Towards Context Modeling for Dynamic Collaborative Embedded Systems in Open Context

Ana Petrovska, Florian Grigoleit
Fakultät für Informatik, Technische Universität München
ana.petrovska@tum.de, florian.grigoleit@tum.de

Abstract

Collaborative embedded systems (CESs) are dynamic systems that integrate control, communication, and computational components and provide functions jointly as a group. Deployed in dynamic environments they often interact with a changing set of context objects and therefore, frequently change their system composition at runtime. To operate correctly, collaborative systems must self-adapt to changing context influences. Such adaptation depends on a common context model and context awareness of the collaborative group. We present an approach for context modeling, which allows context- and system awareness in collaborative systems. A system based on this approach retrieves relevant data from available information sources and aggregates them into a model describing the state of a system and its environment with a set of qualitative context attributes. The aggregated context model serves as a basis for the adaptation of the collaborating system. Problem and approach are illustrated with the use case of collaborative transport robots.

1 Introduction

Dynamic systems react to their environment. For collaborative embedded systems (CES), this includes interacting with other systems and reacting to influences from the environment. Such influences can impact the functionality of the system under consideration, e.g. obstacles or environmental conditions can prohibit activities such as transporting goods. In dynamic CESs, a system is defined as a set of embedded systems which collaborate to jointly achieve a task or provide a function. Dynamic means that both the system's members as well as their behavior (mode) can change at runtime. For example, if one member of a fleet of robots encounters an obstacle on its path and computes an alternative route, it can delegate its task to other members of the fleet or the system control. That means that the CES providing the function adapts to the state of the context. Such an adaption or (re-)configuration of a system depends on the required functionality, available assets, and the system's context situation.

We propose an approach for context modeling for collaborative embedded systems to enable self-adaptation based on situation and context awareness. To this end, we provide a qualitative representation, [Forbus, 1997], of context and context situations. We define a context situation as an aggregation of the state of the system, its context influences, and its context objects. In a dynamic operating environment, a system must adapt or reconfigure itself according to the given influences. Our approach extends a modeling technique for self-adaptive systems [Weiss *et al.*, 2013]: each single system within the collaborating system(s), called a collaborating component, possesses a qualitative model of its context situation. As the robots provide the function jointly, all members of the CES have to share their context models. Therefore, the context models within the collaborative system are distributed among the the collaborating components, allowing all collaborating components to assess the relevance of the various context situations for themselves. For example, a group of robots would share information about their individual surroundings, such as obstacles, blocked paths, or humans in the work area. This allows the members of the collaborative system to preemptively act on risks and, if necessary, to adapt the planning of their activities. The qualitative character of the models reduces the situation description to relevant aspects and allows an effective exchange of knowledge within the collaborative systems.

In Section 2, we present a use case from the CrESt (Collaborative Embedded Systems) project [cre, 2017a], followed by background overview in Section 3 and an analysis of the described problem summary of the proposed approach in Section 4. The paper continues with related work on context modeling and context awareness in Section 5. Lastly, we conclude with an outlook on the next steps in Section 6.

2 Use case: Transport Robots

Autonomous transport robots (proANT) are vehicles that carry loads in manufacturing and storage sites. Robots have the ability to collaborate together towards realizing a common goal. In order to achieve this, they can form fleets and a typical fleet consists of 4 to 20 robots, carrying between 50-200 kg load each. ProANTs are vehicles, which navigate and operate autonomously, except for activities they have to perform when no alternative route can be found. They do the path planning on their own, including receiving the in-

Figure 1: The proANT transport robots

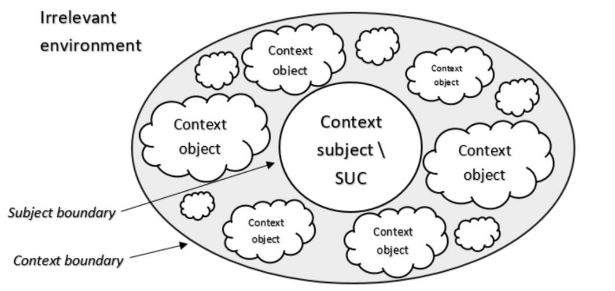


Figure 2: System and its context in SPES2020

formation of the pick-up and drop-off locations of the goods that they have to transfer, which are administered by a central management unit. Navigation is based on local maps, which are pre-installed on every vehicle in the fleet. When a robot receives a request to transport a good from point A to point B, it autonomously calculates the optimal path. If a robot encounters an unknown obstacle on this path, e.g. a pallet the vehicle cannot simply circle around, it comes up with an alternative route. When all alternative paths are being exhausted and the robot fails to execute an assigned transportation, this is reported to the central management unit [cre, 2017b].

