Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2014) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Bridging the Gap between Systems and Software Engineering by Using the SPES Modeling Framework as a General Systems Engineering Philosophy

Wolfgang Böhm[a], Stefan Henkler[b], Frank Houdek[c], Andreas Vogelsang[a], Thorsten Weyer[d]

[a]*Technische Universität München, Institut für Informatik, München, Germany*
[b]*OFFIS Institut für Informatik, Oldenburg, Germany*
[c]*Daimler AG, Ulm, Germany*
[d]*University of Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Essen, Germany*

Abstract

A multitude of disciplines is involved in the engineering of embedded systems. One major challenge of engineering these systems is to consider the synchronization between different engineering disciplines on the process and artifact level in a coherent manner. Process standards address this challenge by defining a transition between the involved activities of the disciplines but they do not provide sufficient support with respect to the corresponding artifacts and their relationships. The missing artifact-oriented integration of the engineering disciplines (e.g. electrical engineering, software engineering, and mechanical engineering) within the systems engineering process leads to error-prone and cost-intensive synchronizations. In recent years, we participate in a consortium of more than twenty partners from academia and industry that jointly developed the SPES (Software Platform Embedded Systems) Modeling Framework (SPES MF) to support a seamless artifact-based engineering of software for embedded systems. In this paper, we present an approach that fosters the close integration of systems engineering and software engineering activities based on ISO/IEC 15288 and ISO/IEC 12207, by using the core concepts of the SPES MF as a general engineering philosophy for the architectural design of embedded systems.

Selection and peer-review under responsibility of the University of Southern California.

Keywords: Model-based systems engineering, software engineering, engineering methodology, embedded systems, embedded software

Selection and peer-review under responsibility of the University of Southern California.

1. Introduction

A number of approaches and standards have been proposed that addresses the methodology of systems and software engineering processes as well as their integration. For instance, the ISO/IEC 12207[1] is a standard that defines software engineering processes, while ISO/IEC 15288[2] is a standard that relates to systems engineering processes. The latter also suggests how to integrate ISO/IEC 12207-compliant software engineering processes with ISO/IEC 15288-compliant systems engineering processes. This integration is highly desirable because embedded systems become more and more complex and specific challenges, such as ensuring safety and real-time properties, can only be met if systems and software engineering activities are seamlessly integrated.

Existing standards such as ISO/IEC 15288[1], ISO/IEC 12207[2] as well as ISO/IEC 29148[14] that aim at integrating systems and software engineering on a process level, typically try to achieve integration by proposing activities and corresponding relationships between them within the related disciplines. This means, for each activity it is defined which other activities need to be performed as a prerequisite, since these activities provide necessary input. The existing process approaches, which consider also artifacts for describing the input and output of the activities, do this only on an abstract level (e.g. V-Model_XT[3]). This way of integrating systems and software engineering activities goes along with the problem that the related engineering disciplines may potentially have different ideas on how to structure and refine the underlying input and output artifacts which leads to error prone and cost-intensive synchronizations. On the other hand, the artifact-oriented models which are defined in a formal way (e.g.[4,5,6,7]) lack support for a recursive system structure as defined in ISO/IEC 15288. Furthermore, these approaches typically have only a focus on relations between some specific artifact types of different disciplines and their automatic transformation abilities[8] or the integration are done in a loosely coupled way with less methodological support[9].

We suggest to use an artifact-oriented engineering paradigm as a means for closely integrating the different engineering disciplines (e.g. electrical engineering, software engineering, and mechanical engineering) within the systems engineering process of embedded systems that formally defines how the artifacts of each engineering discipline can be structured and how they are related to each other.

In recent years, we participate in a consortium of more than twenty partners from academia and industry that jointly developed the SPES Modeling Framework (SPES MF) to support the seamless engineering of software for embedded systems. The SPES MF is an artifact-oriented and model-based approach, which among others, defines artifact types and their relationships for different engineering concerns and different levels of system granularity.

