must always be reduced to a level “as low as reasonably practicable” (ALARP) required by ethics, regulatory regimes, and standards (IEC 61508).

For the special case of medical software development, the standard IEC 62304 Medical device software - Software life cycle processes, was released in 2006, and is under review to be harmonized with ISO/IEC 12207:2008 (Systems and software engineering -- Software life cycle processes).

MDevSPICE® [1] [2], released in 2014, facilitates the assessment and improvement of software development processes for medical devices based on ISO/IEC 15504-5, and enables the processes in the new release of IEC 62304 to be comparable with those of ISO 12207:2008. The above points give just a glimpse of the changes heavily affecting software developers in the medical devices domain.

Instead of containing actual recommendations of techniques, tools and methods for software development, IEC 62304 encourages the use of the more general IEC 61508-3:2010 Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems – Part 3: Software requirements, as a source for good software methods, techniques and tools.

Bidirectional traceability is a key notion of all process assessment and improvement models. [3] reports about an extensive literature review which classifies the models involving software traceability requirements according to the scope of the model, that is:

- Generic software development and traceability including CMMI and ISO/IEC 15504 evolving into the ISO/IEC 330xx (Information technology -- Process assessment) series of standards (SPICE).
- Safety-critical software development and traceability including DO-178C (Software Considerations in Airborne Systems and Equipment Certification) and Automotive SPICE.
- Domain specific software traceability requirements which, in the case of medical devices for example, include the already mentioned IEC 62304 (Medical Device Software – Software Life Cycle Processes), MDD 93/42/EEC (European Council. Council directive concerning medical devices), Amendment (2007/47/EC), US FDA Center for Devices and Radiological Health Guidances, ISO 14971:2007. (Medical Devices – Application of Risk Management to Medical Devices), IEC/TR 80002–1:2009 (Medical Device Software Part 1: Guidance on the Application of ISO 14971 to Medical Device Software), and ISO 13485:2003 (Medical Devices – Quality Management Systems – Requirements for Regulatory Purposes)

It is important to highlight that traceability is fully recognized as a key issue by the agile community as well [4] [5].
Unfortunately, complete and consistent traceability as well as the actual assessment of the satisfaction of the crucial traceability requirements is practically impossible to achieve with the heterogeneous variety of application lifecycle management (ALM) tools companies are using [6]. Following a manual approach, traceability assessors can only recur to sampling which has ultimate weaknesses detailed later in this paper.
It is evident that there are software development artifacts that can only be created by humans (customer, sales, marketing, etc.). Yet, there are other artifacts which can hardly be managed manually including for example the documentation of low level test results or results of automated testing (e.g. static and/or dynamic code analysis). Similarly, the number of relationships, including traceability links, between the different artifacts becomes prohibitive even in the simplest practical cases, so the handling and maintenance requires automated support.
Application Lifecycle Management systems (ALMs) are used to support the above mentioned processes. ALMs do not only cover the implementation, but the whole process starting from the initial idea, closing with the end of the product's life [7]. When a company chooses to set up an ALM, it can choose among numerous off-the-self or third party software systems and/or can decide to develop needed elements and optionally complement them with other management tools.
In this paper, after analyzing people and process challenges for achieving traceability in either homogeneous or heterogeneous tool environments, we point out fundamental logical and technical barriers for assessing and improving the completeness and consistency of traceability. Before introducing the suggested Augmented Lifecycle Space approach, we show relevant models which are good candidates to serve as its basis. Finally, RDF Triple Store (a purpose-built database optimized for the storage and retrieval of data entities composed of subject-predicate-object triples) is pointed out as the most suitable generic technology solution for representing artifacts and their relationships whose completeness and consistency must be verified by traceability assessors. RDF is at the core of the Linked Data paradigm the emerging cross-industry OSLC (Open Services for Lifecycle Collaboration) initiative is based on.

2 People and Process Challenges for Traceability in either Homogeneous or Heterogeneous Environments

It is a fact that 50-60% of software defects are related to requirements development [7]. Here, the rate of leakage (inherited defect which is detected only at a later stage) is 53% in the requirements phase and 68% in the design phase [3]. It is trivial that this fact raises the need for the improvement of current tools used to manage software development, especially requirements management.
Despite these facts, a significant proportion of people in charge of software development see traceability as a mandatory burden or as a useful but cumbersome

duty [8-10]. The need for traceability being undeniable, full compliance is difficult to enforce in everyday practice [11]. An example of the need is a developer exploring the code for possible effects of code modification. But a new employee also needs the traceability feature to get familiar with the code and the system it models. Finally, assessors have to rely on the traceability system to ascertain about the capabilities of the processes [12].

