Agile Requirements Traceability Using Domain-Specific Modelling Languages

Masoumeh Taromirad
Department of Computer Science
University of York
York, UK
mtaromi@cs.york.ac.uk

Richard F. Paige
Department of Computer Science
University of York
York, UK
richard.paige@york.ac.uk

ABSTRACT

Requirements traceability is an important mechanism for managing verification, validation and change impact analysis challenges in system engineering. Numerous model-based approaches have been proposed to support requirements traceability, but significant challenges remain, including finding the appropriate level of granularity for modelling traceability and coping with the lack of uniformity in requirements management tools. This paper argues for an *agile modelling* approach to managing requirements traceability and, in this context, proposes a domain/project-specific requirements traceability modelling approach. The preliminary approach is illustrated briefly in the context of the safety-critical systems engineering domain, where agile traceability from functional and safety requirements is necessary to underpin certification.

1. INTRODUCTION

Requirements traceability (or *traceability* for short) is an important mechanism for managing and auditing the entire software development process [14]; arguably this is because all artefacts of software development need to be driven by, and linked to, requirements. Traceability is utilised for various purposes, including requirements management, change management and impact analysis, verification and validation, certification (in the critical systems domain) and audit. Traceability in Model-Driven Engineering (MDE) is widely studied, particularly because the automated support available with MDE (e.g., model transformations) allows trace information (generally called *trace models*) to be automatically generated and managed. However, there are substantial challenges associated with traceability in MDE, which make it a challenge to select or define an effective traceability framework – consisting of traceability models, metamodels and analysis tools – for a specific project.

Since a fully encompassing fine-grained traceability scheme in MDE is (in practical terms) unmanageable, we propose an agile approach to requirements traceability in MDE. Through agility, we aim for a simple and light enough traceability scheme. In particular, we propose to use a *domain-specific requirements traceability approach* to provide *just enough* traceability, while addressing other requirements traceability challenges as well. As discussed in [6], domain characteristics are critical in finding the *right* amount of required information to support in an MDE traceability scheme. Domain characteristics can be elucidated and captured in domain-specific traceability models and metamodels. However, the information that we need to gather is likely to change over the course of a project. As such, these models and metamodels will need to evolve incrementally through iterations. This is both a challenge and a benefit: it will help to ensure that the traceability models and metamodels capture the most useful information at the current time, but will also require model/metamodel evolution mechanisms to be used.

This paper contributes a Domain-Specific Modelling Language (DSML) approach to building a traceability scheme for a specific domain or project *incrementally*. The idea is that we use a DSML to describe a domain-specific traceability metamodel and represent the structures, behaviours and features of the target domain with respect to project traceability goals. MDE tooling is thereafter used to both create traceability models and to help evolve models and metamodels as project-specific traceability requirements change. This approach allows engineers to focus on essential domain-specific traceability information to get the most value out of traceability, especially in comparison to general-purpose traceability frameworks and tools.

We are applying the proposed approach in the context of the safety-critical system engineering domain, in which managing requirements is a critical issue and traceability is both necessary and helpful in addressing challenges in this domain. These challenges include ensuring that evidence is available to demonstrate that safety requirements have been met [16], dealing with changing requirements and the corresponding effects on certification arguments [19], and supporting evidence-based safety assurance [9]. However, evolution of artefacts (including DSMLs and trace models) in domains like safety is problematic, because *safety is not compositional* – that is, if a system has been shown to be acceptably safe to operate (i.e., safety requirements are met), and an engineering artefact evolves, then the safety process has to be fully re-executed.

This paper presents our preliminary work on this topic, focusing on the DSML and how it is connected to other system engineering artefacts in the context of building safety-critical systems. We also identify challenges for manipulating trace models and metamodels in this context, particularly for supporting incremental development of the DSML.

The paper is structured as follows: in Section 2, we talk about the background and related work in requirements traceability area. Section 3 introduces the proposed approach: domain-specific requirements traceability. In section 4, an example of applying the proposed approach in safety-critical system domain is explained. Section 5 discusses the future work and Section 6 gives the conclusion.

