A Requirements Engineering Content Model for Cyber-physical Systems

Birgit Penzenstadler and Jonas Eckhardt
Institut für Informatik
Technische Universität München
Munich, Germany
{penzenst,eckharjo}@in.tum.de

Abstract—The development of highly distributed Systems of Systems (SoS) poses a big challenge on the whole development process of such systems. Especially in Requirements Engineering, one has to cope with the resulting variety of stakeholders and their multitude of different and possibly contradictory goals. This is challenging for requirements elicitation, documentation, and management, especially with regard to communication and consistency.

One promising means to this challenge is to use an artefact-oriented requirements engineering approach that puts emphasis on artefacts and dependencies rather than dictating processes and methods for creating the artefacts.

In this paper, we present a content model that facilitates collaboration between stakeholders from 30 companies in the research project ARAMiS and is used on a SoS; more specifically, on a so-called Cyber-Physical System that spans a variety of application domains. The content model was elaborated iteratively on the basis of models from preliminary work and in discussion with partner representatives. It is now under evaluation by the 30 project partners, so we present a preview of the evaluation.

Keywords—requirements engineering, content model, system of systems, model-driven, artefact, guidance, cyber-physical system

I. INTRODUCTION

The software engineering community has developed methodologies to cope with the engineering process of large systems. The term System of Systems (SoS) was introduced to characterize such large systems. Shenar was one of the first to define the term in [?]; he defines SoS as “a large widespread collection or network of systems functioning together to achieve a common purpose”. Besides this definition, there are many more, but the interesting question is how to distinguish SoS from very large but monolithic complex systems. Neube gives characteristics to distinguish a SoS from a large monolithic system in [?], including the operational and managerial independence of their constituent systems and their evolutionary nature in which functions and purposes are added, removed, or modified as needed. SoS thus stand out because of their composed nature, their large scale, their decentralized control mechanism, their evolving environments, and their large number of stakeholders. These characteristics impose a challenge for requirements engineering, inter alia, with regard to scoping and adequate structuring of artefacts.

Wnuk et al. [?] report on an interview study at Siemens on large-scale requirements engineering, where they explore the challenges of increasing size and complexity in market-driven software development. Their interview study identifies, inter alia, the challenge of choosing adequate artefact structures for large scale requirements engineering. The paper at hand defines such a structure.

A current research project that investigates SoS is the ARAMiS (Automotive, Railway, and Avionics in Multicore Systems) project. ARAMiS is a BMBF-funded project which aims to create the technological basis to further improve security, traffic-efficiency, and comfort in the mobility domains automotive, avionic, and railway by utilizing multi-core technologies2. The ARAMiS project is composed of large systems at different levels of abstraction which are inherently cross-domain (automotive, railway, avionic), controlled decentrally, and evolving rapidly. Furthermore, the stakeholders that have an interest in the system are numerous, diverse, and belong to multiple domains.

Consequently, requirements engineering in ARAMiS is a challenge: the typical challenge of Requirements Engineering for Systems of Systems (RESS) with the additional issue of spanning across domains.

Problem: Requirements engineering for systems of systems faces extremely distributed requirements engineering activities that involve a multitude of stakeholders, while the surrounding SoS is not necessarily in the focus of single system developers working on small units within the SoS. This often results in isolated requirements engineering approaches which, in turn, lead to requirements that can hardly be integrated with the other units of the SoS in order to keep them consistent. Furthermore, problems arise with incomplete and/or redundant contents, consistency, and traceability.

Contribution: We propose a requirements engineering content model that serves as reference for requirements engineering, inter alia, with regard to scoping and adequate structuring of artefacts.

1The Federal Ministry of Education and Research
2http://www.projekt-aramis.de/
elicitation and documentation on the different levels of abstraction. This responds to the need for such a structure identified by Wnuk et al. [?]. Instantiated in a concrete artefact model, contents are documented in a traceable way while at the same time ensuring coverage of an agreed set of contents. The benefits are consistency within the requirements and eased communication between the stakeholders.

Outline: Section ?? provides background on related work and Section ?? on the project ARAMIS. Section ?? presents the requirements engineering content model. Section ?? describes the tool-supported artefact model and the instantiation on the system of systems level as well as in the domains of automotive and avionics. Section ?? concludes the paper with an outlook on future work.