Transport robots are usually deployed in factories, or storage sites where they can form fleets with other robots from the same type. These sites are additionally equipped with machinery, work spaces, and specialized departments that can be considered as static units of the environment.

To increase the autonomy of the system and therefore improve the overall performance with enlarging the efficiency of task handling, a collaborative approach for fleets of proANT robots is proposed. A CES is a group of individual robots that collaborate to achieve a common, *global* goal. For example, a fleet of transport robots can negotiate and optimize a strategy for transporting goods from point A to point B. However, it is important to emphasize that achieving an optimal strategy for the whole group, sometimes might result in non-optimal routes for individual robots. In the proposed collaborative approach, the central management unit will be omitted and the decision making will be conducted in a distributed manner, where each robot will decide at run-time which jobs to accept. One of the biggest challenges emerging from the collaboration of (different) CES is sharing relevant context data within the fleet, and our aggregated context model addresses this issue.

3 Background

The advance of dynamic systems requires a shift in the understanding of basic concepts and definitions in software engineering. Traditionally, software engineering defines a system as a collection of components, organized to implement or provide a specific function or set of functions [Committee and others, 1990]. A *system* is separated from its *environment* and other systems by a system boundary. The system's environment is divided between the *context* and the *irrelevant environment*, which is part of the environment with no significant influence on the system. Context is everything that is relevant to a system, but remains external to its range of action [Philippe *et al.*, 2013]. Separating the system and its context means distinguishing between changeable and unchangeable variables [Pohl, 2010]. Consequently, the context consists of all objects relevant to the system, but cannot be influenced by developers.

SPES 2020 methodology, [Daun *et al.*, 2016], [Pohl *et al.*, 2016] refers to the system, subsystem, function, software or the hardware, for which a certain context is defined as the *context subject*. The context subject additionally can be called system under consideration (SUC). As shown on Figure 2, the context consists of *context objects* that are separated from the context subject by the subject boundary. Additionally, in the SPES methodology, the context is divided into the context of knowledge and the operational context. Context of knowledge contains all sources of information relevant to system development, whereas the operational context consisting of all context objects that the SUC interacts with during runtime. The dynamic nature of CES in open context impacts only the operational context, on which, we will limit this paper.

In [Weiss *et al.*, 2013], authors present a qualitative context modeling approach to support the dynamic configuration of automotive functions. The main goal of the approach is to reduce the energy consumption by activating system functions only when the changing context requires it. For the approach, the paper proposes a context model that abstracts low-level contextual information and subsequently aggregates it. The resulting set of context attributes constitutes a qualitative representation of a system's context situation. The modeling technique is based on principles from qualitative reasoning, [Williams and de Kleer, 1991], [Forbus, 1997]. The model is designed to represent stable dynamic contexts, thus changes in the context situation that occur at runtime but change slowly. i.e. at a rate at which the adaptation of system functionality is helpful and feasible. That means for example, factory conditions or changing different system's situations are part of the model, while rapidly changing influences, such as surpassing robots, are not. The model was designed to initiate system reconfigurations in order to minimize the system's power consumption. In our work, we utilize the concept for CESs by integrating it in an extended version of the SPES 2020 context model and generalizing the modeling approach for multiple applications. Instead of considering single cars, we are building a variant specialized for collaborative embedded systems. Also, instead of merely providing context situations for reducing energy consumption, we aim at providing a comprehensive context model for various adaptation purposes, like fault tolerance or risk avoidance.

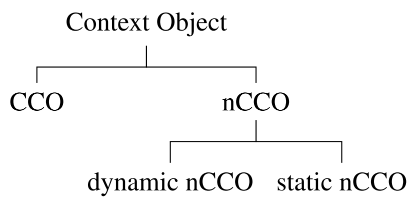
4 A context modeling approach for self-adaptation in dynamic configurations

4.1 Forming collaborative embedded systems and their context

Collaborative embedded systems in open context must react to ongoing, constant changes in their context. Whether a reaction involves reconfiguration of the CES or merely a reparameterization depends on the scale and impact of the context changes. Therefore, the CES is required to be aware of (parts of) its context, to assess the type and impact of the context change, and finally, to correspond correctly to the change.