In this paper, we present an approach that fosters the close integration of ISO/IEC 15288-compliant systems engineering and ISO/IEC 12207-compliant software engineering activities by using the core concepts of the SPES MF. The underlying artifact model of the SPES MF facilitates the seamless integration of the corresponding systems and software engineering processes. The explicit concept of viewpoints and granularity layers combined with the artifact model enables the SPES MF to support the ISO/IEC 15288-compliant recursive engineering of embedded systems. Note, systems engineering considers the complete life cycle of a system, including e.g. the operating phase of the system. In this paper, we focus on the architectural design phase within the life cycle.

The paper is structured as follows. Section 2 introduces the fundamentals for integrating systems engineering and software engineering activities based on the SPES MF. Section 3 describes our contribution, i.e. bridging the gap between systems and software engineering activities by applying the SPES MF. Section 4 contains conclusions and sketches our future research activities.

2. Fundamentals: ISO/IEC 15288, ISO/IEC 12207 and the SPES Modeling Framework

ISO/IEC 15288[2] takes a very broad view on the SuD and is supposed to be applicable for any man-made system. In order to manage the size and the complexity of a system, a system is divided into sub-systems that are implemented independently and finally integrated into the system. As a sub-system itself can again be considered as a system, the recursive application of ISO/IEC 15288 leads to a hierarchy of systems, sub-systems and finally their system elements up to the point, where the SuD is decomposed into a structure of manageable and understandable system elements, which can be implemented, reused, or acquired from another party. Note that the number of hierarchy levels depends on the system being developed. However, not all sub-systems in the hierarchy need to have the same number of hierarchy levels. For the implementation of the system elements (i.e. realization of the

implementation sub-process of ISO/IEC 15288), domain-specific processes may be applied. In the context of developing embedded systems, some system elements will consist of software only. The engineering of the technical platform, which is necessary to operate the software, and the integration of the software items in the hardware environment is addressed in ISO/IEC 15288. These software items can, for example, be implemented using the framework for software life cycle processes as defined in ISO/IEC 12207[1]. Here, software is treated as an integral part of the system and performs certain functions within the system. The software is implemented by extracting the software requirements from the system requirements and design, producing the software, and integrating it into the system. It is important to note that the interlacing of the processes is not necessarily sequential. The output of the processes on a higher hierarchy level will be input for the processes of the level below.

The IEEE Std. 1471[10] and its current successor IEEE Std. 42010[11] introduce a conceptual framework for architectural descriptions. The key concept of both frameworks is the architectural viewpoint (or short: viewpoint). To reduce the complexity, the architectural description of a system is typically divided into a number of interrelated views. The SPES MF[12] supports the development of software for embedded systems by differentiating between the two orthogonal dimensions *SPES MF viewpoints* and *SPES MF system granularity layers*.

The different stakeholders (e.g. requirements engineers, functional analysts, solution architects) in the engineering process of an embedded software have different concerns. Based on the separation of concerns principle, the individual concerns of stakeholders are addressed by certain views that are, in accordance to IEEE Std. 42010, governed by viewpoints in the SPES MF. The SPES MF differentiates between the following four SPES viewpoints: the "requirements viewpoint" addresses the structured documentation and analysis of requirements; the "functional viewpoint" addresses the structured documentation and analysis of system functions; the "logical viewpoint" addresses structured documentation and analysis of the logical solution and the "technical viewpoint" addresses the structured documentation and analysis of the technical solution.