II. RELATED WORK

The related work for the presented RE content model combines the following areas: Other approaches in engineering for systems of systems or for big systems in general, challenges for the SoS requirements engineering, and other RE content models and artefact models.

1) System of Systems Engineering: Ring and Madni [?] identify System of Systems Engineering (SoSE) as to extract value from existing assets and designs new assets to be more easily re-purposed and list key challenges and some opportunities for SoS. Thereby, they focus on sustaining SoS, because they consider managing the ongoing evolution of the SoS more challenging than initializing the SoS. Their SoSE infrastructure opportunities request an artefact model: “SoSE practitioners stand to benefit from the ability to create not only static models, both descriptive and prescriptive, but also dynamic models of a contemplated system, its behaviors and effects” [? , p. 5].

Loren [?] describes issues and challenges for SoSE in the US Air Force on a high level of abstraction. His conclusion is that an integrated set of RE processes could provide a framework for transformation of SoS-level capability statements to manageable, executable requirements.

2) SoS Requirements Engineering Challenges: Ncube [?] describes the high societal impact of SoS and characterizes the operational environments of SoS. On that basis, he identifies interoperation, including composition as special case, requirements engineering, and emergent behavior management as immediate key challenges for the requirements engineering community.

Konrad and Gall [?] report on challenges faced and lessons learned in requirements engineering for the development of large-scale systems at Siemens. Important lessons were to develop a well-structured feature list and a good understanding of customer requirements, to use specification approaches that scale, to separate requirements and design decisions, and to establish a traceability model with a measurement process. The identified lesson that is most relevant for the paper at hand is the need for effective documentation standards, which can be solved by an artefact model.

3) SoS Requirements Engineering Approaches: Requirements engineering approaches usually follow either an activity-based philosophy or an artefact-based philosophy as, for example, the approach presented in this paper. We group the related work according to these philosophies.

Activity-orientation: Savio et al. [?] present considerations for a requirements engineering process model and conclude that such a process model should include an internal, surface and external view. Their idea of a first coarse-grained sketch is a representational framework that is to be delivered as future work.

Avritzer et al. [?] used a system of systems process in their Global Studio Project. It was implemented in an extended workbench model. They do not focus on the artefact structures used throughout the project, but they state that the early definition of a formal specification of a common interface between components reduced the needs for formal requirements specification [? , p. 3].

Lewis et al. [?] propose a requirements engineering approach for SoS. They list the specific challenges (scale, multi-domain, varied operational context, decentralized control, rapidly evolving environments, etc.) and set respective goals in terms of the identification of capabilities and needs. Their approach combines top-down and bottom-up with the main steps of identifying SoS context, identifying SoS and individual system goals, understanding SoS interactions, identifying individual system capabilities and constraints, and analyzing the gap.

Artefact-orientation: Berenbach et al. [? , chap. 2] describe RE artifact modeling and name the key components to be a measurable reference model, a process tailoring approach, and respective process guidelines. Berenbach describes each of these elements and suggests practices for their elaboration.

Silva and Oliveira [?] propose a concept of meta-modeling to define an artefact layer and a process layer for a better organisation of software artefact authoring. They exemplarily illustrate their approach with a use case specification outline but do not provide a complete artefact model.

Méndez et al. [?] present fundamentals and lessons learned in requirements engineering from developing a meta model for artefact-orientation. In [? ], they report on a case study with a street traffic management business unit from Siemens on the application of an artefact-based requirements engineering approach.

The content model presented in this paper is built on our experience from previous research projects and industrial collaboration, in embedded systems as well as in business information systems. In the REMsES project3, Penzenstadler et al. [?] developed a requirements engineering guideline

3http://remses.org/
for software-intensive embedded systems that bases upon a requirements reference model. In the area of business information systems, the BISA approach by Mendez [?] provides a customisable model-based approach for business information systems’ analysis with artefact-based requirements engineering and its integration into a process framework.

Furthermore, templates like the Volère Requirements Specification Template⁴ can provide assistance. For domain-specific requirements modelling, oftentimes domain-specific languages (DSLs) are defined and used.

