An Ontological Context Modeling Framework for Coping with the Dynamic Contexts of Cyber-physical Systems

Jennifer Brings[1], Marian Daun[1], Constantin Hildebrandt[2] and Sebastian Törsleff[2]
[1]*paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen, Essen, Germany*
[2]*Automation Technology Institute, Helmut-Schmidt-University, Hamburg, Germany*

Keywords: Cyber-physical Systems, Collaborative Systems, Context, Context Modeling, Dynamic Context.

Abstract: Cyber-physical systems are highly collaborative by nature. At runtime these systems collaborate with each other to achieve goals that a single system could not achieve on its own. For example, autonomous vehicles can dynamically form convoys at runtime to facilitate higher traffic throughput and a reduction in CO2 emissions. While the importance of context documentation and analysis in system development is well known, current model-based engineering approaches struggle with the size and complexity of cyber-physical systems' contexts. This is due to high variety and dynamicity of the contexts to be considered. For example, a convoy to be formed at runtime may consist of different numbers of participating vehicles. Additionally, it may face different neighboring, not partaking context systems (e.g., non-equipped vehicles, equipped but not participating vehicles) and situations (e.g., speed limits, road construction sites, emergency vehicles). This paper proposes a context ontology to cope with highly dynamic contexts of cyber-physical systems by explicitly differentiating between not only the system and its context but also between the cyber-physical system network the system participates in, as well as the system network's context.

1 INTRODUCTION

Cyber-physical systems (CPS) are closely integrated in their contexts. Not only by monitoring context measurements by means of sensors and influencing their context by means of actuators, but also with one another by means of direct communication devices or the future internet (Wolf, 2009). In doing so, cyber-physical systems form collaborating system networks to achieve common goals (Broy and Schmidt, 2014). For example, a network of transport robots can optimize costs and time used for transporting goods. This might involve single systems deviating from their local optima (e.g., taking a longer route) in order to contribute to the global optimization goal (e.g., minimizing the total distance travelled of all transport robots involved).

The context of a CPS is an important driver for the functionality and behavior the system exhibits. Furthermore, the existence of other context objects, such as barriers, people, or the number and position of production belts influences the actual behavior of the system. Hence, context aspects need to be taken into account during the engineering of cyber-physical systems. For example, the context is explicitly elicited during requirements engineering (Nuseibeh and Easterbrook, 2000), it is considered during safety analyses such as the FMEA (Stamatis, 2003), and the systems' architecture is designed to allow for context awareness at runtime (Whittle *et al.*, 2009).

Since model-based engineering can be seen as the standard approach to cope with today's challenges in cyber-physical system development (Broy, 2013), context models are heavily relied upon (Fouquet *et al.*, 2012). However, current approaches do not take into account the two-sided nature of the context, when it comes to modeling CPS networks. For CPS networks, parts of the context behave like a system on their own, namely the cyber-physical system under development and other cyber-physical systems in its context, which together form a system network to achieve common goals, create emergent functionality, and exhibit an aligned behavior. Hence, the model-based documentation must not only distinguish between the system and its context, but furthermore, between the system and the system network, as well as between the system network and its context. It is important to note that the system, the system network and their contexts are partly overlapping. This must be explicitly taken into

Brings, J., Daun, M., Hildebrandt, C. and Törsleff, S.
An Ontological Context Modeling Framework for Coping with the Dynamic Contexts of Cyber-physical Systems.
DOI: 10.5220/0006603403960403
In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018)*, pages 396-403
ISBN: 978-989-758-283-7

account when it comes, among others, to safety analyses. For example, each CPS must prevent unsafe behavior of the system network it is partaking in and unsafe context conditions identified from a system network perspective must be taken into account for each single system as well.

In this paper, we contribute an ontological context modeling framework for collaborating cyber-physical systems. The framework explicitly differentiates between the system under development, the system network under consideration (that the system under development belongs to) and their contexts. Furthermore, overlaps and mutual relations are identified and reflected in the modeling framework to enable advanced model-based analysis approaches to take advantage of this.