The aforementioned problems coincide with our experiences. Although, senior management is most of the time aware of the importance of traceability, developers are naturally prone to neglecting it. Paradoxically, developers are the ones who first suffer from the deficiency of traceability (e.g. code fragments to redesign for satisfying requirement changes are difficult to find) and their productivity is definitely increased in case of a well-designed traceability environment.

In [13, 14], authors identify and analyze eight challenges for traceability from a goal-oriented perspective only briefly alluded to below:

1. Traceability should be fit-for-purpose and has to support stakeholder needs.
2. The ROI (return on investment) from using traceability has to be adequate.
3. Traceability has to be maintainable and able to accommodate changing stakeholder needs.
4. The stakeholders have to have full trust in traceability.
5. Varying types of artifacts have to be traceable at variable levels of granularity and quantity.
6. Traceability has to be portable.
7. Traceability has to be a strategic priority valued by all.
8. Finally, traceability has to be ubiquitous, but it is achievable only when it is established and sustained with near zero effort.

The above issues are present in the case of even homogeneous ALM environments which can and mostly do provide extensive support for implementing traceability. The issues are however amplified in the case of widely occurring heterogeneous tool environments.

The introduction of a variety of tools can be caused by many factors. The company may choose not to depend on a certain vendor and to consider its unique needs which can only be matched with the offers of different vendors. Similarly, heterogeneity might result if the tool system was established incrementally over time. Moreover, there is qualm about the loss of accumulated intellectual property. ALM systems store tremendous amount of precious information which might be damaged or lost in case of migration. Therefore, companies, most of the time, insist keeping well-tried solutions to avoid such scenarios. Finally, changing habits and getting used to new tools can be cumbersome for anybody even for developers.

In a diversified environment, traceability, consistency and usability issues are naturally amplified. In cross-tool relationships, where direct connections do not exist, (e.g. requirements and their tests may be present in different tools) aging will erode traceability, and can lead to inconsistencies. Thus, it is of upmost importance to create direct connections between artifacts in order to maintaining traceability and enabling

consistency analysis. Furthermore, replication can be eliminated which further improves transparency and decreases the risk of introduced mistakes.

3 Logical and Technical Challenges for Assessing and Improving the Completeness and Consistency of Traceability

As already mentioned, ALM environments can and mostly do provide extensive support for creating traceability links and overviewing existing links.
There is however a fundamental difference between creating, overviewing links and proving that no links are missing which is the exact duty of the assessor and fully justifies the ultimate need for Automated Traceability Assessment.
The difference is a special case of the logical difference in mathematics between the proof of the existence of an object satisfying given properties and the proof of the non-existence of such an object. The existence (∃) can be proven by showing an instance, while the proof of non-existence is equivalent to showing that the property is not satisfied by any object (∀), which can obviously be much more difficult.
The figures below show a glimpse to the technical support and windows that developers are faced with when creating and assessing traceability links between artifacts in a few popular tools:

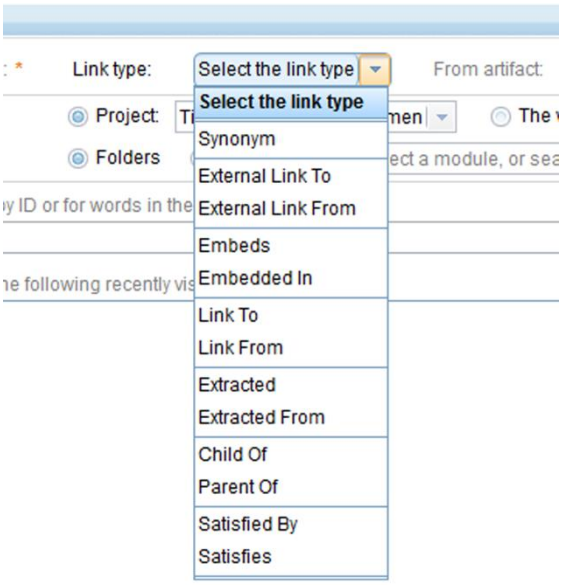

Fig. 1. Add a Link to an Artifact in IBM Rational DOORS

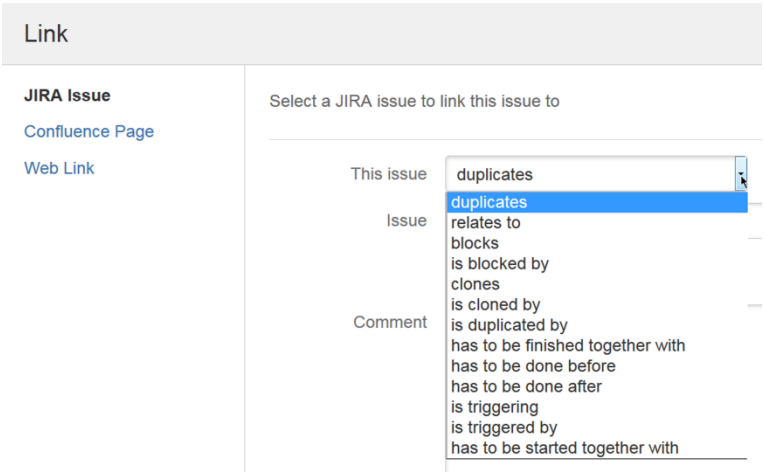

Fig. 2. Add a Link to an Issue in JIRA

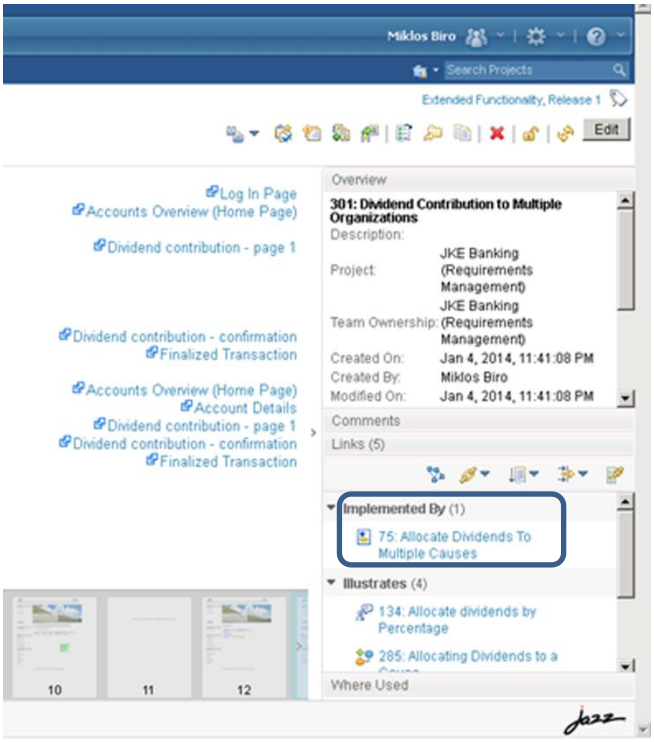

Fig. 3. Window fragment with Traceability Link in IBM Rational CLM

Regarding the task of traceability assessors, the currently only possible manual approach they have to be content with is sampling which has ultimate weaknesses in addition to the logical difficulty of proving non-existence described before:

- Traceability is basically restricted to the closed ALM system. Representational State Transfer (REST) APIs are mostly available for providing internal data. However, there is need for a standardized open form of exchange made possible by the emerging OSLC (Open Services for Lifecycle Collaboration) approach to be discussed.
- Useful traceability reports including the traceability matrix can be generated. However, the manual processing of these reports, due to their size, is only possible with very small examples whose complexity is exceeded by the simplest industrial applications. The reports only contain already existing artifacts while missing artifacts can only be discovered by manual inspection. The reports are static while requirements and identified defects, for example, are very dynamically changing artifacts, and may even originate from outside the ALM system.
- Assessors and users may be easily confused by the complexity of the set of widgets, such as buttons, text fields, tabs, and links which are provided to access and edit all properties of resources at any time.
- Assessors and users need to reach destinations such as web pages and views by clicking many links and tabs whose understanding is not essential for the assessment.

In summary, the logical and technical challenges themselves, together with the people and process challenges, fully justify the need for the automation of the assessment and improvement of the completeness and consistency of traceability [15].

4 Considered Models Addressing Traceability

In this section, preparing the grounds for the following proposed solution, we refer to the general level V-model appearing in the most recent version of the already mentioned Automotive SPICE standard [16] also highlighted in the paper [17], as well as the engineering models developed by the SoQrates Working Group “Traces” also presented in the paper [17].
Figure 4. provides the overview of the bidirectional traceability and newly highlighted consistency requirements of version 3.0 of Automotive SPICE published in 2015.