2. BACKGROUND

In the requirements engineering field, traceability is defined as "the ability to describe and follow the life of a requirement in both a forwards and backwards direction" from inception throughout the entire system's lifecycle [8]. In requirements engineering, traces are valuable for numerous purposes, including validation and verification, where they help to identify pairs of relating artefacts which can then be validated and verified against each other. A recent thorough study on traceability [25] has identified different usage scenarios of traceability, which include diverse tasks such as requirements management, change management and impact analysis, verification and validation, testing, reuse, system understanding, audit, and software project management.

A traceability scheme or metamodel provides a realization of the abstract definitions of traceability. It determines details required to support traceability, such as which artefacts should be traced, the level of details for the traces, and how traceability links should be classified. According to Ramesh and Jarke [20], there are six core questions about artefacts that should be answered by traces: What, Who, Where, When, Why, and What. Additionally, Espinoza et al. [7] provide a formal specification and state that a traceability scheme has to include a *traceability type set*, *traceability role set*, *minimal links set* - links have to exist for correctness and completeness - and a *metrics set*.

Ramesh and Jarke [20] have also introduced a conceptual model for traceability which includes three essential elements in traceability and shows their relationships (Figure 1). Their model has been the source of many later studies which largely focus on the TRACES-TO relationship or traceability links.

The trace can in part be documented as a set of metadata of an artefact (such as creation and modification dates), and in part as relationships describing the influence of a set of stakeholders and artefacts on a different artefact. These relationships are a vital concept of traceability, and they are often referred to as traceability links. Accordingly, there are many studies on identifying and classifying traceability links, such as [20, 7, 22, 4]. These proposed classifications have been introduced based on the characteristics and context of each study and so they are difficult to compare.

Supporting traceability requires spending more effort to create and update the trace links [17]. Theoretically, 'ideal' traceability would be achieved by recording all possible traces. But in real situations, economic constraints must be fulfilled. Existing traceability solutions suffer from the challenge of how much effort and information is enough for requirements traceability. Thus, it is critical to find the *right* amount of required information [6]. Project/domain-specific requirements traceability can be helpful in this regard. This is because project/domain-specific traceability goals drive traceability activities such as the type of captured information [1].

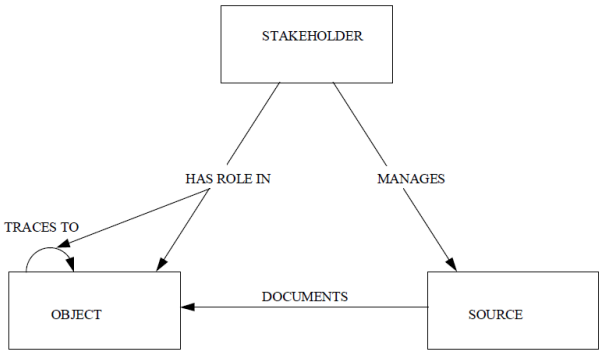

Figure 1: The traceability conceptual model according to Ramesh and Jarke [20]

On the other hand, Egyed et al. [6] state that a traceability strategy should provide trace links more quickly, refine trace links according to user-defined value considerations, and support the later refinement of trace links, and accordingly introduce value-based requirements traceability to balance cost and benefits. [3] introduces dynamic requirements traceability to minimize the need for creating and maintaining explicit links and reduce the effort required to perform manual trace. [15] provide a tool-based approach for traceability in agile development processes.

Although traceability is essential for successful software development as it helps in different ways and has diverse advantages, there are still difficulties and challenges, such as lack of a commonly accepted traceability definition [20], finding the "right" level of granularity for traceability [6], and lack of uniformity in requirements tools [12]. Consequently, traceability is usually considered extra effort.

This paper proposes a domain/project-specific requirements traceability approach to support agile requirements traceability, while addressing mentioned requirements traceability challenges. The next section explains the proposed approach.