The paper is structured as follows. Section 2 briefly introduces the state of the art and previous work the ontological context modeling framework builds upon. Section 3 introduces the core principles of the ontological context framework and the framework itself. Finally, Section 4 concludes the paper.

2 RELATED WORK

In the software engineering field many approaches have been suggested for using contextual information as well as for explicitly documenting the system's context, which is especially important in requirements engineering (Gause, 2005). To this end, various approaches to document context information in requirements models have been suggested. For instance, goal-oriented approaches (e.g., (Yu, 1996; Ali, Dalpiaz and Giorgini, 2010)), refine top-level goals considering context knowledge elicited during requirements engineering. By doing so, goal fulfillment either depends on the system itself, on subsystems, or on entities in the context, such as external systems or human users. Common requirements engineering approaches typically take context information into account, but do so from the perspective of a single system, not considering the system network the system is part of or other systems attached to the system network.

In component-based development (e.g., (Cechich, Piattini and Vallecillo, 2003; Karsai *et al.*, 2003)), a system is refined across several layers of abstraction. Every subsystem can be considered a system in a shared context (i.e., the overall system), such approaches assume that only one overall development process is in place, which sequentially traverses the emerging subsystem tree and does not consider concurrent engineering processes. Ontology-based context modeling approaches, which have been proposed in the past (e.g., (Strang, Linnhoff-Popien and Frank, 2003)), focus on the documentation of context information of a single system under development. These approaches mostly rely on state-based behavior and do not take other types of context information into account, like static-structural or functional dependencies.

In order to document context information explicitly, some approaches (e.g., (Bergh and Coninx, 2006; Dhaussy *et al.*, 2009)) extend existing modeling languages such as the languages of the UML. Explicit documentation of context information is a prerequisite for various quality assurance and analysis approaches, such as model checking of development artifacts (e.g., (Dhaussy *et al.*, 2009)) as well as impact analysis of context changes (e.g., (Alfaro and Henzinger, 2001)).

A more generic view on the meaning of context in system development is given in context theory. For example, in (Gong, 2005), the distinction is made between context subject, i.e., the system, for which the context is being considered, and context objects, i.e., the entities that are within the context subject's context. By selecting the context subject, the scope is clearly defined: In principle, everything that is part of the context subject can be changed during development, whereas context objects are beyond the scope of development and cannot be changed. Context theoretic approaches such as (Jackson, 1995; Jin and Liu, 2006), place particular emphasis on the distinction between the system, the system's context, and the effect of the system onto its context. In this paper, we build on these approaches by extending them with the distinction between system and system network as well as their contexts and the resulting implications of this extension.

In previous work, we introduced an ontology for modeling the context of embedded systems (Daun, Tenbergen, *et al.*, 2016). Thereby, clearly distinguishing between the context of knowledge (Daun *et al.*, 2014), which places emphasis on identifying and documenting knowledge sources as needed in requirements engineering, and the operational context, which describes the context the system will be operating in at runtime. In this paper, we will focus on the operational context, for which the ontology provides the basis for various automated context analysis approaches (Daun *et al.*, 2015) and facilitates the concurrent engineering of interacting systems. (Daun, Brings, *et al.*, 2016). In this paper, we extend this context ontology to account for systems in the same system network and the dynamicity of such system networks.

3 CONTEXT FRAMEWORK

This chapter introduces the proposed context modeling framework for cyber-physical system networks. The framework is based on five principles: (1) the separation between system and context, (2) the consideration of different context subjects and their overlapping contexts (3), the differentiation between individual CPS and the CPS network, (4) the differentiation between different types of context objects, and (5) the dynamic nature of the context to be considered. A detailed description of these principles is given in Sections 3.1-3.5. Section 3.6 introduces the resulting context ontology.

3.1 Principle 1: Separation between System and Context

In software engineering the dividing line between system and context is traditionally drawn between what can be changed and what cannot be changed during development (e.g., (Nature Team, 1996)). While the system can be changed as needed, the context comprises all objects that are of relevance to the system and its development, but cannot be influenced during development and are thus seen as given. For example, during the development of an automotive traffic sign assistant, the object recognition functionality can be implemented as desired. The street signs to be recognized, however, cannot be changed or influenced, but they do have an impact on the object recognition functionality implemented in the system.

