Chapter 4

Functional safety, traceability, and Open Services

There are clear business reasons why the achievement of the functionality of more and more industrial products is definitely shifted to the use of embedded software replacing earlier hardware solutions. It is commonly known that the elicitation, identification, analysis, specification, modeling, validation, and management of the requirements in general and of functional safety requirements in particular is a demanding process with special implications for the case of software also reflected in related general and industry specific standards. This chapter gives a brief overview of standard approaches to functional safety with examples from the medical domain. It is highlighted that one of the key requirements of all approaches is traceability. The emerging Open Services for Lifecycle Collaboration (OSLC) technology, exploiting the architecture of the World Wide Web, is shown to have a determining impact on the future of the satisfaction of traceability requirements of safety-critical software development among others.

4.1 Functional Safety

It is unfortunate but also unquestionable that there is a risk that any product, with embedded software or not, causes harm to the health of people directly or indirectly as a result of damage to property or to the environment. The risk is usually defined as the product of the probability of occurrence and of the severity of the harm. There are of course products whose risk of causing harm is acceptable in a given context, based on the current values of society (IEC 61508). Safety-critical systems, on the other hand, have so high risk of causing harm that this risk must be reduced to a level “as low as reasonably practicable” (ALARP) required by ethics and regulatory regimes. Functional safety is the freedom from unacceptable risk regarding the functionality of the system.

The growing expectations regarding software components of safety-critical systems is a consequence of the changing impact of software on the consumer value of electrical, electronic or programmable electronic (E/E/PE) systems (IEC 61508) which was earlier fundamentally determined by hardware components with software primarily used for algorithmic tasks. Increasingly, embedded software is creating competitive differentiation for manufacturers [Ba11] in many industries including Automation, Aerospace, Automotive, Rail, Medical, Machinery, Nuclear, Process Automation and Consumer Products.

Software is perceived by business as more capable to be adapted to fluid requirements changes than hardware. In the software engineering discipline, it has for long been recognized however that the only way to achieve high reliability is to follow appropriately defined processes [BT95] [BT99] [BM00]. The necessary processes are summarized in international standards (ISO/IEC 12207 for software, ISO/IEC 15288 for systems), while their assessment and improvement is facilitated by the ISO/IEC 15504 series of standards (SPICE) currently evolving into the ISO/IEC 330xx series.

Regarding medical systems for example, the Association for the Advancement of Medical Instrumentation (AAMI) software committee reviewed ISO/IEC 12207:1995 and identified a number of shortcomings due to the fact that it was a generic standard. As a result a decision was taken to create a new standard which was domain specific to medical device software development, and in 2006, the new standard IEC 62304:2006 Medical device software - Software life cycle processes, was released. IEC 62304:2006 is approved today by the FDA (U.S. Food and Drug Administration) and is harmonized with the European MDD (Medical Device Directive). The quality management standard ISO 13485:2003, and the risk management standard ISO/IEC 14971:2007 are considered to be aligned with IEC 62304:2006 and their relationship is documented in IEC 62304:2006 itself. An extensive revision of the ISO/IEC 12207 standard took place in its release in 2008. As a result, all derived standards, including IEC 62304:2006, are under review.

To facilitate the assessment and improvement of software development processes for medical devices, the MediSPICE model based upon ISO/IEC 15504-5 was developed and has been renamed to MDevSPICE® in 2014 [LCMC14], [MCCL14]. The Process Reference Model (PRM) of MDevSPICE® will enable the processes in the new release of IEC 62304 to be comparable with those of ISO 12207:2008 [CMC13]. 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.

4.2 Traceability

Traceability and even bilateral (ISO/IEC 15504) or bidirectional (CMMI) traceability are key notions of all process assessment and improvement models. [MCCS12] 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.

- Safety-critical software development and traceability including DO-178B (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 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)

Let us first consider the definitions of traceability in the following broadly known models.

- CMMI® for Development, Version 1.3 November 2010
  - Traceability: A discernable association among two or more logical entities such as requirements, system elements, verifications, or tasks.
  - Bidirectional traceability: An association among two or more logical entities that is discernable in either direction (i.e., to and from an entity).
  - Requirements traceability: A discernable association between requirements and related requirements, implementations, and verifications.
- Automotive SPICE® Process Assessment Model, v2.5 2010-05-10 using the definition originally adopted in IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology
  - Traceability: The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.

