Architectural Allocation Alternatives and Associated Concerns in Cyber-Physical Systems: A Case Study

Jakob Axelsson
Swedish Institute of Computer Science (SICS)
Kista, Sweden
jakob.axelsson@sics.se

ABSTRACT

Cyber-physical systems is an extension of traditional embedded systems, where communication to the outside world is given more emphasis. This leads to a new design space also for software development, allowing new allocation strategies for functionality. In traditional embedded systems, all functionality was inside the product, but now it becomes possible to partition the software between the embedded systems and IT systems outside the product. This paper investigates, through a case study from the automotive domain, possible new allocation alternatives where computation is offloaded from the embedded system to a server, and what additional architectural concerns this leads to, including performance, resource utilization, robustness, and lifecycle aspects. In addition, the paper addresses new opportunities created by allocating functionality outside the embedded systems, and thus making data available for extended services, as well as the larger concerns that result on the organizational level, including new competency in architecture and DevOps.

Categories and Subject Descriptors

D.2.11 [**Software Architecture**]: Domain-specific architectures.

General Terms

Design, Experimentation.

Keywords

Cyber-physical systems, architecture, system-of-systems, allocation, cloud.

1. INTRODUCTION

Cyber-physical systems (CPS), i.e., systems containing interacting elements of mechanics, electronics, and software, are of ever-increasing importance in our society. They are part of a large range of industrial products, in domains such as automotive, aerospace, and industrial automation. Traditionally, these domains have focused on embedded systems (ES), where the emphasis is on the computers integrated in a product, which are interacting with the physical parts of the same product. One of the fundamental differences between ES and CPS, is the connections from the ES to other software-intensive systems outside the product [1]. These other systems may be tightly integrated with the ES, or be more independent as is the case in systems-of-systems (SoS) [2][3].

The architecture of a system is defined as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" [4]. In a traditional ES, the architecture is often focused on a hardware view, showing the electronic control units (ECUs) and their communication links. This is complemented with various software oriented views, showing the functions of the system, and how the functions are allocated to ECUs. Also, there is often a view showing how the ECU software is layered into application components, operating system, device drivers, etc. In many domains, such as the automotive, it is common to use distributed embedded systems, where the ECUs are connected using communication networks.

It is important to realize that the focus has traditionally always been on the ES as part of a delimited product, with no or limited connections to computer systems outside that product. This is one of the main differences between ES and CPS, where in the latter the connections to the outside are emphasized. In a CPS architecture, it becomes natural to also add computational units outside the product, on dedicated servers or even cloud solutions.

The main contribution of this paper is to study, through a practical case, some of the trade-offs architects need to perform in such a situation. What are the opportunities and challenges that we need to face if the system border is extended to IT and cloud, who thus become parts of the system-of-interest? The traditional ES and new CPS architectures are illustrated in Figure 1.

This paper addresses the following research questions:

1. What are the new allocation alternatives that emerge in CPS compared to traditional, closed ES?
2. Which concerns become important in these allocation alternatives?
3. What are the broader consequences and opportunities of using the new allocation strategies?

To our knowledge, these questions have not been studied in detail in previous research, and the paper thus contributes to an increased understanding of the possibilities and risks provided by CPS.

The research method used is explorative and based on design science [5], where a specific case is designed on an architectural level, and followed up by prototype implementations of different alternatives, thereby giving concrete examples of both the design space and concerns that arise. The observations and experiences gained from this provides empirical evidence that is used to derive at least partial answers to the questions.

ECSAW '15, September 07 - 11, 2015, Dubrovnik/Cavtat, Croatia
 ISBN 978-1-4503-3393-1/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2797433.2797448

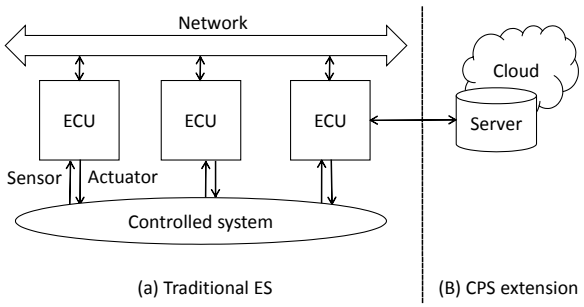

Figure 1. Traditional ES vs. CPS architecture

The remainder of the paper is structured as follows: In the next section, the case study is introduced, followed by a description of the allocation alternatives that occur in traditional ES solutions and in a CPS. Then, in Section 4, the architectural concerns are enlisted and used for evaluating the alternatives. Section 5 discusses the findings from the case study and expands it to broader questions related to the organizations and skills needed, and Section 6 provides an overview of some related work. Finally, in Section 7, the conclusions are summarized together with indications of future work.

