Modeling Traceability in System of Systems

Bedir Tekinerdogan
Information Technology Group
Wageningen University
Wageningen, The Netherlands
bedir.tekinerdogan@wur.nl

Ferhat Erata
Information Technology Group
Wageningen University
Wageningen, The Netherlands
ferhat.erata@wur.nl

ABSTRACT

An important aspect in SoS is the realization of the concerns in different systems that work together. Identifying and locating these concerns is important to orchestrate the overall activities and hereby to achieve the overall goal of the SoS. Moreover, concerns in SoS are rarely stable and need to evolve in different ways and different times in accordance with the changing requirements. To manage the SoS and cope with the evolution of concerns it is necessary that the dependency links between the concerns and the system elements can be easily traced. In this paper, we present the different traceability requirements and the corresponding metamodel to support modeling traceability and supporting traceability analysis approaches within the SoS context.

Keywords

System of Systems, Traceability, Metamodel

CCS Concepts

•**Software and its engineering** → **Traceability;** *Software evolution;*

1. INTRODUCTION

Whereas traditionally systems were addressing a single domain current systems have to be composed of multiple systems that need to be integrated in a coherent way. In addition, the configuration is not static but requires also the dynamic adaptation. To be able to design, analyze, implement and maintain such large so-called systems of systems (SoS), a Systems of Systems Engineering (SoSE) approach is required Traditionally, SoSE methodology is heavily used in the defense domain but is now also increasingly being applied to non-defense related problems such as architectural design of problems in air and auto transportation, healthcare, global communication networks, search and rescue, space exploration and many other SoS application domains [5]

SoSE is more than systems engineering of monolithic, complex systems because design for SoS problems is performed under some level of uncertainty in the requirements and the constituent systems, and it involves considerations in multiple levels and domains. Whereas systems engineering addresses the development and operations of monolithic products, SoSE addresses the development and operations of evolving systems. In other words, traditional systems engineering seeks to optimize an individual system (i.e., the product), while SoSE seeks to optimize network of various interacting legacy and new systems brought together to satisfy multiple objectives of the program. SoSE should enable the decision-makers to understand the implications of various choices on technical performance, costs, extensibility and flexibility over time; thus, effective SoSE methodology should prepare decision-makers to design informed architectural solutions for SoS problems.

An important aspect in SoS is the realization of the capabilities in different systems that work together. Tracking these capabilities is important to orchestrate the overall activities and hereby to achieve the overall goal of the SoS. In addition to the complexity of the SoS we have also to cope with the evolution of the system which could be at different times and places in the SoS. Capabilities in SoS can evolve in accordance with the changing requirements and as such can change or need to be reallocated to other systems in the SoS.

To manage the complexity and the evolution of SoS it is necessary that the dependency links between the capabilities and the system elements can be easily traced. This is because changes to capabilities as such can have consequences for the overall SoS performance or for other system elements, which are directly or indirectly related to it.

Obviously, to address the above problem we need to be able to track and trace the implemented capabilities. This is not only important in the initial construction or integration of the SoS, but also in later phases in which the systems can evolve independently.

The notion of traceability is not new and much research has been carried out in different domains including requirements engineering, model-driven development, and aspect-oriented software development [9, 1]. However, traceability of concerns in SoS has not yet been tackled in depth. In this

SAC 2017,April 03-07, 2017, Marrakech, Morocco

http://dx.doi.org/10.1145/3019612.3019908

paper, we aim to address this problem by providing a list of requirements for addressing traceability in SoS. Based on these requirements we propose a metamodel for traceability in SoS that can be used to support systematic traceability analysis approaches.

The remainder of the paper is organized as follows. Section 2 illustrates the need for Traceability along with a case study. In section 3, we present the requirements for traceability within the context of SoS. In section 4, we present the traceability metamodel for SoS. In section 6, we present the related work and finally in section 7 we conclude the paper.