Automotive SPICE® systematically requires bilateral traceability in all of the following engineering base practices:

- ENG.1 Requirements elicitation
- ENG.2 System requirements analysis
- ENG.3 System architectural design

- ENG.4 Software requirements analysis
- ENG.5 Software design
- ENG.6 Software construction
- ENG.7 Software integration test
- ENG.8 Software testing
- ENG.9 System integration test
- ENG.10 System testing

It is important to highlight that traceability has been considered as a key issue by the agile community as well. Scott Ambler, one of the key personalities of the agile movement, states in 1999 that "My experience shows that a mature approach to requirements traceability is often a key distinguisher between organizations that are successful at developing software and those that aren't. Choosing to succeed is often the most difficult choice you'll ever make—choosing to trace requirements on your next software project is part of choosing to succeed." [A99]

Scott Ambler's advice in 2014 [A14]:
"Think very carefully before investing in a requirements traceability matrix, or in full lifecycle traceability in general, where the traceability information is manually maintained."
He also describes rational arguments which support that maintaining traceability information makes sense in the following situations:

- Automated tooling support exists
- Complex domains
- Large teams or geographically distributed teams
- Regulatory compliance

The already cited IEC 61508 standard lists forward traceability as well as backward traceability as recommended and even highly recommended at the safety integrity levels 3 and 4 (SIL 3 and 4). The safety integrity of a system can be defined as "the probability (likelihood) of a safety-related system performing the required safety function under all the stated conditions within a stated period of time„. Safety integrity levels are defined by probability (required likelihood) of failure which is the inverse of safety integrity.

In IEC 61508, "highly recommended" means that if the technique or measure is not used then it is the rationale behind not using it which has to be carefully demonstrated during safety planning and assessment.

Unfortunately, traceability as well as the actual assessment of the satisfaction of the crucial traceability requirements is difficult to achieve with the heterogeneous variety of application lifecycle management (ALM) tools companies are faced with [PRZ09], [MD12]. Using a manual approach, assessors can only recur to sampling which has ultimate weaknesses:

- Traceability is basically restricted to the closed ALM system. Representational State Transfer (REST) APIs are mostly available for providing internal data. However, a standardized open form of exchange is only made possible by the below described OSLC approach.
- Useful traceability reports can be generated, but they are static while requirements and identified defects 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.

4.3 Open Services

Considering all of the above discussion, the need for the automation of assessing and maintaining traceability is imminent. It is this automation to which the Open Services for Lifecycle Collaboration (OSLC) [OSLC08] initiative opens the way.

OSLC is the recently formed cross-industry initiative aiming to define standards for compatibility of software lifecycle tools. Its aim is 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 OSLC aim is of course of utmost significance in the case of the safety-critical systems. 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 [OCS13] 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. OSLC is having a determining cross-fertilizing effect on the progress of the more general purpose Semantic Web itself.

The OSLC Core Specification is actually the core on which all lifecycle element (domain) specifications must be built upon. Examples of already defined OSLC Specifications include:

— Architecture Management
— Asset Management
— Automation
— Change Management
— Quality Management
— Requirements Management

Regarding for example the Requirements Management Specification whose version 2.0 was finalized in September 2012 [ORM12], it builds of course on the Core, briefly introduced above, to define the resource types, properties and operations to be supported by any OSLC Requirements Definition and Management (OSLC-RM) provider. Examples of possible OSLC Resources include requirements, change requests, defects, test cases and any application lifecycle management or product lifecycle management artifacts. Resource types are defined by the properties that are allowed and required in the resource. In addition to the Core resource types (e.g. Service, Query Capability, Dialog, etc...), the minimal set of special Requirements Management resource types simply consists of:

— Requirements
— Requirements Collections.

The properties defined in the OSLC Requirements Management Specification describe these resource types and the relationships between them and all other resources. The relationship properties describe for example that

— the requirement is elaborated by a use case
— the requirement is specified by a model element
— the requirement is affected by another resource, such as a defect or issue
— another resource, such as a change request tracks the requirement

- another resource, such as a change request implements the requirement
- another resource, such as a test case validates the requirement
- the requirement is decomposed into a collection of requirements
- the requirement is constrained by another requirement

Regarding the Change Management Specification, its version 3.0 is under development in 2014, and builds of course on the Core, briefly mentioned above, to define the resource types, properties and operations to be supported by any OSLC Change Management (OSLC CM) provider.

Examples of possible OSLC CM Resources include defect, enhancement, task, bug, activity, and any application lifecycle management or product lifecycle management artifacts. Resource types are defined by the properties that are allowed and required in the resource.

The properties defined in the OSLC Change Management Specification describe these resource types and the relationships between them and all other resources. The relationship properties describe in most general terms for example that

- the change request affects a plan item
- the change request is affected by a reported defect
- the change request tracks the associated Requirement
- the change request implements associated Requirement
- the change request affects a Requirement

OSLC is currently at the technology trigger stage along its hype cycle [FR08] [B09], it is already clear however that it is the approach which has the potential to have a determining impact on the future of the satisfaction of the traceability requirements of safety-critical software development among others.

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

Another important European project, completed in 2013 and exploiting OSLC, is iFEST (industrial Framework for Embedded Systems Tools).

4.4 Conclusion

The chapter has shown that all approaches to achieving functional safety require the establishment of bilateral traceability between development artifacts. Unfortunately, the manual or even tool supported creation and maintenance of traceability is currently difficult and expensive. The emerging industry supported OSLC initiative has been shown to have a determining impact on the future of the satisfaction of traceability requirements of safety-critical software development among others.