The system and its context are separated by the system boundary.

Figure 1 illustrates the relationship between the system, its context and the irrelevant environment. Everything within the system boundary is part of the system and subject to the development process, while everything outside is considered as given. Not everything outside the system boundary, however, is of relevance for the system and its development process. A car's engine, for example, is of no relevance to the aforementioned traffic sign assistant and thus not part of the traffic sign assistance's context but part of the irrelevant environment.

While the context also includes aspects that mainly influence system development and not the system's runtime behavior (e.g., road traffic licensing regulations), this paper focuses on the operational context, i.e., the part of the context that the system interacts with at runtime.

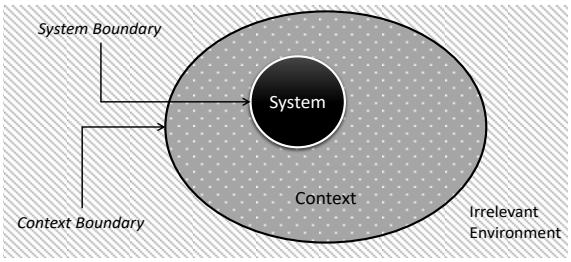


Figure 1: Context.

3.2 Principle 2: Consideration of Different Context Subjects and Their Overlapping Contexts

To cope with the complexity of modern systems, systems engineering frameworks utilize abstraction layers that allow for decomposing systems into sub-systems (e.g., (Böhm *et al.*, 2016)). Figure 2 illustrates this in a simplified fashion for the traffic sign assistant. On the second abstraction layer, the system is decomposed into the three components *camera*, *electronic control unit (ECU)*, and *user interface*. The ECU is further decomposed into an *object recognition* component and a *system management* component on the third abstraction layer.

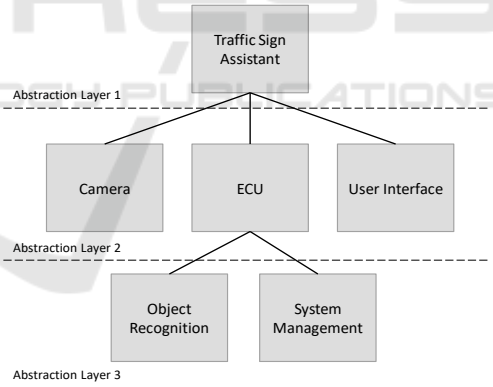


Figure 2: Decomposition of a Traffic Assistant.

From the traffic sign assistant's point of view all three components are part of the system. From the *ECU*'s point of view, however, the *camera* and the *user interface* are part of its context. Similarly, from the camera's point of view, the ECU and the user interface are part of the camera's context and both the ECU and the camera are in the user interface's context. This relation between the different systems and their context is illustrated in Figure 3. As can be seen in Figure 3, the distinction between system and context depends on the development subject and, hence, must be seen as variable.

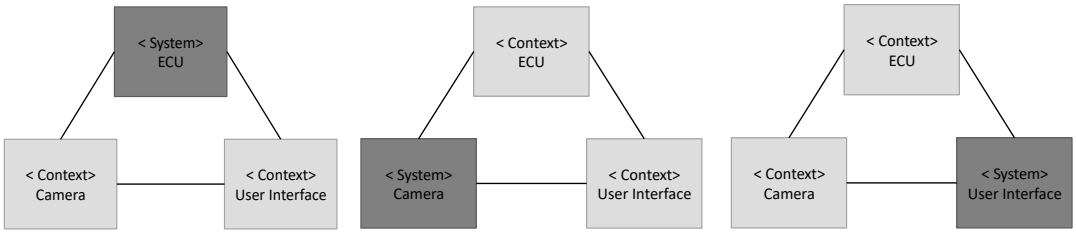


Figure 3: Exemplary mutual relationships between different CPS and their context objects.