To reduce the complexity of the engineering process, a coarse-grained engineering "problem" is decomposed into a number of fine-grained engineering problems following the strategy of divide and conquer. Each time, a coarse-grained engineering subject is decomposed into a number of fine-grained engineering subjects, a new granularity layer is created. SPES MF provides the mechanism to create new granularity layers that can be used by engineers to decompose the overall engineering problem to a level of granularity at which the complexity of the fine-grained systems is manageable without the need of performing another step of decomposition. The main purpose of the viewpoints and granularity layers is to structure the artifacts that are created during the development. The SPES MF takes an artifact-oriented view on the development of embedded software. This means the modeling framework defines structures, content and concepts used in artifacts rather than prescribing processes and methods. An overview on the SPES MF artifacts types can be found in Broy et al.[12]

3. Integrating Systems Engineering and Software Engineering by using the SPES MF

In this section, we describe how to integrate ISO/IEC 15288-compliant system engineering processes with ISO/IEC 12207-compliant software engineering processes using the SPES MF. As depicted in Fig. 1, we apply the horizontal and the vertical dimension of the SPES MF to structure the ISO/IEC 15288-compliant systems engineering processes based on the different viewpoints and granularity layers.

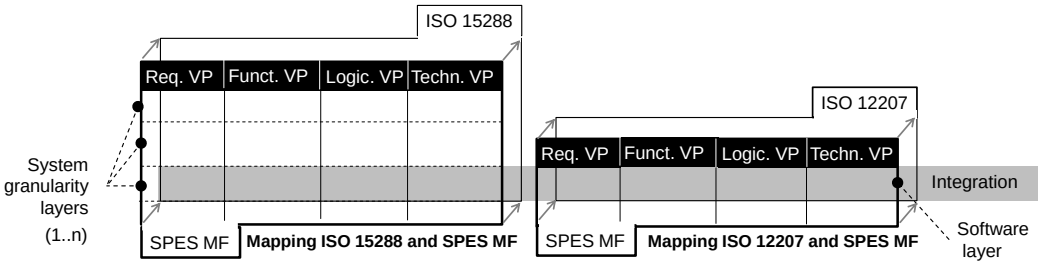

Fig. 1. Integrating ISO/IEC 15288/ISO/IEC 12207-compliant engineering processes by means of the SPES MF

When applying the SPES MF to ISO/IEC 12207-compliant software engineering processes, we also use the different viewpoints of the SPES MF to achieve a coarse-grained structuring of the engineering process. In contrast to the application of the SPES MF to ISO/IEC 15288-compliant systems engineering processes, we do not use the granularity layer concept since ISO/IEC 12207 does not consider the recursive engineering of the corresponding software element through several nested software engineering processes. This implies that the transition between an ISO/IEC 15288-compliant systems engineering process and the ISO/IEC 12207-compliant software engineering process should only be performed if the development of the corresponding software element can be realized without differentiating the software engineering process into nested software engineering processes for fine-grained elements of the software. During the engineering process, typically, the software will be decomposed into fine-grained components. However, within an ISO/IEC 12207-compliant engineering process for software, these fine-grained components do not define dedicated engineering processes on a subjacent granularity layer.

3.1. Mapping the artifact types of the SPES MF to ISO/IEC 15288 and ISO/IEC 12207 processes

When mapping the SPES MF to the ISO/IEC 15288 and ISO/IEC 12207, we have to consider two dimensions: First, we need to map the different sub-processes defined in the standards and, second, we need to understand how the framework can support the characteristics of the standards. Each process defined in the two standards defines a set of activities and lists the outcomes of the process. The SPES artifact model can be used as a reference model that captures the results of the processes. As in artifact-oriented development, the content items are independent of the development process there is a mapping of the artifact model to the development process and vice versa. This mapping is achieved by filtering the results of the process activities and abstracting from the methods to produce them. In this way, we assign the activities and milestones of the process to the artifacts in the SPES MF (see [12]). The main reason why this assignment works is due to the concept model and the strict separation of content and structure of the artifacts, which makes the SPES MF independent of the development processes applied. In [12] also a metamodel for this mapping is given. In the following table, we present a high level mapping of the processes defined in ISO/IEC 15288 and ISO/IEC 12207 to the artifacts of the SPES MF. The SPES MF artifact types that we use to integrate systems and software engineering processes are shown in bold type in the table.

