Towards a QoS-aware Cyber Physical Networking Middleware Architecture

Melanie Brinkschulte
brinkschulte@uni-mannheim.de
University of Mannheim

Christian Becker
christian.becker@uni-mannheim.de
University of Mannheim

Christian Krupitzer
christian.krupitzer@uni-wuerzburg.de
University of Würzburg

ABSTRACT

Cyber-physical systems (CPSs) are used in various application fields. In those, multiple CPSs can cooperate by exchanging information (processed sensor data, derived data, etc.) to fullfill a task. Thus, cyber-physical networking plays an essential role. The communication in a cyber-physical network (CPN) as well as with the local sensors and actuators is complex. The applications of CPSs may use different programming abstractions, e.g. callbacks, streams, etc. for the interaction. Also CPNs are used in changing environments. Thus, they are afflicted with dynamic non-functional requirements. Due to this, an adaptive communication is important to enforce the resulting Quality of Service (QoS) constrains. Classical middleware architectures do not cover all of these aspects. In this paper we propose an architectur for a middleware that is designed to the needs of cyber-physical networking. Therefore, we strive to integrate multiple programming abstractions and communication protocols in an easy to use middleware. Also, we focus on the QoS-aspects of the communication to sensors, actuators and other CPS. Those constrains require an end-to-end view of system and a communication architecture that adapts to the changing available communication infrastructure in CPNs and correspondingly in CPSs. Therefore, this approach eases the development, test and maintenance of CPSs and CPNs.

KEYWORDS

Cyber-Physical Systems, QoS, Middleware, Cyber-Physical Networking, Cyber-Physical Network, End-to-End Communication, MAPE-K, Learning Classifier Systems

MISE '19, December 09–13, 2019, Davis, CA, USA

ACM ISBN 978-1-4503-7034-9/19/12…$15.00
https://doi.org/10.1145/3366616.3368149

ACM Reference Format:
Melanie Brinkschulte, Christian Becker, and Christian Krupitzer. 2019. Towards a QoS-aware Cyber Physical Networking Middleware Architecture. In *Middleware '19: ACM International Middleware Conference June 09–13, 2019, Davis, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3366616.3368149

1 INTRODUCTION

Cyber-physcial systems (CPSs) are omnipresent these days. For example they can be found in smart home environments, remote patient monitoring or traffic management. Those systems offer support in our daily routines. Today, CPS are mostly no longer isolated. Several CPS have to cooperate to achieve their goals. For this, communication between different CPSs, e.g. various cars or home environment systems, is necessary. Thus, cyber-physical networking plays an increasingly important role.

A cyber-physical network (CPN) is formed by several CPSs that are linked to the physical environment by sensors and actuators. These systems are typically coupled by a network and cooperate in order to fulfill a task. Therefore, they exchange processed sensor data and share information that changes their execution including the control of actuators. The interaction over a CPN as well as with the local sensors and actuators is constrained by non-functional requirements e.g. latency and deadlines. Furthermore CPSs are used in changing environments. In those the communication bandwidth can vary and other CPSs can appear or vanish any time. Due to this an adaptive communication is important to enforce Quality of Service (QoS) constrains. In addition, applications of CPSs may use different programming abstractions, such as streams, remote procedure call (RPC), callback or event-based approaches. As a result, the development, testing and maintenance of CPSs and CPN is rather complex. To relieve the system designers and to provide interoperability, a middleware which combines the mentioned aspects in a uniform way is needed.

Therefore, this paper has two contributions: First, we show the lack in the usability of classical middleware architecures for CPS. Furthermore, deficiencies in existing CPS middleware are identified. Second, we consequently will present an

architectural approach which closes the gap between classical middlewares and CPSs.
The paper is structured as follows: Section 2 discusses related work. Following, Section 3 describes research challenges arising in the domain of CPSs. To tackle these challenges, Section 4 presents our concept for a CPN middleware which operates within specified QoS bounds. Finally, Section 5 concludes this paper with a summary and future work.