Collaborative systems cannot be considered and treated as conventional context objects, because they are being rarely defined at design-time; on the contrary they are being formed and defined dynamically at run-time. A collaborative embedded system fulfills a *global* goal, which an individual system of the CES' set of collaborating components cannot fulfill alone.

Initially, before the CES is even established, context objects can be divided between objects that could collaborate with the context subject or SUC and accordingly form a CES, and context objects that do not collaborate. Consequently, we need to distinguish between two different context object classes: *Collaborative Context Objects (CCO)* and *non-Collaborative Context Objects (nCCO)* [cre, 2017b]. Figure 3 illustrates one possible separation of context objects as CCOs and nCCOs. Furthermore, as shown below, nCCOs can be divided between *dynamic nCCOs* and *static nCCOs*. CCOs and dynamic nCCOs are objects of the same class. Static nCCOs are instances of different classes and can never be constituents of CES. They impact the CES in a non-collaborative, passive manner.



CCOs and nCCOs, in particular dynamic nCCOs, differ for different system subjects. If a concrete context object is a nCCO for a specific context subject at time t, it does not mean that it cannot be a CCO for another context subject in a different collaborative system at the same time t, or eventually become a CCO for the initial context object in definite time $t + x$ in future. On the other hand, there might be some dynamic nCCOs, including all static nCCOs, which at a specific time are not collaborating components in any of the collaborative systems, and they only participate passively in achieving the *global* goals for different collaborative embedded systems. For example, in the use case, described in Section 2, if a transported good does not participate in the negotiation process with robots, then they only provide static information, and as long as the system does not provide a specific functionality that may change this static information, transported goods remain (static) nCCO.

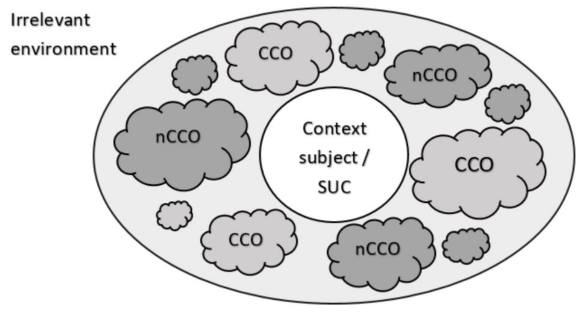


Figure 3: System and its context in CrESt

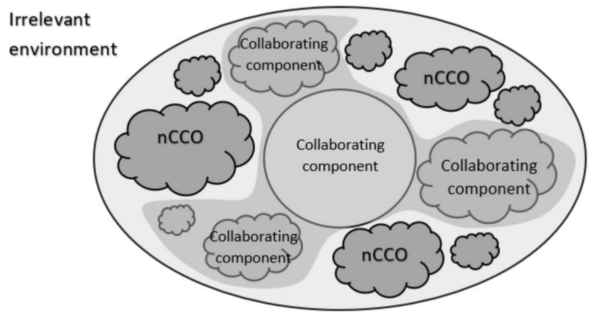


Figure 4: Forming Collaborative Embedded System

Once the CES -the context subject and CCOs- is being created as shown in Figure 4, all the members of that collaborative system become collaborating components - equal participants and contributors in the CES. All collaborating components are able to communicate and share certain information, like system states, properties, and parameters, about themselves that are necessary for achieving the *global* goal. Furthermore, at a particular time a specific collaborating component can be part of only *one* CES.

Additionally, few independent CES can form a *network* of collaborative systems, but the behavior of these networks is being excluded from the scope of this work.

4.2 Modeling context situations for collaborating embedded systems

We propose a modeling concept that merges context modeling for collaborating systems operating in open context [cre, 2017b] with a generalized version of the context modeling approach from [Weiss *et al.*, 2013]. The purpose of this modeling concept is to provide input to mechanisms for system reconfiguration or self-adaptation with a sufficiently detailed, but still small and concise context model, allowing the mechanisms to adapt the system accordingly. Although we use qualitative reasoning to represent and to reason about context, the concept is designed to support further representation and reasoning techniques. This includes, but is not limited to machine learning and pattern recognition techniques for determining context attributes and multi-agent approaches for co-ordinating the collaborative groups. The proposed context model consists of a system model for collaborating systems in a dynamic context comprising context objects and context attributes, described in [Weiss *et al.*, 2013]. Context attributes represent the state and situation of the system under consideration (SUC). Figure 5 depicts the main idea. Here, a collaborative system composed of three different collaborating components is shown. Collaborating Component 1 senses an obstacle in the direction of its moving and notifies the rest of the collaborating components in the group about the existence of that specific obstacle. Albeit at this particular moment, the obstacle is considered as irrelevant to the other components of the collaborative system (Collaborating Component 2 and Collaborating Component 3), as their current directions of movements do not lead to a potential collision with the obstacle, it does not mean that this will hold true when the positions of the components change during the system's execution.