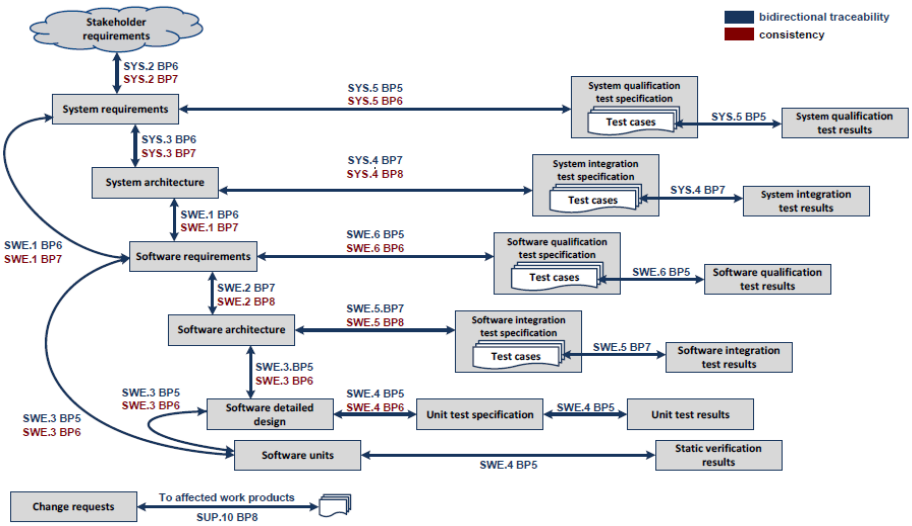

Fig. 4. Bidirectional traceability and consistency requirements of Automotive SPICE v3.0 [16]

The SoQrates Working Group "Traces" provides a systematically detailed engineering model including the traceability layer, addressing in addition reuse and variant management presented in the paper [17].

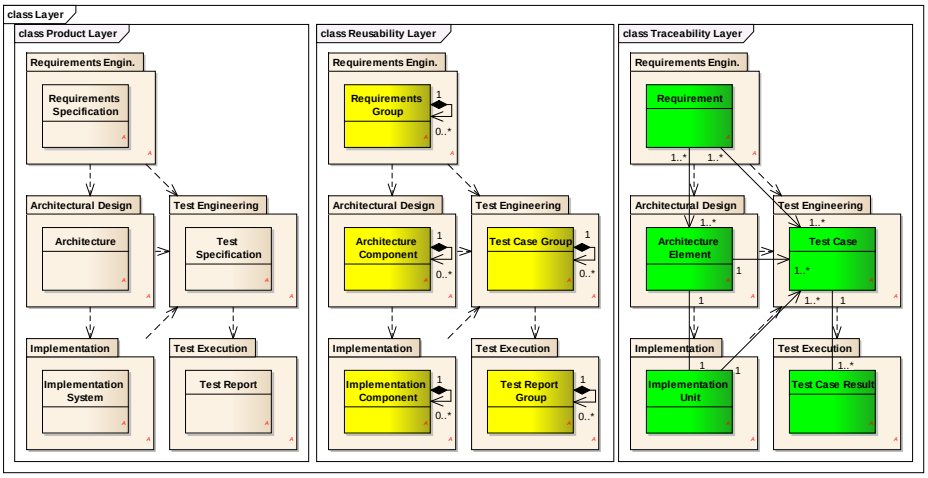

Fig. 5. Layers for Product, Reusability and Traceability in the SoQrates Working Group "Traces" model [17]

Both of the above models are good candidates to serve as base models for augmented lifecycle traceability described in the next section.

5 Traceability Assessment through Augmented Lifecycle Space

As discussed earlier, the assessment of the completeness and consistency of traceability relationships materialized by links requires the proof that no links representing required traceability relationships are missing and that there are no inconsistencies among the links.
In case the proof process identifies missing or inconsistent links, a workflow can be automatically created to address the bridging of traceability gaps and/or remediation of inconsistencies.
The fundamental question is: **How to find missing links and artifacts which, by definition, do not exist**?
Let us define **Lifecycle Space** to be the set of lifecycle artifacts with their relationships which exist anywhere in the lifecycle environment.
The **Augmented Lifecycle Space** approach, sketched along the following steps, is the solution suggested in this paper:

1. Categorize all existing artifacts of the homogenous or heterogeneous tool environment according to the elements of the chosen model containing traceability requirements (e.g. requirement, architecture element, test case, etc.).
2. Analyze the existing relationships (links) and the artifacts in the system and identify those which are missing but should exist according to the traceability requirements of the model.
3. If one of the two artifacts necessary for a required relationship is missing, automatically augment the system with the corresponding artifact whose links will be initially missing of course.
4. Analyze the relationships (links) of the augmented system. If a relationship (link) required by the model is missing, then automatically generate the task of the workflow for bridging the relationship gap.
5. Execute the relationship gap bridging workflow generated in the previous step involving manual intervention if necessary.