2 RELATED WORK

As computing devices became more and more mobile, the research area of CPSs emerged out of the areas of Pervasive/Ubiquitous Computing and developments in the Internet-of-Things (IoT) domain. The connection to the physical world by sensors and actuators in combination with a distributed heterogeneous system is a big challenge. A classical middleware like CORBA (Common Object Request Broker Architecture) allows to map interface specifications to different programming languages and the interoperable inter-orb protocol (IIOP) provides interaction across different platforms with respect to programming languages and operation system [14]. Therefore, CORBA in combinition with IIOP can be used for the development and deployment of applications in distributed heterogeneous environments but there is no interaction to the physical world provided. This also holds true for other classical middleware approaches like e.g. DCOM [9], .NET [10] and Java RMI [8].
Self-organizing middleware architectures have been researched in the frame of the Organic Computing initiative [12]. There, bio-inspired principles are used to acquire so-called self-x capabilities. The OCμ respectively AMUN middleware [11] as well as the AHS [15] rely on artificial hormones. In contrast, the CARISMA middleware [6] uses an agent based approach featuring the ContractNet protocol. All these middleware architectures mainly deal with self-organizing task distribution. CPS aspects like sensor and actuator communication were no research focus there.
The connection of the physical world and computation devices regard new challenges for systems, especially real-time communication, context-awareness, adaptiveness, heterogeneity of devices and communication channels, scalability, or security [1–3]. As a result, the landscape of cyber-physical middlewares is diversified. The existing middlewares focus on different aspects.
Zhang, Gill, and Lu present an extension of the TAO middleware for timing aspects [16]. There, each functionality is seen as a sequence of events and scheduled to devices by a central scheduler. However, the authors do neither further stress the communication functionality nor adaptation and communication aspects. RDDS offers a publish/subscribe based data distribution middleware supporting semantic aware communication [4]. Further, this approach enables reactive and proactive adaptations to keep QoS bounds. Nevertheless, it does not offer programming abstractions and is focused on data dissemination rather than communication itself. iLAND enables time-bound reconfigurations of service-oriented soft real-time systems [1]. It offers reconfiguration graphs and paths for service compositions and discovery but neglects to cover specifics of the communication channels for enabling QoS-bounded communication. In [7], the authors extend the Flexible Time-Triggered (FTT) paradigm by a middleware architecture called FTT-MA. FTT-MA focuses on scheduling and resource management for dynamic reconfiguration at runtime. A very first prototype was implemented in CORBA. However, FTT-MA neglects communication specifics.
None of the presented works address multiple aspects (real-time communication, context-awareness, adaptiveness, heterogeneity of devices and communication channels, scalability, security) at the same time which is vital for a CPN middleware.

3 RESEARCH CHALLENGES

In this paper we want to propose a concept for a middleware that is designed to the needs of cyber-physical networking, hence, networking between CPSs. It differs from existing approaches as it strives to integrate communication with the physical word via sensors and acutators and QoS constrained communication with other cyber-physical devices into one middleware framework. To achieve this, we have to address multiple challenges in the CPSs development at the same time. For instance, the QoS-bound communication in CPSs and the communication to sensors and actuators requires an end-to-end view of the systems. Therefore, an approach that not only controls the network connection but also includes scheduling aspects, data dissemination, and adaptation of applications, devices and network connections is necessary. CPSs are usually build of three components[2]:

(1) **Sensors and actuators**: The connection between cyber world and physical world
(2) **System and application software**
(3) **Communication network**: The connection between CPSs

Whereas most of the research concentrates on the first two components, we want to concentrate on cyber-physical networking. A CPN is shown in Figure 1. In this network, every CPS is linked to its sensors and actuators. In order to fullfill a task, the CPSs cooperates by exchanging information (processed sensor data, derived data, state information etc.), that changes their execution including the control of actuator, via a network. An application example of a CPN in the field of Ambient Assisted Living (Smart Home) is shown in Figure 2.