Table 1: Mapping between ISO/IEC 15288 / ISO/IEC 12207 and the SPES MF

Std.	ISO process step	SPES MF artifact types
ISO/IEC 15288	Stakeholder Requirements Definition	Context Model (Req. VP), Goal Model (Req. VP), Scenario Model (Req. VP)
	System Requirements Analysis	Solution-oriented Requirements (Req. VP), Functional Black Box Model (Func. VP)
	System Architecture Design	Functional White-Box Model (Func. VP), **Logical Architecture (Log. VP)**, **Technical Architecture (Tec. VP)**
ISO/IEC 12207	Software Requirements Analysis	**Context Model (Req. VP)**, Goal Model (Req. VP), Scenario Model (Req. VP), Solution-oriented Requirements (Req. VP), Functional Black-Box Model (Func. VP)
	Software Architecture Design	Functional White-Box Model (Func. VP), Logical Architecture (Log. VP)
	Software Detailed Design	Logical Architecture (Log. VP), Technical Architecture (Tec. VP)

As shown in the table above, the implementation process defined in ISO/IEC 15288 comprises requirements and design phases for software development, which are introduced by ISO/IEC 12207. This suggests progressing in a strictly sequential manner. However, in practice, this sequential course is suspended by interleaving the requirements and design processes of ISO/IEC 15288 and ISO/IEC 12207, which is a kind of violation of the strict process orientation of the standards.

3.2. Integrating ISO/IEC 15288- and ISO/IEC 12207-compliant processes by means of the SPES MF

To integrate ISO/IEC 15288-compliant systems engineering processes with ISO/IEC 12207-compliant software engineering processes based on the SPES philosophy, we apply the two dimensions of the SPES MF together with the corresponding artifact types on both of the ISO standards. Fig. 2 illustrates our solution concept.