For example, during the engineering of the traffic assistant on a high-level of granularity all three subsystems, i.e., ECU, camera and user interface, must be considered as part of the system. In contrast, on a more detailed level, the subsystems will be engineered within different engineering paths. In consequence, the user interface and camera must be viewed as context when engineering the ECU; ECU and user interface are part of the context of the camera; and ECU and camera are context objects for the user interface. In context theory, the system, subsystem, function, software, or whatever the context is defined for, is typically referred to as the context subject.

3.3 Principle 3: Differentiation between System and System Network

CPS form networks to achieve a common goal. For instance, a network of transport robots can negotiate an optimized strategy for transporting goods from A to B. To this end, each robot in the network adjusts its behavior accordingly, which may require it to select a suboptimal route and load for itself. Figure 4 (1) illustrates how a network of transport robots moves goods from A to B.

Considering a system network of autonomous transport robots as the context subject, the context objects comprise goods to be transported, the goods' current positions and destinations as well as obstacles in the room. An important characteristic of system networks that consist of CPS is that they are usually not designed as a whole, but rather piece by piece, i.e., each system separately without explicitly defining all possible system networks which it can be part of.

Therefore, it is reasonable for a single transport robot (e.g., R2) to be considered as the context subject. This, in contrast, leads to the other robots being context objects. In other words, from the point of view of a single robot the context comprises the other robots within the system network as well as the context objects that are outside the system network (goods, obstacles etc.). Both points of view are illustrated in Figure 4 (2) and Figure 4 (3) respectively.

From the point of view of a single CPS, the context consists of the system network as well as other objects outside the system network. In the transport robot example, there might not be the one transport robot as context object but rather different robots of different types. For instance, *R2* and *R3* might be of the same type, while the other six robots are from four different manufacturers.

As can be seen for CPS networks, it can be differentiated between the CPS and its context and the system network and its context. Table 1 summarizes the two different manifestations of context subjects and context objects that are relevant to CPS.

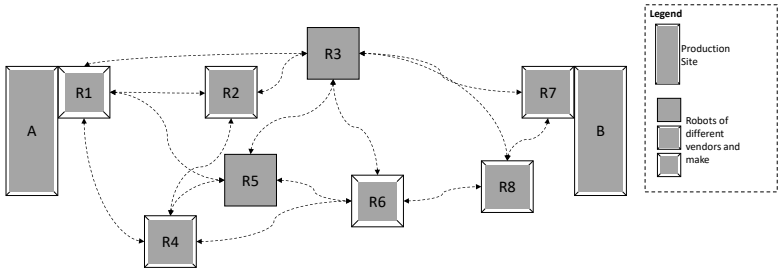
Table 1: Different Definitions of Context Subject and Context Objects for System Networks.

	System network perspective	**CPS perspective**
Context subject	System network	CPS within a system network
Context objects	Relevant objects outside the system network	Other CPS in the system network and relevant objects outside the system network

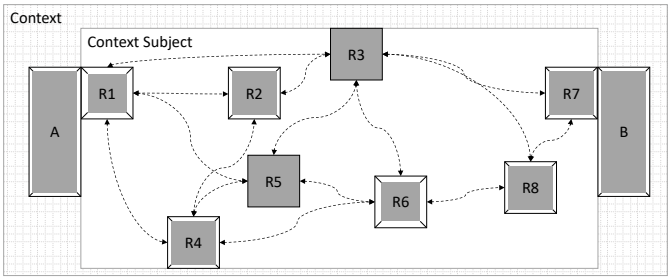
Treating collaborative systems in the system network as context objects, however, ignores the fact that unlike context objects in the conventional sense, these other systems often are not predefined at design time. In fact, many system networks will consist only in part of systems existent at design time.