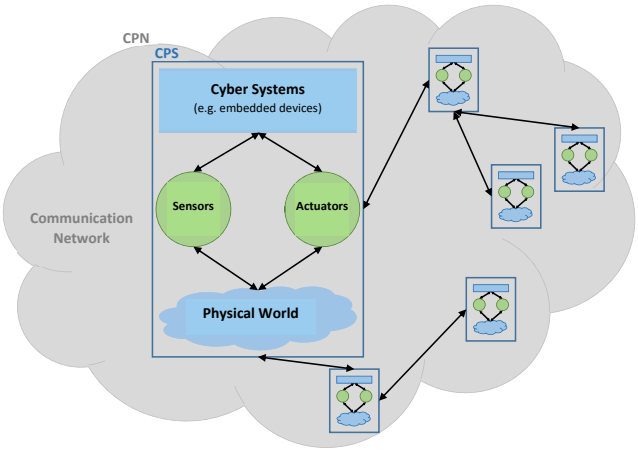

Figure 1: A Cyber-Physical Network containing several Cyber-Physical Systems

The task of this CPN is to monitor and regulate the health condition of a patient, e.g. by a pacemaker and a blood sugar regulator. In order to fulfill this task, multiple CPSs (blue boxes) cooperate with each other. Doing so, the overall system can detect several different health issues by consulting local or more advanced global health knowledge and react accordingly to keep the patient as healthy as possible (immediate detection and reaction to critical issues, emergency calls, automated detection of long term issues, automated consultation of a human doctor etc.). The interaction over a CPN as well as with the local sensors and actuators is constrained by non-functional requirements. For instance the loss of values has to be prevented. Therefore, sampling rates to the sensors have to be guaranteed as well as reaction time has to stay in bounds when a sensor signals data. The same holds true for controlling actuators where deadlines have to be guaranteed. Due to this, different assurances for applications for receiving and delivering data are important. Besides best-effort services, these may be classified in:

- Periods
- Deadlines
- Reaction time

In addition applications may use different programming abstraction, namely the following:

- Streams
- Remote procedure call (RPC)
- Callback
- Event-based communication

All these communication types and programming abstraction can also be seen in the example in Figure 2.
Furthermore, the surrounding conditions of CPSs, and respectively the networks of CPSs, can change during runtime. For example the position of the sensors and actuators can move, e.g. a moving patient being monitored and tracked, a moving car or robot. Additionally, the available communication infrastructure, e.g. the bandwidth, can vary. This for instance can happen when a patient is moved and therefore his communication is switched from fast Home-WLAN connection to a slow mobile connection. Also CPSs can dynamically vanish (out of reach or failure) and appear (within reach) in the CPN. As a result, adaptive communication plays a major role to enforce QoS constrains. Besides this QoS constrains in CPNs, several other challenges for CPSs/CPNs can be found, e.g. interoperability, robustness and dependability [1–3]. In our example in Figure 2 the life of the patient can be dependent of the system functionality. Thus, the system needs a fail-operational behavior. Also the different components (CPSs) have to cooperate by exchanging information and using this information to provide the specified services. Consequently, a system designer has an ambitious job if he wants to consider all possible requirements, programming abstractions and challenges, when working with CPSs and correspondingly with CPNs. In this paper, we present our concept of a CPN middleware, as depicted in Figure 3, which simplifies the development, testing an maintenance of CPSs and CPN by integrating interfaces to several programming abstractions, offering end-to-end QoS guarantees and adapts the communication to changing conditions.

4 ARCHITECTURE

Figure 3 shows the architecture of our CPN middleware. There are four possible scenarios for the communication in a CPN:

(1) Communication between an application *a* and an application *b* in the same CPS
(2) Communication between an application *a* in a CPS to its sensors/actuators
(3) Communication between an application *a* in *CPS 1* to another application *b* in *CPS 2*
(4) Communication between an application *a* in *CPS 1* to the sensors/actuators of another *CPS 2*

As the environment is dynamic and the conditions of the system can change constantly even when the communication is already in progress, an end-to-end view of the system is required for QoS-bound communication. Every presented scenario can have non-functional requirements like deadlines, periods and reaction times. Therefore, the presented architecture offers an easy-to-use modeling approach for QoS constraints of the applications by providing the possibility to define the requirements for the end-to-end communication. This can be either implemented by interface parameters or by aspect oriented programming techniques. The compliance of the time-bounds is observed and managed by the MAPE-K loop [5], which is a central component for the adaptation