3. DOMAIN-SPECIFIC REQUIREMENTS TRACEABILITY

This section introduces the proposed DSML-based approach for requirements traceability: *Domain-Specific Requirements Traceability* and discusses how this approach addresses the requirements traceability challenges while supporting agility.

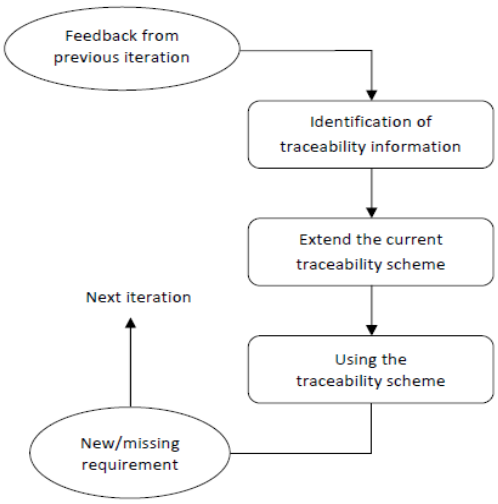

Figure 2: The conceptual process model for defining traceability scheme

As mentioned in Section 2, one of the most important concerns in traceability is finding a suitable traceability scheme. A proper scheme should support traceability goals and, at the same time, be effective, for example regarding economical issues. To address this challenge, we suggest defining the traceability scheme specifically for the current project, target domain and traceability goals - that is, using a DSML. Additionally, we suggest to define the traceability metamodel incrementally through iterations which allows its users to capture the most appropriate type of captured information based on their current needs.

3.1 The Traceability Scheme

As mentioned earlier, the traceability scheme is defined incrementally through iterations. Generally, in each iteration the traceability information is identified regarding the current needs, the traceability scheme is extended according to the new requirements, and then the scheme is used for traceability. Traceability information (mainly includes artefacts) is usually defined according to stakeholders needs, the development lifecycle for the project or the general development process for the specific domain, and usage scenarios (traceability goals). Extending the existing scheme might need to provide evolution and change strategies for maintaining existing trace models properly. Each iteration finishes when a change is identified either because of a missing or a new requirement and, consequently, a new iteration begins. This cycle is repeated over the project and the traceability scheme will be extended whenever needed. Figure 2 shows the conceptual process model for defining traceability scheme.

The extension is performed as necessary to identify the essential concepts and the link types in the domain/project based on the traceability goals (i.e. change impact analysis or validation and verification); extensions are performed until the scheme is sufficient to start modelling traceability.

We suggest to use the conceptual model for requirements traceability introduced by Ramesh and Jarke [20] as the base traceability scheme and extend it incrementally. Each of the elements and link types is extended as indicated by traceability requirements.

The most challenging part of the conceptual model is identifying subclasses of OBJECT. OBJECTs constitute the core of the traceability metamodel and usually represent different types of artefacts in the project. Incremental traceability DSML allows us to elaborate this essential part (OBJECTs) and identify new OBJECTs and expand the metamodel whenever it is required (i.e. new traceability goal or missing concepts).

In addition to the concepts, the conceptual model also shows the relationships which exist between these concepts. The relationships highlight the traces which are important to support traceability goals. Moreover, there are two types of links between concepts: *direct* and *transitive* link (dashed line). Direct links are those that directly link two concepts with no other intermediate concept, while transitive links are those that can be derived from other direct links. The transitive links enhance can be used to provide automation in cases where the traces can be derived automatically.

In this regard, the traceability scheme is always subject to change and evolution along the time. So, evolution and change in the scheme are the two main concerns in this approach; how does the scheme change? how is the change applied in the already exist traceability information? Traceability information is recorded according to what the scheme dictates. So, when the scheme changes, the change would impact existing traceability information, such as requiring new information to be recorded or change in the exiting links between elements, which should be managed properly. Therefore, it is important to consider change and its impact in both traceability scheme and the traceability model.