Through the communication between the collaborating components of the collaborative system and their ability to share individual context information with the other members of the collaborating group, an "awareness-knowledge" map of the group and its surrounding is being created and this is exactly the source of the self-awareness of the collaborative system.

The context model of individual collaborating components consists of a set of context attributes which represents the context situation of the SUC. A context situation aggregates all considered context influences on a system (or context subject) at a given time. The context attributes, that are the constituents of a context situation, are defined, over a finite set of qualitative values or equivalent intervals for continuous variables. For example, the velocity of a robot would be represented with values like slow and fast instead of the exact speed. Using the same principle on the condition of an automated factory's floor, values such as dry or wet would be used instead of quantitative values about the surface quality. Type and number of context attributes depend on the type of the component and the application domain. A robot in a factory that operates individually on its own requires a different context model than a fleet of transportation robots that are collaboratively transporting goods. Although the individual context models in heterogeneous collaborating systems will differ, they require a common - usually domain specific - set of shareable context attributes.

A context attribute is the qualitative representation of one or more context information sources, as depicted in Figure 6. For example, a set of sensors like cameras or LIDARs can be used to detect obstacles and map the surroundings of a SUC. The surrounding can then be represented as a context attribute: obstacles. Mapping algorithms need to be applied, to obtain the values of context attributes from context information. Such algorithms map one or more information sources on one context attribute. In some cases, like velocity, this can be trivial, while in other cases, this requires more sophisticated algorithms. For example, an algorithm for determining the state of the context attribute OBSTACLES assesses detected objects on their impact on the SUC and thereby assigns a value to the context attribute.

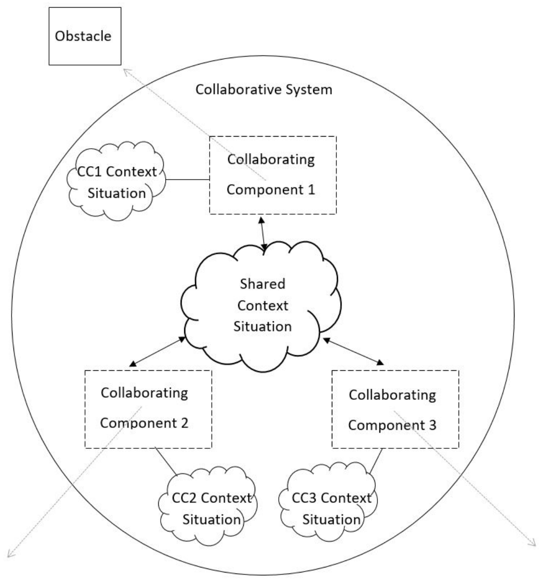


Figure 5: Collaborating Components sharing their context situations

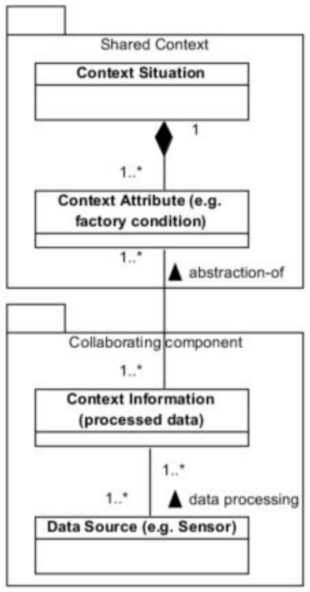


Figure 6: Context Situation Model

5 Related work

Explicitly designing all configurations of CES in open-context a priori is unfeasible. Gerostathopoulos et al. [Gerostathopoulos *et al.*, 2017] focus on the problem of cyber-physical systems (CPS) dealing with a large number of situations and configurations and corresponding self-adaptation strategies to enable adaptability in such systems. They propose a three-layer architectural style approach for handling of situations for which the system is not specifically designed for, and to generate new self-adaptation strategies at runtime to reflect the changes in the environment. The adaptability of a system is collectively enhanced by a number of meta-adaptation strategies proposed in the top layer of the architectural style. While addressing the similar problem as this paper, Gerostathopoulos et al. focus on system architecture and adaption strategies instead of a distributed context model as we do.