We did not find any related work on artefact-oriented approaches dedicated to develop systems of systems.

III. BACKGROUND: ARAMiS

One class of systems of systems with the additional challenge of integrating different system types are cyber-physical systems (CPS). CPS are integrations of computation and physical processes. Lee [?] defines CPS as embedded computers and networks that monitor and control physical processes, usually with feedback loops where physical processes affect computations and vice versa. This leads to complex functionality that spans a variety of application domains.

The German academy of technical sciences (acatech) has recently completed a study on the perspectives in CPS research, development, and application [?]. This study serves as scientific basis for the publicly funded research project ARAMiS: Automotive, Railway, and Avionics in Multicore Systems⁵.

The projects duration is 3 years, it has a total budget of 36.5 mio Euro, and it is partially funded by the German federal ministry of education and research. Partners are inter alia KIT, AUDI, BMW, Bosch, Continental, Daimler, Airbus, EADS, Diehl, Liebherr, Freescale, Infineon, Intel, SYSGO, Vector, Wind River, FHE ISE, and AISSEC, OFFIS e. V., TU Braunschweig, TU Kaiserslautern, TU München, Universität Paderborn, Universität Stuttgart, and ForTIS GmbH.

The main goal of ARAMiS is to provide for the technological basis for improving safety, efficiency, and comfort in the mobility domains of automotive, railway, and avionics by using multicore technology. The insights gained in the project build the indispensable foundation for the successful integration of embedded systems to cyber-physical systems.

The project is structured in 6 sub-projects: scenarios and requirements, continuous development method, system design, hardware, software, and demonstrators. Results that are common to all application domains are captured in the so-called Domain Common. Our research group has the academic sub-project lead in “scenarios and requirements” together with AUDI as industrial lead.

The structure and decomposition of the ARAMiS systems of systems, the CPS, is depicted in Fig. ?? This structure was elaborated in a workshop with the core partners of the sub-project requirements (where “core” partners is only due to the specific partner having assigned more effort to the sub-project than other partners). The discussion led to an agreement on the most adequate terms and a decomposition according to the partners’ respective experiences from previous research (e.g., the project SPES⁶) and lessons learned in earlier industrial projects.

The structure in Fig. ?? starts with the CPS levels that features scenarios like smart cities and smart mobility, continues on the domain levels, which are specific domains within automotive (e.g., intelligent infotainment) and avionics (e.g., autonomous flight), and further decomposes into the system level, for example, car2car communication.

Due to the nature of CPS and the high number of partners, the project faces typical challenges for performing requirements engineering for systems of systems: highly distributed development on different levels of abstraction with a diversity of collaborators on each level.

The next section presents the content model we developed to tackle these challenges.

IV. REQUIREMENTS ENGINEERING CONTENT MODEL

The challenge of a large number of partners collaborating in requirements engineering for systems of systems on different levels of abstraction can either be solved via a detailed process description that determines the small-grained steps to be taken, or via artefact-orientation. The main principle of

http://www.volere.co.uk/template.htm
http://video.sae.org/11214
http://spes2020.informatik.tu-muenchen.de/
artefact-orientation is to define what has to be accomplished (the work product) instead of how it has to be accomplished (the steps that have to be taken). As in Méndez et al. [7], an artefact is thereby defined by its content, structure, the concepts that describe the content, and its notation (syntax and semantics).

As artefact-orientation is rather independent of the surrounding development process, it serves well for distributed development with many stakeholders from different companies with varying development processes, since artefact-orientation restricts itself to defining the intermediate and final results of the development process and does not detail how exactly these results are to be elaborated. An artefact model can easily be integrated into existing processes by using already defined roles and interfaces, as shown in the case study by Méndez et al. [7].

The first step for defining an artefact model is to define the underlying RE content model. A content model is a first sketch of contents and the overall structure to be covered by an artefact model, while still abstracting from the notation. Berenbach et al. call such a content model an RE taxonomy [?, p. 24], but our project partners seemed to relate more to the term ‘content model’.