The Augmented Lifecycle Space approach allows the assessment and improvement of the completeness and consistency of traceability in either homogeneous or heterogeneous tool environments.
The question the following section intends to answer: **how to technically access artifacts and their relationships in a homogeneous or a heterogeneous tool environment.**

6 Technical Approaches to Access Artifacts and their Relationships – the OSLC Initiative

A homogeneous tool environment usually contains an Application Programming Interface (API) which allows access to the artifacts and their relationships within the tool A itself. This tool-A-dependent API could then be used from any other tool B to build a specific interface between the tools A and B allowing access to the artifacts and relationships stored in tool A from tool B.

In the case of the very common heterogeneous tool environments, whose roots were discussed earlier, the artifacts and their relationships have of course to be accessed across the usually numerous tools applied at the company. It is apparent in this case that the just described point-to-point tools integration is professionally unreasonable.

The already mentioned Open Services for Lifecycle Collaboration (OSLC) cross-industry initiative was recently created to overcome the professionally unreasonable approach of point-to-point tools integrations.

OSLC's aim is to define standards for compatibility of software lifecycle tools to make it easy and practical to integrate software used for development, deployment, and monitoring applications. This aim seems to be too obvious and overly ambitious at the same time. However, despite its relatively short history starting in 2008, OSLC is the only potential approach to achieve these aims at a universal level, and is already widely supported by industry. The unprecedented potential of the OSLC approach is based on its foundation on the architecture of the World Wide Web unquestionably proven to be powerful and scalable and on the generally accepted software engineering principle to always focus first on the simplest possible things that will work.

The elementary concepts and rules are defined in the OSLC Core Specification [18] which sets out the common features that every OSLC Service is expected to support using the terminology and generally accepted approaches of the World Wide Web Consortium (W3C). One of the key approaches is Linked Data being the primary technology leading to the Semantic Web which is defined by W3C as providing a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. And formulated at the most abstract level, this is the exact goal OSLC intends to achieve in the interest of full traceability and interoperability in the software lifecycle.

Full traceability of a requirement throughout the development chain and even the entire supply chain was also a major focus point of the European CESAR project (Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems) which adopted interoperability technologies proposed by the OSLC initiative [19].

Another important European project exploiting OSLC, was iFEST (industrial Framework for Embedded Systems Tools).

CRYSTAL (CRitical sYSTem engineering AcceLeration) is an ARTEMIS Innovation Pilot Project (AIPP) whose Interoperability Specification (IOS) is also based on OSLC [20].

In conclusion, the most suitable generic technology solution for representing artifacts and their relationships, that have to be accessed in any tool environment, is the RDF triple which is the object type the information in an OSLC resource is composed of and which is at the core of the Linked Data paradigm [21].
The technical manifestation in the OSLC paradigm of the Lifecycle Space, abstractly defined above, is the set of OSLC resources which all live in some OSLC ServiceProvider exposed by OSLC ServiceProviderCatalogs whose more detailed discussion is beyond the scope of this paper.

7 Conclusions and further works

The paper has shown that all approaches to achieving functional safety require the establishment of bilateral traceability between development artifacts and that the manual creation and maintenance of traceability links is possible, but the assessment of the completeness and consistency of traceability is not supported by current tools.
The Augmented Lifecycle Space approach, introduced in the paper, allows the automation of the assessment and facilitates the improvement of the completeness and consistency of traceability in either homogeneous or heterogeneous tool environments.
The technical solution suggested to handle artifacts and their relationships, in either homogeneous or heterogeneous tool environments, is OSLC (Open Services for Lifecycle Collaboration) which is based on the Linked Data paradigm.
Work is in progress targeting the implementation of the approach described in this paper in safety-critical software development environments.