3.4 Principle 4: Differentiation between Collaborative Context Objects and Non-Collaborative Context Objects

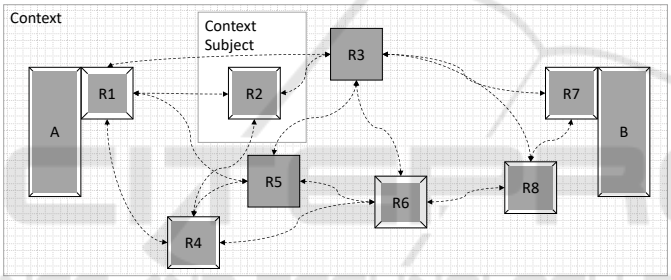
Systems collaborating in a system network emergently create some kind of overall functionality to achieve super goals that the individual systems cannot achieve on their own. Hence, there is a significant difference between context objects that



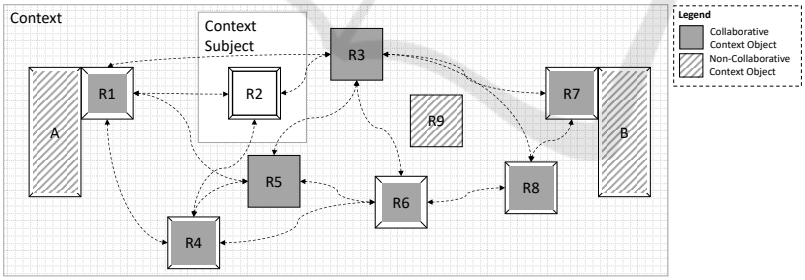
(1) CPS Network of Transport Robots



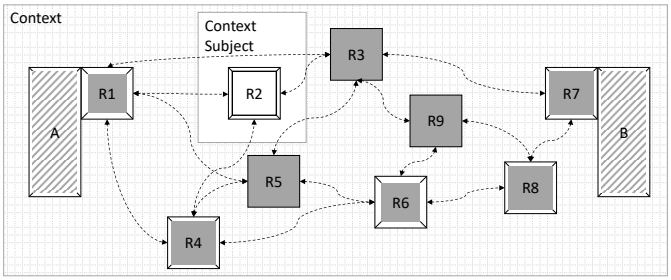
(2) CPS Network of Transport Robots as the Context Subject



(3) A Single CPS (Transport Robot) as the Context Subject



(4) Illustration of Collaborative and Non-Collaborative Context Objects



(5) Illustration of Dynamicity (R9 changed from Non-Collaborative Context Object to Collaborative Context Object

Figure 4: Illustration of a System Network of Transport Robots.

collaborate with the system under development (i.e., the context subject) to achieve such a super goal and context objects that do not collaborate.

Therefore, there is a need to distinguish between different context object classes. We choose to label them as *Collaborative Context Objects* and *Non-Collaborative Context Objects*. Figure 4 (4) illustrates the separation of collaborative and non-collaborative context objects for the transportation robot example.

Collaborative and non-collaborative context objects both belong to the relevant context. Since CPS provide their functionalities to a system network, the system network can achieve goals that a single system would not be able to achieve on its own. Therefore, objects that cooperate actively in achieving a goal through providing their functionalities belong to the collaborative context. For instance, the collaborative context of robot R2 contains all CPS that R2 collaborates with in order to achieve a common goal, i.e., transporting a good from A to B, optimized regarding time and cost.

All objects in the collaborative context of a system under development (i.e., the context subject) are able to actively communicate certain information (e.g., states, properties, parameters) about themselves that are necessary for evaluating how to achieve this goal. On the other hand, non-collaborative context objects participate only passively in achieving the system network's goal. As long as a transported good does not engage in a negotiation process with the robots (e.g., negotiation of transportation price and duration), it remains a non-collaborative context object.

3.5 Principle 5: Dynamicity of CPS Networks

CPS networks change at runtime (Broy, 2012). Therefore, CPS that are part of such a network have to cope with a dynamic context. Considering our running example, new robots might be introduced to the system network over time, or an individual robot might receive an upgrade that enables it to carry higher loads. In principle, whether a given context is dynamic depends on the time-span considered, respectively the observation horizon. Coming back to the transport robots, robot R2 might be considered the context subject. If the observation horizon is chosen to be infinitely short, there will be no change in the collaborative context as well as in the non-collaborative context of robot R2. If instead the observation horizon is chosen to be longer, for the context subject R2, several changes are possible:

New objects may enter the relevant context. This can be collaborative objects, e.g., a new robot is introduced to the fleet (see Figure 4 (5)), or non-collaborative objects, e.g., a new good has to be transported. Similarly, context objects may leave the context and become part of the irrelevant environment. Again, these can be collaborative context objects, e.g., a robot leaves the fleet because it is not working profitably any more, or non-collaborative context objects, e.g., a transported good reached its final destination. Furthermore, objects may change their context class, e.g., a transported good is equipped with software that enables it to participate in a price negotiation with the robots, resulting in a change of the context objects class from non-collaborative to collaborative object.