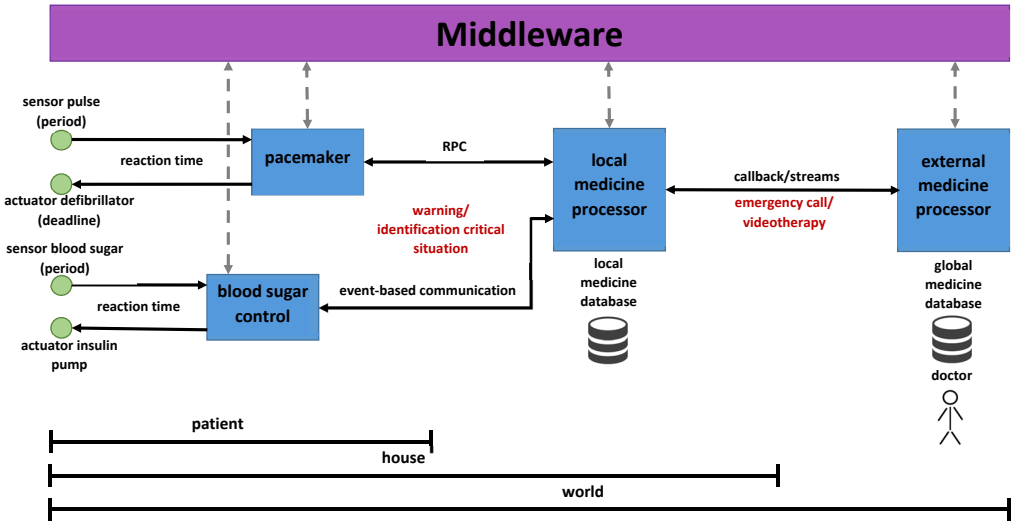

Figure 2: Application example of a CPN in the field of Ambient Assisted Living (Smart Home)

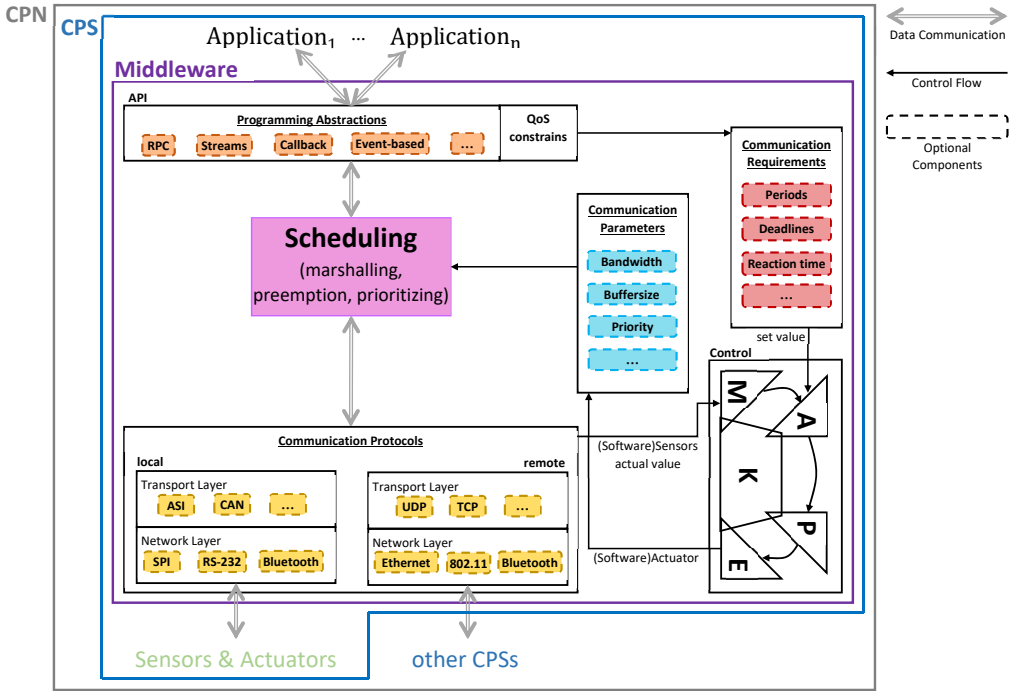

Figure 3: Architecture of our Cyber-Physical Networking Middleware

in this approach. The MAPE-K cycle is a feedback control loop which was first introduced by IBM and can be seen as architectural blueprint for autonomic computing. Therefore, it can be realized in many different ways using techniques from e.g. artificial intelligence. It has five essential functions:

(1) **Monitor:** Observation of the details from the managed resources
(2) **Analyse:** Analysis of the monitored data
(3) **Plan:** Planning of the actions needed to achieve the objectives

(4) **Execute:** Change the behavior of the managed resources based on the actions recommended by the plan function
(5) **Knowledge:** Knowledge about current and past data, strategies, etc. to manage the resources successfully

Due to the end-to-end nature of the QoS constrains, this loop not only has to act locally, but as well in coordination with MAPE-K loop of other middleware instances. This can happen by monitoring the actual values of local and global communication properties (communication requirements: available data bandwidth, communication time, ping time, etc.) and comparison of them to their set values. Afterwards the needed actions (adjustment of the communication parameters and communication channels, dislocation of functions and dropping of unnecessarily functions) to keep the defined QoS-constrains can be planed and executed. The output of the execute stage finally controls the communication parameters of the scheduler in our middleware, which is responsible for marschalling the programming abstraction, prioritizing the resulting messages, preempt lower priority messages, etc. A possible realization of the analyze and plan stages of the MAPE-K loop is to extend the concept of learning classifier systems[13]: the distributed learning classifier systems (DLCS). Figure 4 shows the basic principles of such a system.