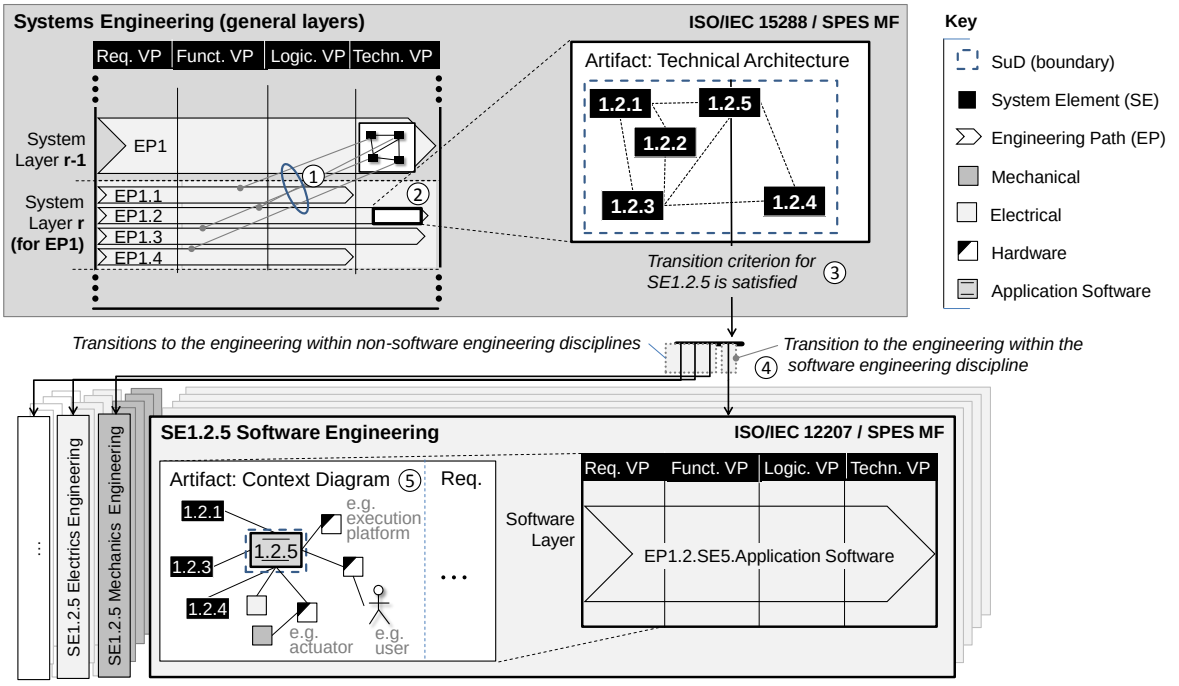

Fig. 2. Integration ISO/IEC 15288- and ISO/IEC 12207-compliant engineering processes by using the SPES MF

To illustrate our approach, we use an example from the automotive domain. The example refers to an automatic tailgate, i.e. a rear door of a station wagon that can be opened and closed electrically. The tailgate can be opened in three ways: (i) with the tailgate door handle while the car is unlocked, (ii) by pressing the tailgate button (located in the driver's door), or (iii) by pressing the open tailgate button on the vehicle's key. If the car is equipped with keyless go, the automatic tailgate can also be opened with tailgate door handle while the car is still locked (as long as the vehicle key is within the recognition zone around the car. If the car detects a collision while opening or closing the tailgate, the tailgate shall immediately stop. Collision detection bases on deviations from expected power consumption or slower movement.

3.2.1. Integrating system granularity layers into ISO/IEC 15288 by means of the SPES MF

When applying the SPES MF to ISO/IEC 15288-compliant engineering processes, we differentiate between the requirements viewpoint, the functional viewpoint, the logical viewpoint and the technical viewpoint. In addition, we use the SPES granularity layer concept to distinguish between different layers of granularity in the system engineering process. On the first layer, the technical system that has to be developed is regarded as a whole. Within the requirements view, the operational context as well as the requirements for the system are documented. In the functional view, the user functions as well as the corresponding system functions are specified from a structural and a behavioral perspective. In the logical view, the structure of a specific conceptual architecture is defined in terms of logical components and their relationships. Within the technical view, the concrete technical solution is specified that consists of technical components, their properties and relationships. Following the granularity layer concept of the SPES MF, the transition to a subjacent granularity layer is defined by the structure-defining characteristic of the corresponding granularity layer. Typically, the logical architecture (part of the logical viewpoint) or the technical architecture (part of the technical viewpoint) is used as the structure-defining characteristic for the corresponding engineering path on the subjacent granularity layer. The structure-defining characteristic defines a set of logical or technical components that are realized within dedicated engineering paths on the subjacent granularity layer of the system. Note that the source of the structure-defining characteristic can change between different granularity layers even between different engineering paths within one granularity layer. In our example, the technical architecture of the automatic tailgate describes which technical components (e.g. electronic control units, ECU) are involved in

implementing this feature. Fig. 2 shows the technical architecture of the vehicle (also called E/E-architecture[13]) and highlights the system elements that are used by the automatic tailgate system. Each technical component of the automatic tailgate system, like the power lift gate module (PLM), is regarded in a dedicated engineering path on the subjacent granularity layer (see Fig. 2 ①).