The content model was developed on the basis of experiences from previous research and industrial collaborations as well as lessons learned in these projects, e.g. [8], [9], and [10]. The first version was then proposed to the ARAMiS project partners and discussed and further tailored in two workshops. The lead question used to structure the discussion in the first workshop was about the information needs and requirements types as well as the relations of proposed content items in the model. The second workshop reviewed the updated version of the content model and discussed roles, and responsibilities, and process-related questions. In a third workshop, the release version of the RE content model was presented to the whole consortium of the ARAMiS project.

An overview of the resulting content model is depicted in Fig. 2. The model has four levels of abstraction: context level, system level, subsystem level, and technical architecture level. Requirements Engineering focuses on the upper two levels. The two lower abstraction levels are included to facilitate integration with design and to enable bottom-up propagation of technical constraints. The individual levels comprise the following descriptions of the system under development:

- **Context (Operational and Business):** The Environment is described independently of a concrete system, including the business environment that the system supports as well as the operational environment that (physically) surrounds the system to be developed.
- **System / Product Requirements:** On this level, the interaction between users and system is described within the problem domain (black box).
- **Subsystem / Logical Architecture:** On this level, the system is described as white box in terms of structure and behaviour while still not distinguishing between hardware or software.
- **Technical Architecture:** This level describes the realisation of the system in hardware and software.

According to Ferrari et al. [11], tighter integration of software architecture to requirements engineering is necessary across different subsystems due to the strong influence that architecture has on requirements engineering decisions. The lower two levels of our proposed model support such a tight integration by illustrating the relations already in the RE content model.

Each level features a number of content items that were reviewed by and discussed with requirements engineering experts from the project partners during the second workshop. The content items are denoted in ovals in Fig. 2. The depicted relations between the content items are not exhaustive but depict only a first initial input for derived content items that is described in the following.

- **Domain Model:** A system-independent description of the domain and the operational and business environment.
- Stakeholder: A description of individuals or organisations that have an interest in the project. Stakeholders are referenced from their respective Objectives, Goals, and Constraints.
- Objectives, Goals and Constraints: An objective is a major statement of intent to be achieved by a project, while a goal is a more concrete, “prescriptive statement of intent of one or more stakeholders” as defined by Lamsweerde [?]. Constraints are restrictions of any type. Objectives and goals are represented by the same concepts: we refine objectives into goals.
- System Vision: Domain elements, Stakeholders and Goals provide the information for the System Vision. It is an outline of the system’s capabilities and characteristics in its context — a vision of the system under development agreed upon by all stakeholders.
- Usage Model: The usage processes for the System Vision are detailed in the Usage model. It provides description of observable system behaviour and interaction from the viewpoint of the user (use cases, scenarios, interface requirements). The Usage Model is a major information source for the data requirements to be captured in the Data Model, the Functional Requirements, and the Quality Requirements.
- Data Model: The Data Model captures the coarse-grained information objects processed needed for the execution of system functions. The latter are derived from the Usage Model.
- Functional Requirements & Service Model: The Services are the single units of functionality that realise the behaviour specified in the Usage Model. Usage Model and Service Model are both views on black box behaviour from different perspectives (domain vs. technical).
- Quality Requirements: A description of properties and conditions for structure or behaviour of the system by use of measures. Quality requirements and constraints refer to a common quality model. There are several views for quality that are determined by the stakeholder who issues a quality requirement, for example, user, maintainer, or legislation. Quality requirements can thereby have several stakeholders.
- Product Constraints: The quality-independent restrictions on functional, logical, and technical architecture. These are derived from the Constraints on the Context level.
- Lifecycle Constraints: The restrictions and agreements concerning development process, release, integration, and maintenance. Equally to the Product Constraints, they are derived from the Constraints on the Context level. Product constraints and lifecycle constraints are based on the same concepts but have different stakeholders.
- Risk Report / List: The description of potential risks and proposed mitigation measures related to either specific requirements (derived from Product Constraints) or the project in general (derived from Process Constraints).
- Component Model: The structure and behaviour of the logical components that realise the functions defined in the Service Model. Thereby, a component can again be perceived as and treated like a system (for example, in SoS).
- Software Model: The model describes the packaging and communication of the software components to be implemented for the logical components of the Component Model.
- Deployment: The model provides the mapping of software components to hardware topology and the integration into the surrounding operational environment that is defined on the Context level.
- Topology: The topology outlines the hardware and communication infrastructure.