3.2 Representation

Currently, requirements traceability is generally supported by requirements management tools such as DOORS and RequisitePro or through small and simple research-based tools to support traceability, such as [15]. All of these tools provide a common traceability scheme and are mostly interested in the links between requirements and design models to determine rationale and decision making traces. None of the existing tools support the introduced approach in this paper. They do not allow to have a user-defined traceability scheme. This paper applies a different approach for the tool. It suggests to use the model-driven engineering (MDE) facilities to represent the traceability scheme and support traceability.

We suggest to develop a domain-specific modelling language for the traceability scheme, which has been defined regarding the domain/project and traceability goals. We propose to use Ecore metamodel [23], to describe the traceability metamodel as a modelling language, and Epsilon as the model management tool. Epsilon (Extensible Platform for Specification of Integrated Languages for mOdel maNagement) [13] is a suite of tools and comprises a number of integrated model management languages, for performing tasks such as model merging, model transformation and intermodel con-

sistency checking. EMF and Epsilon have been developed for MDE tasks and provide specialised powerful facilities to work with models. For example, the Ecore model can be annotated to use EuGENia to automatically generate an editor to provide an easier way to generate models based on a user-defined metamodel.

3.3 Application

The developed DSML (the traceability metamodel) can be used for different purposes which are identified as the usage scenarios for requirements traceability. Usage scenarios can be general scenarios or can be defined according to the domain or project. Regarding the context of this paper and the proposed solution, Epsilon model management languages can be used to deal with models to support these scenarios. For example Eclipse Object Language (EOL) is used to enrich the traceability models based on the derived information from other models, Eclipse Generation Language (EGL) can be used to generate Requirements Document (RD) in the desired format, and Epsilon Validation Language (EVL) is helpful in validation and verification or change impact analysis.

3.4 Discussion

Project/domain-specific traceability goals allow to focus on the main concerns in the domain/project and prevent to spend resources on recording information which are not related, essential, and helpful for traceability. Moreover, defining the traceability metamodel incrementally leads to find the most appropriate type of captured information based on the current needs. Therefore, the traceability metamodel, at any time, is as simple as possible and contains the minimum required information. This is because it identifies the *essential* elements regarding the *current* needs. This also complies with agility: providing a simple and light enough traceability scheme which is achieved through an iterative and incremental approach in defining the scheme.

On the other hand, using DSMLs for traceability requires us to consider change and evolution issues in the context of MDE. Evolution in MDE is an open research topic. [5] highlights multiple dimensions for evolution in MDE. One of these dimension is called artefact co-evolution and focuses on changes made to MDE development artefacts which may require that related artefacts (models, code, documentation) be updated to remain synchronised. Traceability in MDE and change impact analysis are the two potential research solutions in this case. In MDE, traceability is an active topic and focuses on traces recorded automatically as a by-product of model transformations. Traces recorded that way can be used to keep the models consistent and to support change propagation between models which are derived from each other. There is much research in this field, for example [18, 1, 4].

Additionally, model co-evolution is one of the important dimensions regarding the scope of this paper. In this case, when a metamodel evolves, any instance models may require migration to remain conformant. Automated migration and appropriate notation for describing model co-evolution are examples of the challenges. [24, 2] are the two approaches addressing these problems. There is also a recent study which aims to introduce structures and processes for managing model-metamodel co-evolution to increase productivity [21].

The next section discusses some preliminary results from applying the proposed approach in the safety-critical software system domain.

4. EXAMPLE

This section provides a brief description of a preliminary example of using the proposed approach in the safety-critical software systems domain, in which traceability is both a mandatory requirements of standards (i.e. DO-178B) and helpful in addressing engineering challenges, such as dealing with changes in requirements and their consequent effects [19], and supporting evidence-based safety assurance [9].

Following the introduced approach, the traceability metamodel was gradually developed, taking into account the characteristics of the target domain which was identified through investigating the domain, the general development process, case studies, and the traceability goals.