3.6 Context Ontology

The context ontology shown in Figure 5 is based on the five principles introduced in sections 3.1-3.5 and illustrates the different concepts and their relationships. The *environment* is split into the *irrelevant environment* and the *context*, which are separated by the *context boundary*. The context itself is comprised of various *context objects* (e.g., the traffic sign, the precipitation) and separated from the context subject (e.g., the traffic sign assistant) by the *subject boundary*. The context subject can be a *system network*, an individual *system*, a *subsystem*, *software*, or *hardware*. The context ontology further distinguishes between two types of context objects; *non-collaborative context objects* (nCCO) and *collaborative context objects* (CCO). The nCCOs do not participate in a collaboration with the context subject. This could be a traffic sign, which does not communicate with the traffic sign assistant, but is nevertheless part of the *context*. A CCO would be a traffic light, which is able to communicate with the *context subject* (i.e., the car's traffic sign assistant) in order to change from red to green when the car is approaching. The dynamicity (*Dynamic*) of the *context objects* relies on the chosen *observation horizon*, which can be illustrated by two extremes. Having an infinitesimal short *observation horizon*, there would not be any change in any *context object* at all. Having an infinite long *observation horizon*, there can be many changes. All the described concepts are generic and not tied to a specific domain. As part of our future research we will develop domain-specific extensions for these concepts.

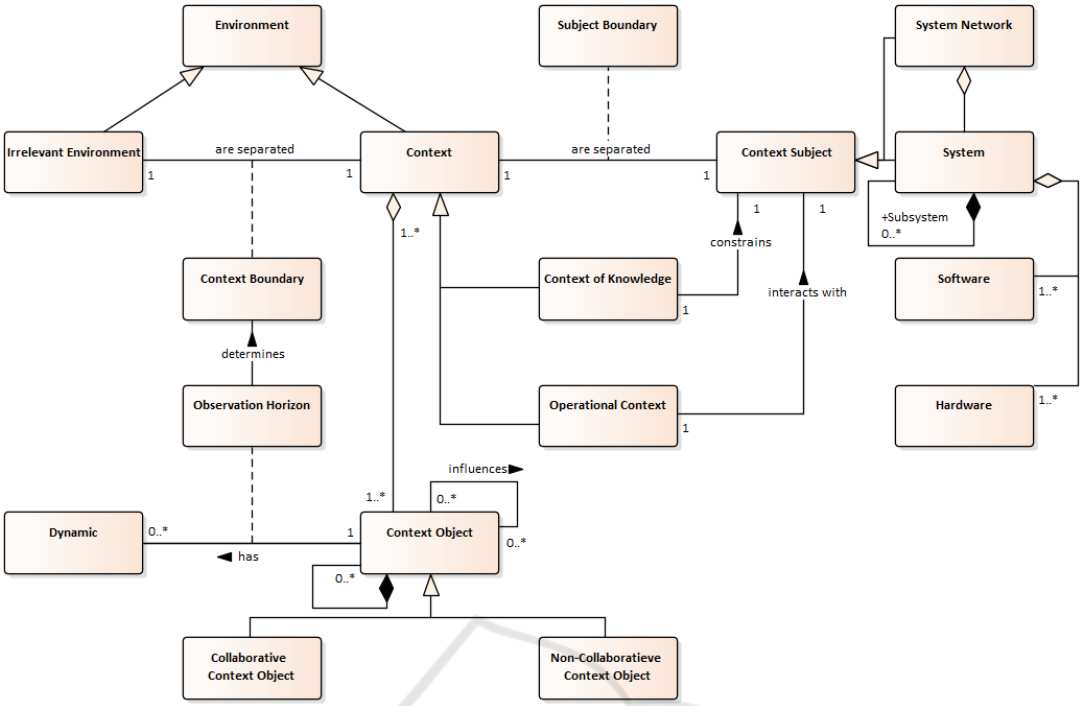


Figure 5: Context Ontology.