2. SMART CITY ENGINEERING

To illustrate the need for traceability in SoS we will adopt the case study for smart city engineering [13]. It is expected that the gross of the world population will live in urban cities in the near future. This will have a huge impact on future personal lives and mobility. A smart city uses information and communication technology (ICT) to enhance the quality and performance of urban services, to reduce costs and resource consumption, and to engage more effectively and actively with its citizens. Sectors that have been developing smart city technology include government services, transport and traffic management, water and waste, health care, and energy. Smart city systems typically can be considered as a SoS since it consists of multiple systems that are integrated to achieve a given goal. Smart city applications are developed with the goal to improve the management of urban flows and allowing for real time responses to challenges. To illustrate the problem of traceability within and across systems within SoS we define a set of change scenarios:

Reduce Waste and Pollution. Waste and pollution prevention and reduction is an important aspect of a city to enhance the well-being of the city habitants. In general multiple system units will need to have the capability for monitoring and controlling the waste and pollution in a city. To optimize this process it is important that each system unit is traced for this capability.

Enhance security. Security is usually an important concern in a city. For enhancing the overall security in a city it is important that we have a clear picture on the system units that have an impact on supporting security, that is, that require the security capability.

Optimize Help in Emergency. A city can cope with different emergency states that require a coordinated action of the system units. To support the effective and efficient communication and emergency handling the different capabilities related to this need to be well-defined and allocated over the different system units.

The above scenarios are selected examples that could be required in a SoS and we could easily identify several other scenarios that impact the SoS and that require traceability of the capabilities. In general, we encounter the following two problems in realizing such kind of changes.

Identification of the capabilities. First of all, each change of a capability requires the identification of the systems that implement the capabilities. For example, for realizing the scenario enhance security we need to identify all the system elements that are related to security. For the scenario reduce pollution we need to identify the system elements that are related to pollution. In some cases we could derive from the names of the architectural elements which concerns are implemented, however, like in this case this is usually not that straightforward. Moreover, each concern might also map to more than one architectural element.

Orchestration of Capabilities. Identifying the distribution and allocation of capabilities to system elements is not sufficient in case we also need to optimize the systemic properties. For this reason, the state of the capability in each system needs to be communicated and orchestrated with the state of the capabilities of other system elements. Only then we could manage the systemic properties. In the other case local optimization at the single system level could be achieved but the global systemic level optimization would be largely ignored.

3. REQUIREMENTS FOR TRACEABILITY

Based on the work on the literature on traceability and SoS we provide a set of requirements for traceability of the capabilities.

Explicit Modeling of Concerns. In order to explicitly reason about traceability of the capabilities it is necessary that the corresponding capabilities are explicitly modeled as first class abstractions. The detail of concern model could range from just a description of its name to a full semantic model including attributes such as stakeholder, the domain of the capability, the date at it was raised, the impact that it has, etc.

Explicit Modeling of Dependency Relations. Assuming that capabilities are related to system units, it is necessary to make these relations explicit. This can be only done when dependency relations are recorded as traceability links. For this, traceability should be specified as first class abstractions in the adopted traceability model.

Support for Automated Tracing Queries. The explicit models for capabilities and the traceability help to define the links between the different concerns and the system units. In case traceability links among capabilities, the tracing can only be done in an implicit manner. By providing the traceability links, capabilities can be more easily traced by just following the tracing links. Nevertheless, for a complex system following the traceability links manually might not be trivial. Even though the traceability links are made explicit it may be hard to expose the required traceability links. To support the tracing the system should provide automated support for defining generic and user-defined queries to identify and trace the capabilities.

Support for Traceability within and across System Boundaries. An SoS typically consists of multiple integrated systems or system units. Tracing should be supported within and across life system units. Obviously, capabilities will be allocated and realized in various systems of the SoS. To understand the relations among the capabilities and system units, it is necessary to model traceability for the given system unit. Figure 1 shows the abstract model

for tracing within a system. We define her two types of traceability: (1) *intra capability to system unit traceability* (2) *intra system unit to capability traceability.*

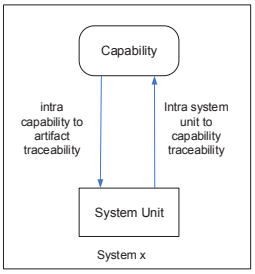

Figure 1: Traceability within a single System

Figure 2 and Figure 3 presents the abstract model for traceability relationships across system units. In Figure 2 two types of relations are defined that we think are necessary. *Inter capability to capability traceability* defines the traceability of a capability in one system to another capability of another system. *Inter system unit traceability* defines the traceability of a system unit in one system to another system unit in another system.