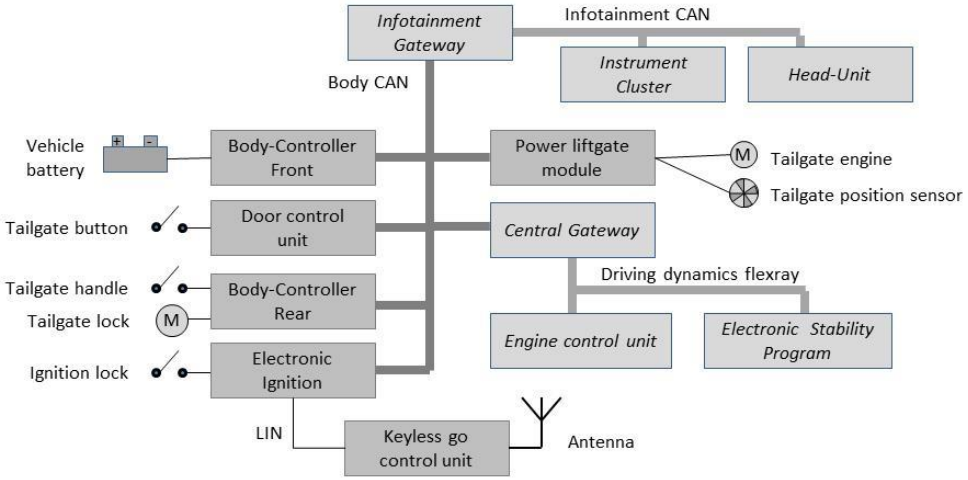

Fig. 3. The technical architecture of the automatic tailgate system and its system elements

3.2.2. Transition from ISO/IEC 15288-compliant to discipline-specific processes based on the SPES MF

In ISO/IEC 15288, the physical architecture on the lowest granularity layer consists of system elements that are developed by a set of activities defined by the standard without decomposing the corresponding system element into fine-grained parts that are engineered within separate engineering paths on a subjacent granularity layer. When applying the SPES MF to ISO/IEC 15288-compliant system engineering processes, system elements are resident either in the logical architecture within the logical view or the technical architecture within the technical view (see Fig. 2 ②). Fig. 4 shows the logical architecture of the power liftgate module on the lowest granularity layer of the ISO/IEC 15288-compliant systems engineering process and the corresponding system elements in terms of logical components.

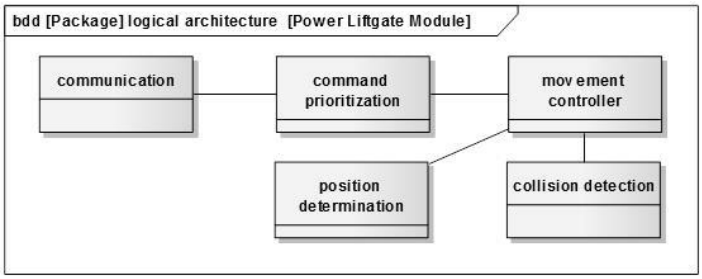

Fig. 4. Logical architecture of the power liftgate module (PLM)

For further consideration, we assume that the logical component “collision detection” is a system element, i.e. the “collision detection” is not realized by applying the recursive engineering processes as defined in ISO/IEC 15288 (see Fig. 2 ③). Typically, a system element consists of hardware (e.g. mechanical or electrical parts) and software and has to be realized in an interplay of different system engineering disciplines (e.g. mechanical engineering, electrical engineering, control engineering, and software engineering). Even if the SuD on the system layer consists exclusively of software, this is not a sufficient criterion to perform a transition to an ISO/IEC 12207-based software engineering process. Since ISO/IEC 12207 does not support recursive development, i.e. the definition of granularity layers, a large or complex software element should to be decomposed into fine-grained software elements on a system granularity layer while applying ISO/IEC 15288. The transition to the ISO/IEC 12207-based software

engineering process should be performed if the corresponding software elements do not need to be decomposed into finer-grained software elements that have to be engineered in a dedicated engineering path.

If this transition criterion is satisfied, the engineering focus changes from ISO/IEC 15288-compliant systems engineering to discipline-specific engineering processes; for instance, to an ISO/IEC 12207-compliant software engineering process for developing the application software of the corresponding system element (see Fig. 2 ④). In the remainder of this section, we focus on the transition between the general systems engineering process and the software engineering discipline. Nevertheless, the software engineering process depends on the results of the other disciplines (e.g. mechanical engineering, control engineering, electrical engineering) and vice versa. This means that during the detailed engineering of a system element, an intensive collaboration between the different engineering disciplines is required which has to be coordinated by the overall systems engineering discipline. These dependencies are modeled in the context of the software element in the SPES MF.