4 CONCLUSIONS

In this paper, we discussed the need to not only document and analyze the context of a CPS under development, but also the context of the collaborative system network the individual CPS takes part in. To this end, we presented a context ontology, which not only distinguishes between system and context, but also takes mutual relations between different CPS, the differentiation between individual systems and system network, the distinction between collaborative and non-collaborative objects, as well as the dynamicity of the context for collaborative CPS into account.

First evaluation results in industry are promising. However, future work will have to deal with a thorough investigation of the proposed methodological context framework. Furthermore, future work will deal with the instantiation of the context modeling framework for specific purposes, such as behavioral modeling, documenting the logical architecture of CPS networks, applying model verification techniques.

ACKNOWLEDGEMENTS

This research has partly been funded by the German federal ministry for education and research under grant no. 01IS16043U and grant no. 01IS16043V.

REFERENCES

Alfaro, L. de and Henzinger, T. A. (2001) 'Interface automata', in Tjoa, A. M. and Gruhn, V. (eds) *Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering 2001, Vienna, Austria, September 10-14, 2001*. ACM, pp. 109–120. doi: 10.1145/503209. 503226.

Ali, R., Dalpiaz, F. and Giorgini, P. (2010) 'A goal-based framework for contextual requirements modeling and analysis', *Requirements Engineering*, 15(4), pp. 439–458. doi: 10.1007/s00766-010-0110-z.

Bergh, J. V. den and Coninx, K. (2006) 'CUP 2.0: High-Level Modeling of Context-Sensitive Interactive Applications', in Nierstrasz, O. et al. (eds) *Model Driven Engineering Languages and Systems*. Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 140–154. doi: 10.1007/11880240_11.

Böhm, W. *et al.* (2016) 'SPES XT Modeling Framework', in Pohl, K. et al. (eds) *Advanced Model-Based Engineering of Embedded Systems*. Springer International Publishing, pp. 29–42. doi: 10.1007/978-3-319-48003-9_3.

Broy, M. (2012) 'Engineering Cyber-Physical Systems: Challenges and Foundations', in Aiguier, M. et al. (eds) *Complex Systems Design & Management, Proceedings of the Third International Conference on Complex Systems Design & Management CSD&M 2012, Paris, France, December 12-14, 2012*. Springer, pp. 1–13. doi: 10.1007/978-3-642-34404-6_1.

Broy, M. (2013) 'Challenges in modeling cyber-physical systems', in Abdelzaher, T. F., Römer, K., and Rajkumar, R. (eds) *The 12th International Conference on Information Processing in Sensor Networks (co-located with CPS Week 2013), IPSN 2013, Philadelphia, PA, USA, April 8-11, 2013*. ACM, pp. 5–6. doi: 10.1145/2461381.2461385.

Broy, M. and Schmidt, A. (2014) 'Challenges in Engineering Cyber-Physical Systems', *Computer*, 47(2), pp. 70–72. doi: 10.1109/MC.2014.30.

Cechich, A., Piattini, M. and Vallecillo, A. (2003) 'Assessing Component-Based Systems', in Component-Based Software Quality. Springer Berlin Heidelberg (Lecture Notes in Computer Science, 2693), pp. 1–20. doi: 10.1007/978-3-540-45064-1_1.

Daun, M. *et al.* (2014) 'On the Model-based Documentation of Knowledge Sources in the Engineering of Embedded Systems', in Schmid, K. et al. (eds) *Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering 2014, 25.-26. Februar 2014 in Kiel, Deutschland*. CEUR-WS.org (CEUR Workshop Proceedings), pp. 67–76.