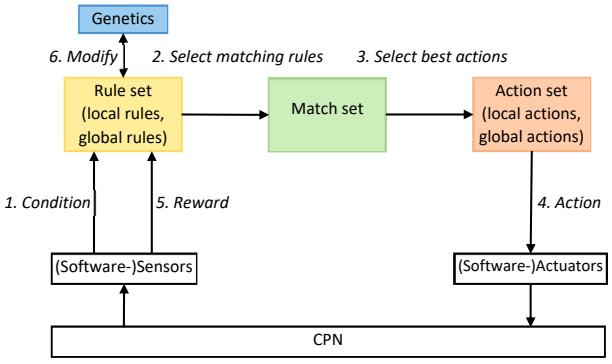

Figure 4: Distributed Learning Classifier Systems

Starting from the monitor stage, the current system condition (e.g. a current response time) is retrieved and compared to a rule set. This rule set consists of local rules only valid for this CPS and global rules valid for the entire CPN. Rules matching the condition are selected and written to the rule set. From this, the actions of the best matching and rewarded rules are written to the action set. Action can contain local actions only executable in this CPS or global actions which can be executed in the entire CPN. The best actions are then executed. The result is monitored and the reward for the applied rules is updated (increased if positive result, decreased if negative result). Furthermore, new rules can be created by genetic algorithms. The big advantage of DLCS is the low resource demands, which are suitable for CPS with low computational resources.

The adaptation to the changing environment has to be automatic and autonomous. Therefore, we use concepts from the field of self-adaptive systems. We have decided to rely on the MAPE-K cycle because it very well fits our architecture and has be proven to be effective in self-management of distributed systems. This approach even offers the possibility to schedule the communication of tasks with dynamic non-functional requirements.

Furthermore, the data transmission tasks between the communications partners in the earlier presented scenarios might need different programming abstractions. Those can be mapped, also dependent on the communication partners, to various communication protocols. Therefore, the task of the middleware is to find out which programming abstractions and which communication protocols are required and to provide them for the data transmission between the communication partners. In the following, some example combinations of the requirements, abstractions and protocols for the scenarios are shown.

Scenario 1 is a basic communication between two applications in the same CPS. Therefore, the applications can use the same programming abstractions, e.g. RPC, or different ones, e.g. RPC and callback. Local communication between applications on the same CPS will usually be routed via the loopback interface on the communication protocol level. The communication between two applications in the same CPS also can have non-functional requirements like delivery deadlines of messages in order to meet the execution deadlines of applications. Scenario 2 is an application communicating with local sensors and actuators of the CPS. Here, we can assume that we have a specific period for the sensor communication and a deadline for the actuator communication. For the data transmission between an application *a* and the sensors/actuators mostly event-based communication is used. This programming abstraction can then e.g. be mapped to the communication protocol ASI (Actuator-Sensor-Interface) for the direct communication with the sensors/actuators. Scenario 3 is a little bit more complex, because now we look at communication between different CPSs. This time we might have a fixed reaction time as QoS-bound. For the communication between two applications of different CPSs we pass data through two middleware instances. Therefore, the applications can use different programming abstractions for the communication. As example application *a* in *CPS 1* can use RPC and application *b* in *CPS 2* can use streams. For the communication between those application both programming abstractions can be mapped to the same communication protocol, e.g. TCP (Transmission Control Protocol). In scenario 4 we have a communication between an application *a* in

CPS 1 and the sensor/actuator of another CPS. Let's assume that this communication has a specific period for the sensor sampling, a reaction time to the sensor data and a deadline for the actuator control. For the data transmission application *a* in *CPS 1* can use e.g. callback (as soon as a sensor data value becomes available). This can be mapped to the communication protocol UDP (User Datagram Protocol) for the connection to the other CPS. There the communication protocol CAN (Controller Area Network) can be used for the communication with the sensor/actuator.
As can be seen, multiple combinations of requirements, programming abstractions and communication protocol can occur in CPNs. The presented middleware architecture therefore supports multiples of them for the CPSs to use. Since not all application scenarios might need all of these components, we rely on a microkernel concept. Components like programming abstractions and communication protocols are only included when needed. This saves resources, which is important on CPS with low computational resources.