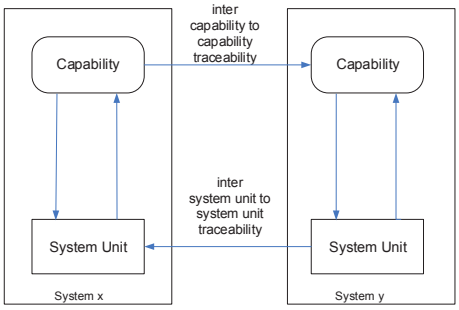

Figure 2: Traceability across Systems in SoS

Figure 3 adopts the traceability between systems, but here the relations are defined from a capability to a system unit and vice versa. It should be noted that, the relations from Figure 3 can be also derived from the traceability relations of Figure 1 and Figure 2. The traceability relations of Figure 3 can be practical to optimize the tracing.

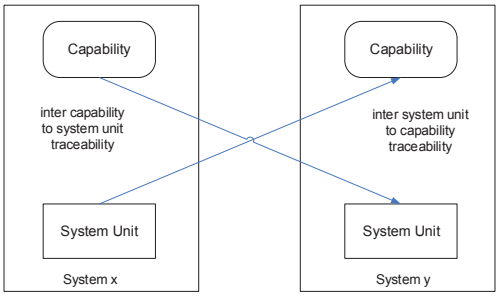

Figure 3: Traceability across Systems in SoS

4. METAMODEL FOR TRACEABILITY

In the following we present the traceability metamodel for tracing capabilities in SoS as depicted in Figure 4. The metamodel represents SoS & capability modeling and tracing modeling. The metamodel should be preferably read from the bottom to the upper part.

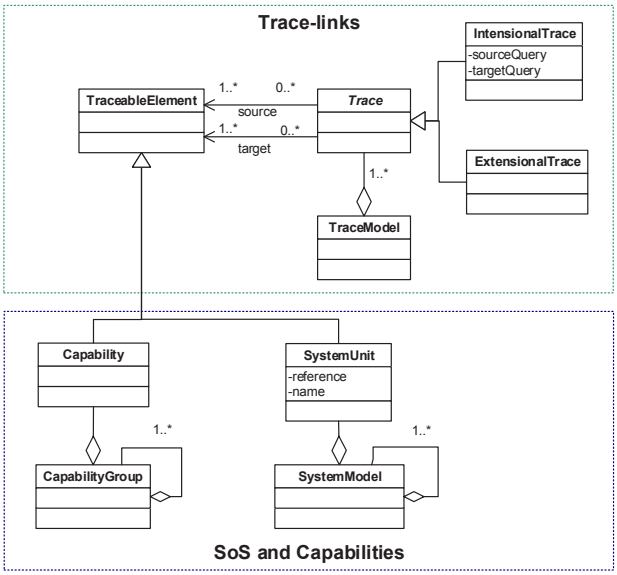

Figure 4: Traceability Metamodel for SoS

5. APPROACH

In the previous sections, we have described the properties of SoS, the different types of SoS, the different traceability requirements and the metamodel. Based on these we can build the traceability approaches. A traceability approach can in essence include two different activities including traceability modeling and traceability analysis approach.

In the traceability modeling process the focus will be on defining the trace links between the elements that need to be traced. In the previous section, we have discussed the need for tracing capabilities within and across system units in SoS.

Traceability analysis aims to trace a capability or system unit for different purposes. This could be for example, for checking the consistency or for analyzing the impact of a change of a system unit or capability in the SoS. For implementing a traceability modeling and analysis approach we need to consider the different types of SoS types. The different traceability requirements will be similar for all types of SoS. The difficulty for traceability approach will be however different for the different SoS types.

In the directed SoS defining trace links and such supporting traceability analysis seems to be the most feasible. This is because in directed SoS we have an upfront directed engineering effort for integrating a set of systems. Hereby when designing the system traceability could be considered as an explicit concern thereby realizing the trace links beforehand, and if possible also prepare the trace queries.

6. RELATED WORK

The topic of traceability is not new and has been discussed in various domains. The IEEE provides the following definition of traceability [2]: "Traceability is 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 one another; for example, the degree to which the requirements and design of a given software component match." In requirements engineering lots of work has been done on tracing requirements from the stakeholders and in the design process [6, 7].