Daun, M. *et al.* (2015) 'Documenting Assumptions About the Operational Context of Long-Living Collaborative Embedded Systems', in Zimmermann, W. et al. (eds) *Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering 2015, Dresden, Germany, 17.-18. März 2015*. CEUR-WS.org (CEUR Workshop Proceedings), pp. 115–117.

Daun, M., Brings, J., *et al.* (2016) 'Fostering concurrent engineering of cyber-physical systems a proposal for an ontological context framework', in *2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC). 2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC)*, pp. 5–10. doi: 10.1109/EITEC.2016.7503689.

Daun, M., Tenbergen, B., *et al.* (2016) 'SPES XT Context Modeling Framework', in Pohl, K. et al. (eds) *Advanced Model-Based Engineering of Embedded Systems*. Springer International Publishing, pp. 43–57. doi: 10.1007/978-3-319-48003-9_4.

Dhaussy, P. et al. (2009) 'Evaluating Context Descriptions and Property Definition Patterns for Software Formal Validation', in Schürr, A. and Selic, B. (eds) Model Driven Engineering Languages and Systems. *Springer Berlin Heidelberg* (Lecture Notes in Computer Science, 5795), pp. 438–452. doi: 10.1007/978-3-642-04425-0_34.

Fouquet, F. *et al.* (2012) 'A Dynamic Component Model for Cyber Physical Systems', in *Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering*. New York, NY, USA: ACM (CBSE '12), pp. 135–144. doi: 10.1145/2304736.2304759.

Gause, D. C. (2005) 'Why context matters - and what can we do about it?', *IEEE Software*, 22(5), pp. 13–15. doi: 10.1109/MS.2005.143.

Gong, L. (2005) 'Contextual modeling and applications', in *2005 IEEE International Conference on Systems, Man and Cybernetics*. *2005 IEEE International Conference on Systems, Man and Cybernetics*, p. 381–386 Vol. 1. doi: 10.1109/ICSMC.2005.1571176.

Jackson, M. (1995) 'The World and the Machine', in *Proceedings of the 17th International Conference on Software Engineering*. New York, NY, USA: ACM (ICSE '95), pp. 283–292. doi: 10.1145/225014.225041.

Jin, Z. and Liu, L. (2006) 'Towards Automatic Problem Decomposition: An Ontology-based Approach', in *Proceedings of the 2006 International Workshop on Advances and Applications of Problem Frames*. New York, NY, USA: ACM (IWAAPF '06), pp. 41–48. doi: 10.1145/1138670.1138678.

Karsai, G. *et al.* (2003) 'Model-integrated development of embedded software', *Proceedings of the IEEE*, 91(1), pp. 145–164. doi: 10.1109/JPROC.2002.805824.

Nature Team (1996) 'Defining visions in context: Models, processes and tools for requirements engineering', *Information Systems*, 21(6), pp. 515–547. doi: 10.1016/0306-4379(96)00026-9.

Nuseibeh, B. and Easterbrook, S. (2000) 'Requirements Engineering: A Roadmap', in *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM (ICSE '00), pp. 35–46. doi: 10.1145/336512.336523.

Stamatis, D. H. (2003) *Failure Mode and Effect Analysis: Fmea from Theory to Execution*. 2 Rev Exp. Milwaukee, Wisc: American Society for Quality Press.

Strang, T., Linnhoff-Popien, C. and Frank, K. (2003) 'CoOL: A Context Ontology Language to Enable Contextual Interoperability', in Stefani, J.-B., Demeure, I., and Hagimont, D. (eds) *Distributed Applications and Interoperable Systems*. Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 236–247. doi: 10.1007/978-3-540-40010-3_21.

Whittle, J. *et al.* (2009) 'RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems', in *2009 17th IEEE International Requirements Engineering Conference*. *2009 17th IEEE International Requirements Engineering Conference*, pp. 79–88. doi: 10.1109/RE.2009.36.

Wolf, W. (2009) 'Cyber-physical Systems', *Computer*, 42(3), pp. 88–89. doi: 10.1109/MC.2009.81.

Yu, E. S.-K. (1996) *Modelling Strategic Relationships for Process Reengineering*. University of Toronto.