References

[A14] S. Ambler, Agile Requirements Best Practices, 2014. http://www.agilemodeling.com/essays/agileRequirementsBestPractices.htm (accessed: 15/08/2014).

[A99] S. Ambler, Tracing Your Design. Dr.Dobb's Journal: The World of Software Development, 1999.

[B09] M. Biro. The Software Process Improvement Hype Cycle. Invited contribution to the Monograph: Experiences and Advances in Software Quality (Guest editors: D.Dalcher, L. Fernndez-Sanz) CEPIS UPGRADE Vol. X (5) pp. 14-20 (2009) http://www.cepis.org/files/cepisupgrade/issue%20V-2009-fullissue.pdf (accessed: 15/08/2014)

[B14] M. Biro. Open services for software process compliance engineering. In V. Geffert, B. Preneel, B. Rovan, J. Štuller, & A. Tjoa (Eds.) SOFSEM 2014: Theory and Practice of Computer Science, vol. 8327 of Lecture Notes in Computer Science, (pp. 1–6). Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-04298-5_1 (accessed: 15/08/2014)

[Ba11] M. Bakal. Challenges and opportunities for the medical device industry. IBM Software, Life Sciences, Thought Leadership White Paper, (2011)

[BM00] M. Biró, R. Messnarz R. Key success factors for business based improvement. Sofware Quality Professional 2:(2) pp. 20-31, 2000. http://asq.org/pub/sqp/past/vol2_issue2/biro.html (accessed: 15/08/2014)

[BR98] M. Biró, T. Remzső. Business motivations for software process improvement. ERCIM NEWS 32: pp. 40-41, 1998. http://www.ercim.eu/publication/Ercim_News/enw32/biro.html (accessed: 15/08/2014)

[BT95] M. Biró, P. Turchányi. Software Process Assessment and Improvement from a Decision Making Perspective. ERCIM NEWS 23: pp. 11-12, 1995. http://www.ercim.eu/publication/Ercim_News/enw23/sq-sztaki.html (accessed: 15/08/2014)

[BT99] M. Biró, C. Tully. The software process in the context of business goals and performance. In: Messnarz R, Tully C (ed.) Better Software Practice for Business Benefit: Principles and Experience. 409 p. Washington; Paris; Tokyo: Wiley - IEEE Press, 1999. pp. 15-28. (ISBN: 978-0-7695-0049-2)

[CMC13] V. Casey, F. McCaffery. The development and current status of MediSPICE. In T. Woronowicz, T. Rout, R. OConnor, A. Dorling (Eds.) Software Process Improvement and Capability Determination, vol. 349 of Communications in Computer and Information Science, (pp. 4960). Springer Berlin Heidelberg, 2013.

[FR08] J. Fenn, M. Raskino. Mastering the Hype Cycle. Harvard Business Press, 2008.

[LCMC14] M. Lepmets, P. Clarke, F. McCaffery, A. Finnegan, A. Dorling. 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)

[MCCL14] F. McCaffery, P. Clarke, M. Lepmets. Bringing Medical Device Software Development Standards into a single model - MDevSPICE. To appear in: Irish Medicines Board Medical Devices Newsletter Vol 1(40). (2014)

[MCCS12] F. McCaffery, V. Casey, M.S. Sivakumar, G. Coleman, P. Donnelly, J. Burton. Medical device software traceability. In J. Cleland-Huang, O. Gotel, & A. Zisman (Eds.) Software and Systems Traceability, (pp. 321-339). Springer London, 2012. http://dx.doi.org/10.1007/978-1-4471-2239-5_15 (accessed: 15/08/2014)

[MD12] T.E. Murphy, J. Duggan. Magic Quadrant for Application Life Cycle Management. Gartner, 2012.

[OCS13] Open Services for Lifecycle Collaboration Core Specification Version 2.0. (2013) http://open-services.net/bin/view/Main/OslcCoreSpecification (accessed: 15/08/2014)

[ORM12] Open Services for Lifecycle Collaboration Requirements Management Specification Version 2.0, 2012. http://open-services.net/bin/view/Main/RmSpecificationV2 (accessed: 15/08/2014)

[OSLC08] Open Services for Lifecycle Collaboration,2008. http://open-services.net/ (accessed: 15/08/2014)

[PRZ09] G. Pirklbauer, R. Ramler, R. Zeilinger. An integration-oriented model for application lifecycle management. Proceedings of the 11th International Conference con Enterprise Information Systems (ICEIS 2009), pages 399-403, INSTICC. (2009)

[RBMC14] G. Regan, M. Biro, F. Mc Caffery, K. Mc Daid, D. Flood. A traceability process assessment model for the medical device domain. In B. Barafort, R. O'Connor, A. Poth, & R. Messnarz (Eds.) Systems, Software and Services Process Improvement, vol. 425 of Communications in Computer and Information Science, (pp. 206–216). Springer Berlin Heidelberg, 2014. http://dx.doi.org/10.1007/978-3-662-43896-1_18 (accessed: 15/08/2014)