In the model-driven engineering approach [1, 3] traceability is considered important for tracing model elements [12]. The problem of traceability has also been addressed by the AOSD community [4, 8] encompassing the adoption of aspects throughout the lifecycle.

In our earlier work we have focused on modeling traceability within architecture views [8, 9, 10] and for different lifecyle activities [11]. In this paper we have neither focused on system views nor on the lifecycle activities. Both dimensions would imply an elaboration of the traceability requirements, the metamodel and also the approach. We consider this as part of our future work.

In a SoS the capabilities will be typically distributed and allocated over different system units. Some capabilities will have a more systemic character cannot be localized in one system unit and be scattered over different system units. We could state that some capabilities will have a crosscutting property [4, 8]. Aspect-Oriented Software Development provides solution for the crosscutting problem and could as such be used to solve the problem of crosscutting capability problems.

7. CONCLUSION

Traceability is an important quality factor that has been addressed in various domains to improve other quality factors such as understandability, maintenance and adaptability. In this paper, we have built on the general literature on traceability and explored traceability within the context of system of systems (SoS). We have provided a metamodel for traceability, the key requirements for traceability in SoS, and the approaches for traceability analysis. We have used the classification of SoSs in the literature including directed SoS, acknowledged SoS, Collaborative SoS, and Virtual SoS. We have seen that modeling traceability in directed SoS will be the easiest thanks to a strong central management and the possibility to design the system for traceability earlier on. Virtual SoS does not have a central management, no agreed purpose and the systems in the SoS are largely independent. This makes it difficult to integrate traceability analysis in such systems. Our future work will elaborate on providing design abstractions for traceability in SoS and adopting the presented ideas in a real SoS.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge networking support by European Cooperation in Science and Technology (COST) Action IC1404 "Multi-Paradigm Modelling for Cyber-Physical Systems".

9. REFERENCES

[1] L. Bondé, P. Boulet, and J.-L. Dekeyser. Traceability and interoperability at different levels of abstraction in model transformations. In *forum on specification and design languages, FDL*, volume 5, 2005.

[2] P. Bourque, R. E. Fairley, et al. *Guide to the Software Engineering Body of Knowledge: Version 3.0*. IEEE Computer Society Press, 3rd edition, 2014.

[3] J. Champeau and E. Rochefort. Model engineering and traceability. In *Workshop SIVOES-MDA, UML'03*, 2003.

[4] R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, M. P. Alarcon, J. Bakker, B. Tekinerdogan, S. Clarke, and A. Jackson. Survey of analysis and design approaches. *Network of Excellence AOSD-Europe*, 2005.

[5] J. S. Dahmann and K. J. Baldwin. Understanding the current state of us defense systems of systems and the implications for systems engineering. In *Systems Conference, 2008 2nd Annual IEEE*, pages 1–7. IEEE, 2008.

[6] O. C. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *First International Conference on Requirements Engineering (ICRE'94)*, pages 94–101, April 1994.

[7] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, January 2001.

[8] B. Tekinerdogan. ASAAM: Aspectual software architecture analysis method. In *4th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 5–14. IEEE, 2004.

[9] B. Tekinerdogan, C. Hofmann, and M. Aksit. Modeling traceability of concerns for synchronizing architectural views. *Journal of Object Technology*, 6(7):7–25, 2007.

[10] B. Tekinerdogan, C. Hofmann, and M. Aksit. Modeling traceability of concerns in architectural views. In *International Workshop on Aspect-oriented modeling*, volume 209, pages 49–56. ACM, 2007.

[11] B. Tekinerdogan, C. Hofmann, M. Aksit, and J. Bakker. Metamodel for tracing concerns across the life cycle. In *Early Aspects: Current Challenges and Future Directions*, pages 175–194, Vancouver, Canada, March 2007. Springer.

[12] K. Van Den Berg, B. Tekinerdogan, and H. Nguyen. Analysis of crosscutting in model transformations. In *European Conference on Model-Driven Architecture, Traceability Workshop*, pages 51–64, July 2006.

[13] Y. Yoshikawa, A. Sato, S. Hirasawa, M. Takahashi, and M. Yamamoto. Hitachi's vision of the smart city. *Hitachi Review*, 61:111–118, 2012.