2. CASE DESCRIPTION

In this section, the case study is described, which is about extending an existing product in the automotive domain with additional functionality. First, the existing product and its architecture will be presented, followed by a description of the added functionality and some important properties.

2.1 Existing product and architecture

The case study is placed in the context of automotive systems or mobile robots. It has been carried out using a demonstration system called the Mobile Open Platform for Experimental Design (MOPED) [6], which consists of a model car in scale 1:10. The car has three ARM-based ECUs running at 700 MHz onboard, two of which execute the automotive software standard AUTOSAR [7], and a third which is used for telematics and runs Linux. The ECUs are connected via an internal network in the car, which is using the CAN protocol running at 500 kbit/s. The car is equipped with various sensors and actuators, including steering and propulsion. It also has external communication through WiFi, making it possible to send and receive data from other systems. The overall architecture is thus similar to the generic CPS architecture shown in Figure 1 above.

The purpose of the demonstration platform is that it should be very realistic and mimic real, industrial systems with high accuracy when it comes to the software. This means that experiments done using MOPED will yield results that are likely to be useful also for CPS in practice.

2.2 Added functionality

In the case study, the purpose was to extend the MOPED platform with a vision sensor, together with advanced image processing algorithms. The results from the algorithms will be used by control functions in the car, which means that it is important to achieve a sufficient update frequency and sufficiently short response times. The output from the algorithm is also useful for other cars, and there is thus a need for functionality to communicate those results externally to a server (in this case based on the MQTT protocol, [8]), in order to form a SoS. An overview of the functional flow is shown in Figure 2.

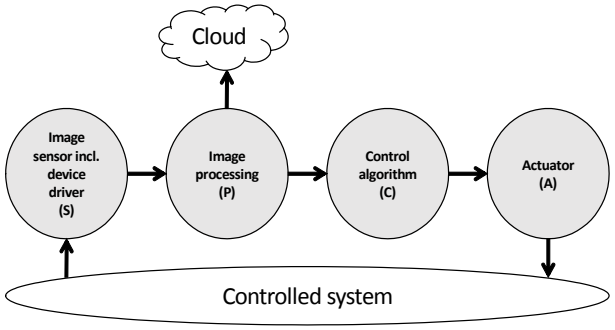

Figure 2. Functional flow.

The initial intent was to implement the functionality in a traditional ES fashion, i.e. by allocating the algorithm in one of the ECUs. The image sensor requires device drivers, which are only available under Linux, and therefore the telematics unit is the one where the sensor must be connected. The control algorithms and actuators are also by necessity in one of the other ECUs.

2.3 Performance

The image processing algorithm is computationally intensive, and even if an ECU was dedicated to this algorithm, it would be difficult to reach the response times needed by the control algorithms. If the ECU also has other functionality allocated, it would be difficult to reach satisfactory response times. Some jitter is also inevitable, due to the nature of the algorithm, since the processing time depends on what is actually present in the captured image.

The data transfer from the sensor to the image processing is in the order of a few hundred kb for each sample, although there is some flexibility in this using image compression or reduced resolution. In the subsequent steps, only a few tenths of bytes are transferred.

3. ALLOCATION ALTERNATIVES

In this section, we will study what allocation alternatives exist for the functionality described in the previous section, thereby providing the answer to the first research question: What are the new allocation alternatives that emerge in CPS compared to traditional, closed ES?

The discussion will focus on four alternatives for allocating the image processing component to each of the existing ECUs or to a server. The reason for focusing on this component is that it is the most demanding one in the system, and the one which is least constraint in allocation since it does not interface directly with the hardware.

An additional alternative could have been to add a fourth ECU to the system, dedicated to the image processing, but this was ruled out due to lack of physical space in the vehicle. One could also have considered redesigning one of the ECUs to include more powerful hardware, or hardware acceleration through application specific integrated circuits (ASIC) or graphical processing units (GPU). However, this would have both a significant cost for redesign and production, but also increase the lead time.

In the following subsections, each of the four alternatives are described. They are also summarized graphically in Figure 3.