Götz et al. [Götz *et al.*, 2015] propose a role-based adaptive knowledge exchange technique working on partial runtime models of highly dynamic systems, that only reflect part of the state of the CES. A partial runtime model reduces the amount of knowledge that is shared between CPS entities. They present three strategies for knowledge exchange between collaborating subsystems, which support adaptation with respect to two dimensions: the runtime type of knowledge and conditions over the knowledge. In contrast to our qualitative approach, these strategies lead to either significant knowledge exchange or a limited knowledge of the context for the single system members. Potentially, a combination of both approaches could provide all members with a complete qualitative knowledge, but also allow members to obtain detailed knowledge on specific aspects by changing the exchange strategy.

6 Conclusion

In this paper, we presented an approach for modeling open context and context situations for collaborative embedded systems. To this end, we have explained how the definition and the understanding for system and system's context have been evolving. Beginning with traditional software engineering, through the SPES2020 approach, concluding with our definition for dynamic collaborative embedded systems operating in open context, which are subjects of constant run-time changes. Furthermore, we explained how such CESs are being formed.

The purpose of the proposed modeling concept is to provide input to mechanisms for system reconfiguration or self-adaptation with a sufficiently detailed, but still small and concise context model, allowing the mechanisms to adapt the system accordingly. Following this preliminary study, we will apply the approach to the transport robot use case from the CrESt project. This example will be used to determine the practicality and helpfulness of our approach.

Acknowledgments.
We would like to thank Wolfgang Böhm and the anonymous reviewers for their valuable feedback. This research has in part been funded by the *German Federal Ministry of Education and Research* under the grants no. 01IS16043, 01IS16043, and 01IS16043.

References

[Committee and others, 1990] IEEE Standards Coordinating Committee et al. Ieee standard glossary of software engineering terminology (IEEE std 610.12-1990). los alamitos. *CA: IEEE Computer Society*, 169, 1990.

[cre, 2017a] Collaborative Embedded Systems (CrESt). `https://crest.in.tum.de/`, 2017. [Online; accessed 01-July-2017].

[cre, 2017b] Open context of Collaborative Embedded Systems (CrESt EC4). `https://crest.in.tum.de/EC4.html`, 2017. [Online; accessed 01-July-2017].

[Daun *et al.*, 2016] Marian Daun, Jennifer Brings, Thorsten Weyer, and Bastian Tenbergen. Fostering concurrent engineering of cyber-physical systems a proposal for an ontological context framework. In *Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), 2016 3rd International Workshop on*, pages 5–10. IEEE, 2016.

[Forbus, 1997] Kenneth D Forbus. Qualitative reasoning. In *The Computer Science and Engineering Handbook*, pages 715–733. 1997.

[Gerostathopoulos *et al.*, 2017] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Adam Hujecek, Frantisek Plasil, and Dominik Skoda. Strengthening adaptation in cyber-physical systems via meta-adaptation strategies. *ACM Transactions on Cyber-Physical Systems*, 1(3):13, 2017.

[Götz *et al.*, 2015] Sebastian Götz, Ilias Gerostathopoulos, Filip Krikava, Adnan Shahzada, and Romina Spalazzese. Adaptive exchange of distributed partial models@ run. time for highly dynamic systems. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 64–70. IEEE Press, 2015.

[Philippe *et al.*, 2013] Lalanda Philippe, Julie A McCann, and Ada Diaconescu. Autonomic computing principles design and implementation, 2013.

[Pohl *et al.*, 2016] Klaus Pohl, Manfred Broy, Heinrich Daembkes, and Harald Hönninger. Advanced model-based engineering of embedded systems. In *Advanced Model-Based Engineering of Embedded Systems*, pages 3–9. Springer, 2016.

[Pohl, 2010] Klaus Pohl. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[Weiss *et al.*, 2013] Gereon Weiss, Florian Grigoleit, and Peter Struss. Context modeling for dynamic configuration of automotive functions. In *Intelligent Transportation Systems (ITSC), 2013, 16th International IEEE Conference on*, pages 839–844, 2013.

[Williams and de Kleer, 1991] Brian C Williams and Johan de Kleer. Qualitative reasoning about physical systems: A return to roots. *Artificial Intelligence*, 51, 1991.