A description of how such a content model can be filled (independent of its realization in an artefact model) is available from Penzenstadler [?].

V. INSTANTIATION OF THE CONTENT MODEL

To use the content model, it first has to be instantiated as a concrete artefact model. Instantiation is performed by enriching the specific content items with concrete syntax, i.e., the artefact model defines what are the concrete artefacts of a content item and how they are documented. Then, the resulting artefact model can be used for software system development.

A. Artefact Model for ARAMiS

The instantiation of the content model in an artefact model was elaborated in a workshop with the core partners’ requirements experts and subsequent iterational releases for the tool support. The two major decisions taken at the initial workshop were to decide on the notation forms (syntax) for the content items and to select a tool that would support the chosen notation forms.

For the notation, we decided to use a combination of SysML and UML to define the items of the content model, since UML and SysML are both well understood across the industrial partners and used much in practice. More specifically, we created a self-contained UML profile, which uses concepts of, e.g., SysML, to create specific concepts and terminology for the artefact model and to add visual notations.

For each of the content items of the content model, we specified one or several diagram types and stereotypes.

7Full resolution of the picture available online: 
http://www4.in.tum.de/~eckharjo/RESS/ContentModelEA.png
Figure ?? shows parts of the UML profile which defines the main stereotypes and their meta-properties.

On the left side of Fig. ??, one can see the stereotypes for the three different types of requirements: Interfacerequirement, Functionallrequirement, and Qualityrequirement. These three types of requirement are specializations of the stereotype Abstractrequirement, which extends the SysML metaclass Requirement. The stereotypes Lifecycle-constraint and Productconstraint, as well as Goal and Constraint also extend the SysML metaclass Requirement by their generalization Abstractconstraint and Abstractgoal, respectively.

On the right side of Fig. ??, one can find the stereotype Usecase, which directly extends the UML metaclass Use-case and the stereotypes Actor and Stakeholder, which both extend the UML metaclass Actor via their generalization Abstractactor. The figure also depicts the relationships between the stereotypes. For example, the stereotype Usecase is associated with the stereotype Abstractgoal in order to connect corresponding goals or constraints with one specific use case.

Table ?? shows the mapping between all content items (listed in the left column) and their corresponding diagrams and stereotypes that are used in the artefact model. In addition to that, we introduced generalization and dependency associations that can be used to express relationships between stakeholders. Furthermore, we adapted the KAOs [?] goal-tree modelling approach by introducing influence, contribute, conflict, and constraint associations for the stereotypes Goal and Constraint.

Table 1

<table>
<thead>
<tr>
<th>Content Item</th>
<th>Diagram</th>
<th>Stereotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Model</td>
<td>SysML Internal Block Diagram</td>
<td>none</td>
</tr>
<tr>
<td>Stakeholder Model</td>
<td>UML Actor Diagram</td>
<td>Stakeholder</td>
</tr>
<tr>
<td>Goals &amp; Constraints</td>
<td>UML Class Diagram</td>
<td>Goal, Constraint</td>
</tr>
<tr>
<td>System Vision</td>
<td>none</td>
<td>None</td>
</tr>
<tr>
<td>Usage Model: Use Cases</td>
<td>Use Case Diagram</td>
<td>UseCase*</td>
</tr>
<tr>
<td>Usage Model: Interface</td>
<td>SysML Requirements Diagram</td>
<td>InterfaceRequirement</td>
</tr>
<tr>
<td>Data Model</td>
<td>UML Class Diagram</td>
<td>None</td>
</tr>
<tr>
<td>Functional Requirements</td>
<td>SysML Requirements Diagram</td>
<td>FunctionalRequirement</td>
</tr>
<tr>
<td>Quality Requirements</td>
<td>SysML Requirements Diagram</td>
<td>QualityRequirement</td>
</tr>
<tr>
<td>Product Constraints</td>
<td>SysML Requirements Diagram</td>
<td>ProductConstraint</td>
</tr>
<tr>
<td>Lifecycle Constraints</td>
<td>SysML Requirements Diagram</td>
<td>LifecycleConstraint</td>
</tr>
<tr>
<td>Risk List</td>
<td>SysML Requirements Diagram</td>
<td>Risk</td>
</tr>
</tbody>
</table>

*We used the Cockburn [?] template for use cases as meta-properties in the UseCase stereotype.