8 Acknowledgement

The authors are grateful for the support of Research and Innovation Center of Óbuda University. The work is supported by the European Research Council Starting Grant ERC-StG 679681.
The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

9 References

1. Lepmets, M., Clarke, P., McCaffery, F., Finnegan, A., Dorling, A.: Development of a Process Assessment Model for Medical Device Software Development. In *industrial proceedings of the 21st European Conference on Systems, Software and Services Process Improvement* (EuroSPI 2014), 25-27 June, Luxembourg. (2014)

2. McCaffery, F., Clarke, P., Lepmets, M.: Bringing Medical Device Software Development Standards into a single model - MDevSPICE. In *Irish Medicines Board Medical Devices Newsletter* Vol. 1(40). (2014)
3. McCaffery, F., Casey, V., Sivakumar, M. S., Coleman, G., Donnelly, P., & Burton, J.: Medical device software traceability. In *Software and Systems Traceability*, pp. 321-339. Springer London. (2012)
4. Ambler, S.: Agile Requirements Best Practices. In *Agile Modeling* (2014) http://www.agilemodeling.com/essays/agileRequirementsBestPractices.htm (accessed on 08.04.2016)
5. Ambler, S.: Tracing Your Design. In *Dr. Dobb's Journal: The World of Software Development* (1999) http://www.drdobbs.com/tracing-your-design/184415675 (accessed on 08.04.2016)
6. Murphy, T.E., Duggan, J.: Magic Quadrant for Application Life Cycle Management. In *Gartner*, (2012)
7. Chapman, D.: What is application lifecycle management?, white paper (2010) http://www.davidchappell.com/writing/white_papers/What_is_ALM_v2.0--Chappell.pdf, (accessed on 08.04.2016.)
8. Capers, J.: Software Quality in 2011: A Survey of the State of the ART. In *Capers Jones & Associates LLC.* (2011)
9. Nistala, P., Kumari, P.: Establishing content traceability for software applications: An approach based on structuring and tracking of configuration elements. In *7th Workshop on Traceability in Emerging Forms of Software Engineering* (TEFSE), (2013)
10. Bouillon, E., Mäder, P., Philippow, I.: A survey on usage scenarios for requirements traceability in practice. In *19th International Working Conference on Requirements Engineering: Foundation for Software Quality* (REFSQ), Vol. 7830, LNCS, Springer, pp. 158-173. (2013)
11. Biró, M.: Open services for software process compliance engineering (invited paper). In V. Geffert, B. Preneel, B. Rovan, J. Štuller, A. Tjoa (editors), SOFSEM 2014: Theory and Practice of Computer Science, Lecture Notes in Computer Science, volume 8327, pages 1-6, Springer, January, (2014).
12. Panis, M.: Successful deployment of requirements traceability in a commercial engineering organization...really. In *18th IEEE International Requirements Engineering Conference* (RE), pp. 303-307. (2010)
13. Cleland-Huang, J, Gotel, O. C., Huffman Hayes, J., Mäder, P., Zisman, A.: Software traceability: trends and future directions. In *Proceedings of the on Future of Software Engineering*. ACM pp. 55-69. (2014)
14. O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic.: The grand challenge of traceability (v1.0). In *Software and Systems Traceability*, pp. 343-409. Springer, (2012)
15. G. Regan, M. Biro, F. Mc Caffery, K. McDaid, D. Flood. A traceability process assessment model for the medical device domain. In Systems, Software and Services Process Improvement, Springer Berlin Heidelberg, 2014, p. 206-216.
16. Automotive, S. I. G., VDA QMC Working Group 13.: Automotive SPICE Process Assessment / Reference Model, v3.0, (2015) http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf, (accessed on 08.04.2016)
17. Rainer Dreves, Frank Hällmeyer, Lutz Haunert, Bernhard Sechser. Method to Realize Traceability in Development Processes. Proceedings of EuroSPI22015.

18. OSLC Core Specification Workgroup.: OSLC core specification version 2.0. Open Services for Lifecycle Collaboration (2013) http://open-services.net/bin/view/Main/OslcCoreSpecification (accessed on 08.04.2016)
19. Jolliffe, G.: Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems (CESAR)–An Objective Overview. In *Making Systems Safer* pp. 37-50. Springer London. (2010)
20. Pflügl, H., El-Salloum, C., & Kundner, I.: Crystal, critical system engineering acceleration, a truly European dimension. In ARTEMIS Magazine, Vol. 14, pp. 12-15. (2013)
21. Biró, M.: *Functional Safety, Traceability, and Open Services.* In: L. Madeyski, M. Ochodek (ed.) Software Engineering from Research and Practice Perspective. Wyd. Nakom, Poznan, pp. 73-82. ISBN 978-83-63919-16-0 (2014)