5 CONCLUSIONS

In this paper we presented an architectural approach for a CPN middleware. Our objective is to develop such a middleware that combines all the challenges presented in Section 3 in a uniform way. To achieve this, it is necessary to classify and understand interactions between applications in CPNs with respect to communication over the network as well as communication to sensors and actuators. For the analysis of functional and non-functional requirements we will examine different application fields in the CPS domain, e.g. the automotive area and smart home environments. In each area, different aspects of cyber physical networking can be found. We are going to analyze the communication between CPS by e.g. considering the vehicle-to-vehicle and vehicle-to-infrastructure communication in the automotive domain. Also the data transfer within a car, and correspondingly in one CPS, to sensors and actuators will be observed. Eventually, we will determine which programming abstractions and communication protocols are used in CPNs and therefore have to be provided by the uniform middleware. Furthermore, we want to focus on the QoS-aspects of the network communication. For this we will analyze the requirements of the CPSs for the end-to-end communication and provide a possibility to define them, e.g. by parameter passing or in an aspect-oriented programming style.
To evaluate our results, a prototypic implementation will be realized. Thereby, the communication can be adjusted based on the available communication infrastructure and the non-functional requirements of the CPSs. For this adaptive behavior we use the MAPE-K approach in our conception (see Figure 3). For this is only an abstract control loop, an appropriate implementation of the sections monitor, analyze, plan, execute and knowledge and their connections has to be realized. As example, classifier systems can be used for the implementation of the analyse and plan section.
Finally, the developed prototype will be evaluated in different scenarios in the CPS domain.

REFERENCES

[1] M. Garcia Valls, I. Rodríguez Lopez, and L. Fernández Villar. 2013. iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems. *IEEE Trans. Ind. Informatics* 9, 1 (Feb 2013), 228–236.

[2] V. Gunes, S. Peter, T. Givargis, and F. Vahid. 2014. A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems. *KSII Transactions on Internet and Information Systems* 8 (Dec 2014), 4242–4268.

[3] Z. Jaroucheh, X. Liu, and S. Smith. 2009. A Perspective on Middleware-Oriented Context-Aware Pervasive Systems. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*. IEEE, 249–254. https://doi.org/10.1109/COMPSAC.2009.142

[4] W. Kang, K. Kapitanova, and S. H. Son. 2012. RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics* 8, 2 (May 2012), 393–405.

[5] J. Kephart and D.M. Chess. 2003. The Vision Of Autonomic Computing. *Computer* 36 (Feb 2003), 41 – 50.

[6] M. Nickschas and U. Brinkschulte. 2008. Guiding Organic Management in a Service-Oriented Real-Time Middleware Architecture. In *6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2008)*. Capri, Italy.

[7] A. Noguero, I. Calvo, and L. Almeida. 2012. A Time-Triggered Middleware Architecture for Ubiquitous Cyber Physical System Applications, Vol. 7656. 73–80.

[8] Oracle [Online]. [n.d.]. Java Remote Method Invocation - Distributed Computing for Java. https://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html

[9] F. Plášil and M. Stal. 1998. An architectural view of distributed objects and components in CORBA, Java RMI and COM/DCOM. *Software - Concepts & Tools* 19, 1 (01 Mar 1998), 14–28.

[10] J. Prosise. 2002. *Programming Microsoft .NET*. Microsoft Press, Redmond, WA, USA.

[11] M. Roth, J. Schmitt, R. Kiefhaber, F. Kluge, and T. Ungerer. 2011. Organic Computing Middleware for Ubiquitous Environments. *Organic Computing - A Paradigm Shift for Complex Systems, Springer Verlag, Basel* (2011).

[12] H. Schmeck. 2005. Organic Computing - A New Vision for Distributed Embedded Systems. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005)*. Seattle, USA, 201–203.

[13] O. Sigaud and S. W. Wilson. 2007. Learning classifier systems: a survey. *Soft Computing* 11, 11 (01 Sep 2007), 1065–1078.

[14] S. Vinoski. 1997. CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments. *IEEE Communications Magazine* 35, 2 (Feb 1997), 46–55.

[15] Alexander von Renteln, Uwe Brinkschulte, and Mathias Pacher. 2011. The Artificial Hormone System - An Organic Middleware for Self-organising Real-Time Task Allocation. *Organic Computing - A Paradigm Shift for Complex Systems, Springer Verlag, Basel* (2011).

[16] Y. Zhang, C. Gill, and C. Lu. 2008. Reconfigurable Real-Time Middleware for Distributed Cyber-Physical Systems with Aperiodic Events. In *2008 The 28th Inter. Conf. on Distributed Computing Systems*. 581–588.