B. Application in Aramis

As shown in Fig. ??, the ARAMiS project consists of systems located at different levels of decomposition. Starting from the highest level, ARAMiS consists of the Smart City system at CPS Level 0, then the Smart Mobility and Smart Health systems at CPS Level 1, the different domain-specific systems at the domain level, and finally the concrete systems at the system level.

In order to apply the content model for the entire SoS, its instantiation needs to be applied to each system (at each level of decomposition) in an iterative manner. For a concrete SoS, as for example the ARAMiS project, this results in one instantiation of the artefact model at each of the systems. Furthermore, the iterative nature of the content model propagates bottom-up requirements to the upper levels, e.g., requirements that are specific to one of the systems in the system level. This leads to an integrated top-down as well as bottom-up approach.

In order to provide a project-wide tooling support for ARAMiS, we decided to use Enterprise Architect (EA)* for model-based requirements engineering and implemented the artefact model as a Model Driven Generation (MDG) Technologies plugin for EA*. The MDG plugin provides the ARAMiS specific diagram types and stereotypes as well as the structure of the artefacts.

In the following, we describe how we applied the artefact model on different levels of granularity, namely the SoS level and the domain level from Fig. ??.

C. Instantiation on CPS-Level

The CPS scenario [?] resides on the highest level of decomposition and conceives a future scenario for mobility spanning the automotive, railway and avionic domains, based on a generic Smart Mobility scenario described in the Agenda CPS study report [?]. In our derived and more specific scenario, Ms Weber, the main character, wants to spend the Christmas break with her two children at her parents in Paris and a Traffic Management System (TMS) is responsible for organizing the trip.

Figure ?? shows the basic path: First, she uses public transportation to pick up her children at school, then she drives in a car-sharing vehicle to the airport, and finally goes by plane to Paris. Furthermore, the scenario contains three alternative scenarios in which (i) the train is cancelled, (ii) the emission limit exceeds the limits, and (iii) the flight is delayed or cancelled. For the ARAMiS Project, we specified a model of the CPS scenario in EA using the MDG plugin presented here.

The CPS scenario creates the uppermost frame in the SoS which keeps the different domains together, and provides

---

*Available for download including slide tutorial at [http://www.sparxsystems.de/uml/neweditions/]
*Available for download including slide tutorial at [http://www4.in.tum.de/~eckharjo/RESS/ARAMiS-MDG-Plugin.zip]
points of contact for the different domains and for the actual demonstrators. Additionally, bottom-up requirements from the different domains can be propagated up to this level and included in this scenario.

D. Instantiation on Domain Level

The scenarios on the domain level derive from common content of the CPS scenario. In each domain, a set of scenarios is developed: intelligent infotainment, cooperative vehicle, virtual engineering, and zero impact car scenarios in the automotive domain and free flight and cabin management scenarios in the avionic domain.\textsuperscript{10}

Each scenario has been created in collaboration across multiple companies. For that, we had a number of tutorial meetings with some of the partners in order to familiarize them with the tool.

It is important to note, that the scenarios are still work in progress, and thus, the following information and diagrams may be incomplete. Nevertheless, as with the CPS scenario, EA has been used to specify a model of these systems. Figure ?? and ?? show selected parts of the EA model of the automotive scenarios. Figure ??\textsuperscript{11} shows parts of the goal tree; for example, one can see on the left side of the tree the refinement of the top level goal “Security/Safety”. Fig. ??\textsuperscript{12} shows an use case overview diagram which shows the use case “DB-II-01 Business Trip”, and corresponding included use cases. Additionally, two actors are included: the car occupant and one of its specializations, the driver. This diagram is useful to get an overview on specific use cases and to depict the relationships between the use cases as well as between the use cases and the corresponding actors.