3.2.3. Starting point of the ISO/IEC 12207-compliant software engineering processes based on the SPES MF

The starting point of the software engineering process for a system element is the logical or technical architecture of the engineering path in which the transition decision for the corresponding system element has been made (see Fig. 2 "technical architecture" of EP1.2). The technical architecture documents the relationship of the regarded system element to other system elements. Additionally, the context diagram of the requirements view in that engineering path provides important information about the relationship of the regarded system element to system elements on a coarse-grained granularity layer and the operational environment of the overall technical system. Fig. 5 shows the context diagram of the application software for the collision detection, which becomes the SuD in the context of our software engineering process.

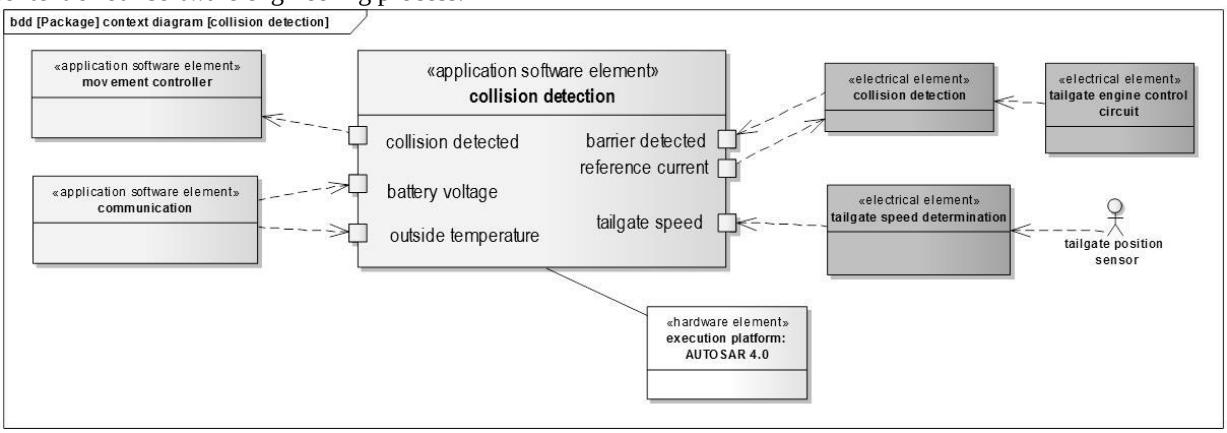

Fig. 5. Context diagram of the application software element "collision detection"

In the software engineering process, we consider the systematic development of the application software for the corresponding system element. The application software runs on an execution platform, which consists of hardware (e.g. ECUs, CPU, memory) as well as platform-related software (e.g. operating system software, device drivers). Although the engineering of platform-related software is not within the scope of this paper, the SPES MF can also be used for this purpose. Following the philosophy of the SPES MF, in the software engineering discipline the context diagram of the corresponding SuD (i.e. the application software) has to be created that documents the environment in which the application software will operate when the overall embedded system is in operation (see Fig. 2 ⑤). The context diagram shows all the operational relationships of the application software to other systems, other system elements, and the operational constraints that are defined by other engineering disciplines. Note that some context elements (like the tailgate engine hardware driver) might be elements that are not implemented by using an ISO/IEC 12207-compliant process (as they are realized, for instance, by hardware). Based on this context diagram, the goals and corresponding scenarios of the application software are elicited, analyzed and documented in the next step. The specification of the detailed requirements for the application software is based on the context diagram as well as on the goals and scenarios of the requirements view.