In addition to the general software system development activities, to develop a safe system (1) the system hazards are identified and the safety requirements are defined, (2) it is determined how the various components in the system can contribute to these hazards, (3) derived safety requirements for the components are defined, and then (4) components are developed to meet these safety requirements [10]. The system is also certified before entering the operational environment

Furthermore, in this example, we focus on change management and certification challenges, as the general and domain-specific traceability goals. In the context of safety-critical software, change management is more challenging in comparison to general software development. This is because *safety is not compositional* and any change generally requires having to perform the safety process again. Additionally, the system should be certified before entering the operational environment to show that it is acceptably safe, which normally requires creating a safety case. A safety case is a comprehensive document which refers to many and various pieces of information, such as safety analysis, acceptance test, compliance with standards [11].

Accordingly, the resulting traceability metamodel for safety-critical domain is depicted in Figure 3.

As mentioned earlier, there are several general and domain-specific usage scenarios for the traceability model. In our example, EGL was used to generate the Requirements Document which is updated automatically regarding changes in the model. Additionally, regarding the importance of safety case in safety-critical software system development, the traceability model is used to support safety case (evidence and arguments). In this example, a new DSML developed for safety case development based on GSN [11]. The new DSML is linked to the traceability DSML, in the metamodel level. Therefore, the safety case model is linked to the traceability model which helps in providing valid evidence and arguments in the safety case. This is because the evidence and arguments are provided according to the