From the domains, i.e. the respective stakeholders in the participating companies, information is fed back into the CPS level and the respective model description.

E. Discussion

The application of an artefact model eases communication, improves consistency, and provides traceability of the contents. These benefits of artefact-orientation are discussed in Mendez et al. [?]. In addition, we provide argumentation and indicative (preliminary) evidence for the respective benefits in the following:

\textsuperscript{11}Full resolution of picture online: http://www4.in.tum.de/\~eckharjo/RESS/Automotive-Goal.png

\textsuperscript{12}http://www4.in.tum.de/\~eckharjo/RESS/Automotive-UC.png
Figure 5. A part of the Objectives, Goals & Constraints content item for the automotive domain represented in EA using the presented MDG plugin.

Figure 6. A selected part of the use cases from the Usage Model content item for the automotive domain represented in EA using the presented MDG plugin.
**Communication:** Using one agreed content model and one agreed tool supports the interoperability and eases the communication and collaboration between the partners. The indicator that we have noticed since the workshops on the content model is that discussions, for example in the weekly conference calls in the project, are more to the point and require less extensive explanations of the current context.

Additionally, since the same content model is being used on different levels of decomposition, the organization and structure of the models is easy to understand by project partners who work on other levels of abstraction while using the same content items and notation.

**Consistency:** Artefact-orientation provides for commonly agreed work products with a single point of content. This avoids redundant information in different documents that otherwise would have to be kept consistent with (potentially high) manual effort. Instead, there is only one master (of content) from which deliverables in form of documents can be exported.

In addition to the single point of content, all content elements that refer to a common system model, i.e., starting on the subsystem level, are checked automatically for consistency in the tool. The consistency between elements on the upper levels has to be checked manually but it is eased by the content items referencing related content items. For example, if a use case changes in the Usage Model, the referenced system services in the Service Model have to be checked.

**Traceability:** The model relies on a meta model that is (partly) depicted in Fig. ?? The meta model defines the relations between the individual elements and, thereby, how they are traced. Furthermore, the tool support automatically provides the tracing as defined in the meta model by linking diagrams and model elements.

Since the model is used on the different levels of decomposition, the process of including and propagating bottom-up requirements is structured and reproducible.

**Acceptance across Domains:** One exceptional characteristic of SoS is the variety of stakeholders with different specializations, background, skills, and vocabulary (as one can see in the ARAMiS project), that are working together at one complex SoS. This creates an immense but not intended additional complexity.

The participation of the partners in the coordination process led to an acceptance of the content model across the different partner companies. Despite other methods or approaches that were not directly developed or adapted together with the partners, the resulting identification with the content model constitutes a substantial step to reduce this inherent additional complexity.

**VI. CONCLUSION AND FUTURE WORK**

This paper presents a domain-independent requirements engineering content model (depicted in Fig. ??) that can be applied on multiple levels in requirements engineering for SoS. It is concretised in an artefact model and instantiated as an Enterprise Architect MDG plugin. It is evaluated on the different ARAMiS granularity levels (depicted in Fig. ??) with cyber-physical system scenarios and derived, domain-specific scenarios from the automotive domain and the avionics domain by 30 industrial and academic partners.

The content model has led to common wording and efficient collaboration between the partners and across the domains, thereby showing first indicators of successfully approaching some of the challenges of requirements engineering for systems of systems.

**Future Work:** Currently, we are working on cross-cutting scenarios that tackle such aspects as evolution, maintenance, and sustainability. Furthermore, future work is to perform an evidentiary study on the evaluation that provides a proof of the improved collaboration by using the content model.

The communication performed with the different industrial partners in the ARAMiS project led to a common vocabulary for the whole project. This results in a common basis for discussions and collaboration. This coordination process can likely be supported by tailoring guidance for the content model.

Since there are partners involved from both the avionics and the automotive domain, certification is also an important issue. This will be addressed in future work.

**ACKNOWLEDGEMENTS**

This work was funded within the project ARAMiS by the German Federal Ministry for Education and Research with the funding IDs 01IS11035. The responsibility for the content remains with the authors. We would like to thank Daniel Méndez, Henning Femmer, and Wolfgang Schwitzer for feedback on earlier versions of this paper.