4. Summary and Outlook

In this paper, we presented the SPES MF as a model-based approach that bridges the gap between system and software engineering processes. More specifically, we related the basic activities defined by the system engineering process standard ISO/IEC 15288 and the software engineering process standard ISO/IEC 12207 with artifact types defined in the SPES MF. In this way, we contribute to the problem that the engineering disciplines may have different ideas on how to structure the underlying input and output artifacts. By means of our approach, we increase consistency between the system and the software engineering artifacts and enable capabilities of tracing changes and performing automated analyses and transformations, which in turn lead to increased effectiveness and efficiency.

The work we have presented so far only considers the specification part of the development processes. However, we are convinced that the artifacts and their inter-process relationships also facilitate validation and verification as well as integration and testing activities that are subject of future work. The context model, for example, has a central role in the SPES MF. Besides the documentation of the operational environment for supporting the systematic requirements elicitation, analysis and specification process within the requirements viewpoint, the context model also describes how to integrate the corresponding SuD into its operational environment. Consequently, the context model also supports the systematic integration of realized system elements and the integration of the overall embedded system in its operational environment.

Acknowledgements

This research was partly funded by the German Federal Ministry of Education and Research (BMBF), under grant “SPES XTCore” (01IS12005).

References

1. International Organization for Standardization. *ISO/IEC 15288 Systems and software engineering - System life cycle processes* (also: IEEE Std 15288-2008), Geneva.
2. International Organization for Standardization. *ISO/IEC 12207 Systems and software engineering - Software life cycle processes* (also: IEEE Std 12207-2008), Geneva.
3. Rausch A, Bartelt C, Ternité T, & Kuhrmann M. The V-Modell XT Applied–Model-Driven and Document-Centric Development. In *3rd World Congress for Software Quality*, Vol. 3; 2005. p. 131-138.
4. Gausemeier J, Giese H, Schäfer W, Axenath V, Frank U, Henkler S, Pook S, Tichy M. Towards the Design of Self-Optimizing Mechatronic Systems: Consistency between Domain-Spanning and Domain-Specific Models. In: *International Conference on Engineering Design*, ICED'07, 28-31 August, Paris, France, 2007.
5. Suh NP.: On functional periodicity as the basis for longterm stability of engineered and natural systems and its relationship to physical laws. In: *Research in Engineering Design*, Springer-Verlag, London, Februar 2004.
6. Braun P, Broy M, Houdek F, Kirchmayr M, Müller M, Penzenstadler B, Pohl K, Weyer T. Guiding requirements engineering for software-intensive embedded systems in the automotive industry. *Computer Science - Research and Development*, 2010.
7. Fernandez DM, Lochmann K, Penzenstadler B, Wagner S. A case study on the application of an artefact-based requirements engineering approach. In *15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011)*, 2011.
8. Giese H, Henkler S: A Survey of Approaches for the Visual Model-Driven Development of Next Generation Software-Intensive Systems. In: *Journal of Visual Languages and Computing*, **17**; Dec. 2006. p. 528-550.
9. Weilkins T. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. The MK/OMG Press; 2008.
10. Institute for Electric and Electronic Engineers. *IEEE Recommended Practice for Architectural Description of Software Intensive Systems*. IEEE Standard 1471-2000; New York.
11. International Organization for Standardization. *ISO/IEC/IEEE Systems and Software Engineering: Architecture description*. ISO/IEC/IEEE Standard 42010:2011; Geneva.
12. Broy M, Damm W, Henkler S, Pohl K, Vogelsang A, Weyer T: Introduction to the SPES Modeling Framework. In: Pohl, K.; Hönninger H, Achatz R, Broy M. *Model-Based Engineering of Embedded Systems*, Springer, Berlin; 2012.
13. Schäufele J, Zurawka T. *Automotive Software Engineering: Principles, Processes, Methods, And Tools*. SAE International; 2005.
14. International Organization for Standardization. *ISO/IEC/IEEE 29148 Systems and software engineering - Life cycle processes – Requirements Engineering*, Geneva.