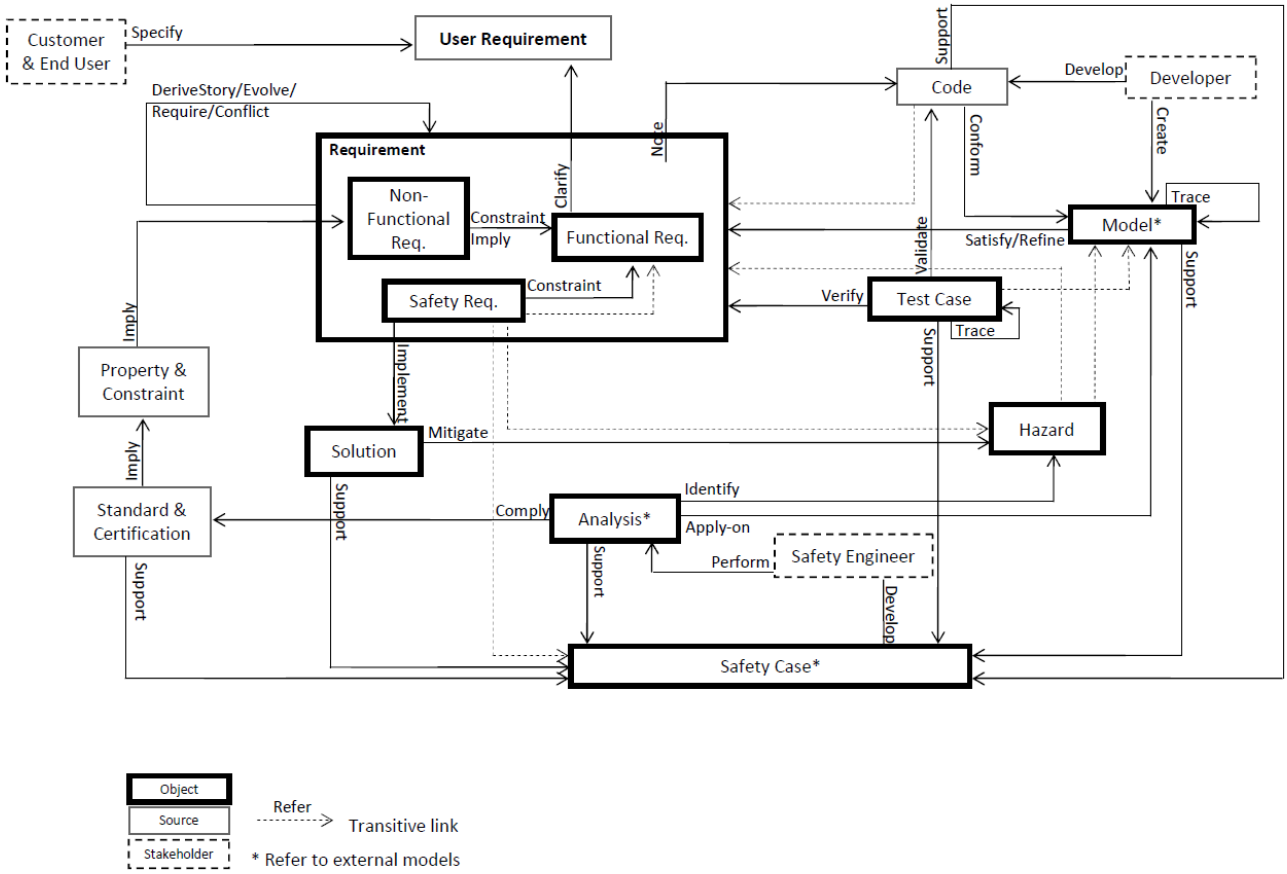

Figure 3: The traceability scheme for safety-critical domain

traceability model and updated automatically if there is any change in the model. In this case, EOL, EVL, and ETL were used to support this domain-specific usage scenario.

5. FUTURE WORK

This paper proposed to use DSMLs to incrementally define requirements traceability schemes according to a specified domain and, at the same time, benefit from MDE technology to automate the scheme and, potentially, evolve it over time. As discussed in Section 3, this approach leads to consider the evolution and change concerns in the context of MDE, too. Then it was suggested to study the existing work in this domain and apply them for evolution and extension mechanisms. This is an acceptable approach because by using DSMLs, the requirements traceability problem is transformed to ta problem in the MDE domain. Nevertheless, the requirements engineering issues are still important and should be considered.

By the way, evolution and extension mechanisms need more investigation and work. This is because they are still an active and challenging research areas in MDE, and it would be good to work on domain-specific (requirements traceability) model evolution and extension mechanisms to provide specialised mechanisms for agile requirements traceability. Additionally, the research needs more case studies, in different domains, to show the applications and benefits of the proposed solution. Case studies are also useful for evaluation, though other evaluation means, such as peer review, should be considered.

Improved tool support is also one of the future work for this research. Currently, the traceability tool is produced with Epsilon. Although this tool is simple and easy to use, a more user friendly tool is better which provides graphical interfaces or wizards for dealing with models instead of working with Epsilon model management languages directly.

6. CONCLUSION

Requirements traceability is studied as an effective mechanism to deal with various software development challenges, such as verification and validation and change impact analysis. However, traceability still faces difficulties and challenges, such as finding the “right” level of granularity for traceability and lack of uniformity in requirements tools.

We proposed an agile approach to requirements traceability in MDE through a *domain-specific requirements traceability approach* to provide *just enough* traceability, while addressing requirements traceability challenges. We suggested to use a DSML to describe a domain-specific traceability metamodel and represent the structures, behaviours and features of the target domain with respect to project traceability goals. Then, MDE tooling is used to both create traceabil-

ity models and to help evolve models and metamodels as project-specific traceability requirements change. This approach allows engineers to focus on essential domain-specific traceability information to get the most value out of traceability, especially in comparison to general-purpose traceability frameworks and tools. We also identified challenges for manipulating trace models and metamodels in this context, particularly for supporting incremental development of the DSML.

Finally, the paper presented the preliminary work on this topic, focusing on the DSML and how it contributes to other system engineering activities in the context of building safety-critical systems.

7. REFERENCES

[1] N. Aizenbud-Reshef, R. F. Paige, J. Rubin, Y. Shaham-Gafni, and D. S. Kolovos. Operational Semantics for Traceability. In *Proceedings of ECMDA Traceability Workshop*, ECMDA-TW '05, pages 8–14. Sintef, 2005.

[2] A. Cicchetti, D. D. Ruscio, R. Eramo, and A. Pierantonio. Automating Co-evolution in Model-Driven Engineering. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, EDOC '08, pages 222–231, Washington, DC, USA, 2008. IEEE Computer Society.

[3] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou. Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, RE '05, pages 135–144, Washington, DC, USA, 2005. IEEE Computer Society.

[4] A. Dahlstedt and A. Persson. Requirements Interdependencies: State of the Art and Future Challenges. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, pages 95–116. Springer-Verlag, 2005.

[5] A. V. Deursen, E. Visser, and J. Warmer. Model-Driven Software Evolution: A Research Agenda. In *Proceedings of International Workshops on Model-Driven Software Evolution held with the ECSMR '07*, 2007.

[6] A. Egyed, P. Grunbacher, M. Heindl, and S. Biffl. Value-Based Requirements Traceability: Lessons Learned. In *Proceedings of the 15th IEEE International Conference on Requirements Engineering*, RE '07, pages 115 –118, oct. 2007.

[7] A. Espinoza, P. Alarcon, and J. Garbajosa. Analyzing and Systematizing Current Traceability Schemas. In *Proc. 30th Software Engineering Workshop*, april 2006.

[8] O. C. Z. Gotel and C. W. Finkelstein. An Analysis Of The Requirements Traceability Problem. In *Proceedings of the 1st International Conference on Requirements Engineering*, RE '94, pages 94 –101, apr 1994.

[9] I. Habli and T. Kelly. Process and product certification arguments: getting the balance right. *SIGBED Rev.*, 3(4):1–8, Oct. 2006.

[10] M. P. E. Heimdahl. Safety and software intensive systems: Challenges old and new. In *2007 Future of Software Engineering*, FOSE '07, pages 137–152, Washington, DC, USA, 2007. IEEE Computer Society.

[11] T. Kelly and R. Weaver. The Goal Structuring Notation - A Safety Argument Notation. *Elements*, 2004.

[12] V. Kirova, N. Kirby, D. Kothari, and G. Childress. Effective Requirements Traceability: Models, Tools, and Practices. *Bell Labs Technical Journal*, 12:143–157, February 2008.

[13] D. Kolovos, L. Rose, and R. Paige. *The Epsilon Book*. Dec. 2010.

[14] P. Lago, H. Muccini, and H. van Vliet. A Scoped Approach To Traceability Management. *Journal of Systems and Software*, 82(1):168 – 182, 2009. Special Issue: Software Performance - Modeling and Analysis.

[15] C. Lee and L. Guadagno. FLUID:Echo Agile Requirements Authoring and Traceability. In *Proceedings of the Midwest Software Engineering Conference*, pages 50–61, 2003.

[16] R. R. Lutz. Software Engineering For Safety: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 213–226, New York, NY, USA, 2000. ACM.

[17] P. Mäder, O. Gotel, and I. Philippow. Getting back to basics: Promoting the use of a traceability information model in practice. In *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, TEFSE '09, pages 21–25, Washington, DC, USA, 2009. IEEE Computer Society.

[18] R. Paige, G. Olsen, D. Kolovos, S. Zschaler, and C. Power. Building Model-Driven Engineering Traceability Classifications. In *Proceedings of ECMDA Traceability Workshop*, ECMDA-TW '08, pages 49–58. Sintef, 2008.

[19] R. F. Paige, A. Galloway, R. Charalambous, X. Ge, and P. J. Brooke. High-integrity Agile Processes For The Development Of Safety Critical Software. *International Journal of Critical Computer-Based Systems*, 2:181–216, July 2011.

[20] B. Ramesh and M. Jarke. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27:58–93, 2001.

[21] L. M. Rose. *Structures and Processes for Managing Model-Metamodel Co-evolution*. PhD thesis, University of York, 2011.

[22] G. Spanoudakis and A. Zisman. Software Traceability: A Roadmap. In *Handbook of Software Engineering and Knowledge Engineering*, pages 395–428. World Scientific Publishing, 2004.

[23] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley, Boston, MA, 2. edition, 2009.

[24] G. Wachsmuth. Metamodel Adaptation And Model Co-adaptation. In *ECOOP '07, LNCS 4609*. Springer, 2007.

[25] S. Winkler and J. Pilgrim. A Survey Of Traceability In Requirements Engineering And Model-Driven Development. *Software and Systems Modeling (SoSyM)*, 9:529–565, September 2010.