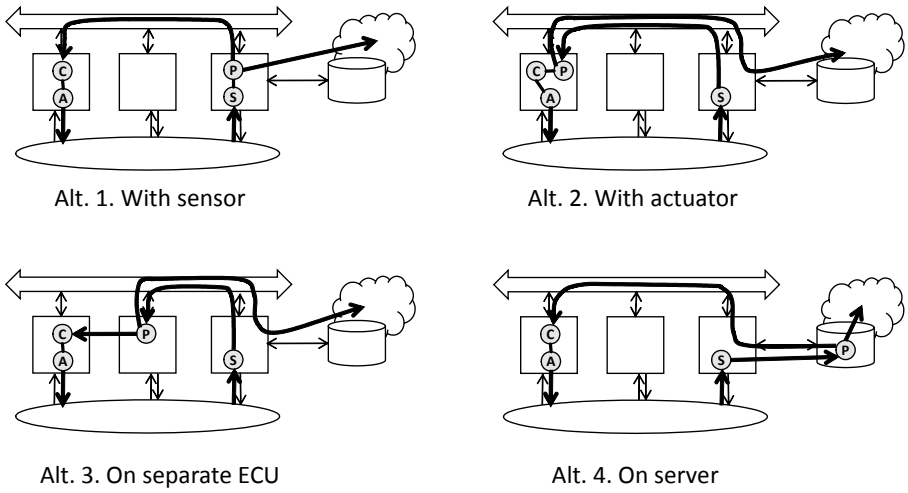

Figure 3. Image processing allocation alternatives.

3.1 With sensor

The first alternative studied is to allocate the image processing together with the sensor. This makes a lot of sense, since it would keep the massive data transfer from the sensor out of any communication networks. It has also the benefit that this ECU, which is responsible for telematics, has a direct external connection for publishing the result to other users.

At the same time, it limits the processing frequency severely, since the ECU has other essential and time critical tasks. In the implementation used, the typical processing time was in the order of almost 1s, but with a considerable jitter depending on other workloads. Since this node is running Linux, software development is fairly easy, since many image processing libraries used are available on this platform.

3.2 With actuator

The second alternative was to instead move the processing to the AUTOSAR based ECU where the control algorithm and actuator are allocated. This ECU has more time deterministic behavior due to a known workload of control algorithms and a real-time operating system, and the jitter of the image processing would then be reduced compared to the previous alternative. However, the raw processing time would be similar, i.e. around 0.5-1s.

A major drawback of this solution is that the image data has to be transferred over the CAN network, and this can be expected to take several seconds, and potentially have a negative impact on the performance of other traffic. Also, software development would suffer from the lack of image processing libraries on this platform, leading to an increased effort in porting the algorithms. Since AUTOSAR does not provide a file system, it is difficult to save sensor data for post mortem analysis. Finally, the result of the processing has to be sent over the network back to the telematics unit for transfer to external servers.

3.3 On separate ECU

Another possibility is to use the third ECU in the system for the image processing. This ECU has a slightly lower workload than the actuator ECU, giving a potential of reducing the processing time marginally. However, in all other aspects it suffers from the same performance problems as the previous alternative.

3.4 On server

Given the difficulties encountered with a traditional ES solution to the problem, it was decided to also investigate the potential benefits of a CPS system in offloading the image processing to an external server. This would solve the performance problems of the processing, since even a simple server can do the processing within about 100ms. Scalability would also not be an issue, since each processing task is independent and thus possible to allocate to different hardware units. The processing result would also be directly accessible to other cars. Table 1 gives an overview of the timing of the different alternatives, based on estimates of implementations on the MOPED platform.

Software development would also be easier, given the access to a wide range of libraries and also a better development environment than when working on embedded ECUs, including easy access to the sensor input data.

The main question about this solution is of course the external communication, both regarding response time and jitter caused by other traffic over the shared network.

Table 1. Indicative timing analysis of control loop.

	Time (ms)	Alt. 1	Alt. 2	Alt. 3	Alt. 4
CAN, parameter	0.1	1	0	0	1
CAN, image	2000	0	1	1	0
External, parameter	0.01	0	0	0	1
External, image	200	0	0	0	1
Computation ECU	500-1000	1	1	1	0
Computation server	< 100	0	0	0	1
Total roundtrip (ms)		500.1-1000.1	2500-3000	2500-3000	< 300.11

4. CONCERNS

In the previous section, the alternatives were described, and for each of them a number of concerns were identified, primarily in relation to performance which would be the main differentiator in many trade-off situations in traditional ES. In this section, we will expand on those concerns to cover some new ones which become relevant as a consequence of including CPS based alternatives in the evaluation, thereby addressing the second research question.

At the end of the section, a systematic evaluation of the alternatives based on the full set of concerns is presented.

4.1 Traditional ES concerns

As indicated above, the traditional ES concerns apply also in this case, and the primary ones are:

- *Response time*: What is the time from reading the sensor data to writing to the actuator?
- *Jitter*: How much is the variation in response time?
- *Resource utilization*: How much computational resources are needed? In the example, the main concern was CPU time, but the concern also includes consumption of other limited resources, such as memory footprint and network bandwidth. The resource utilization has a direct relation to the cost of the product.

4.2 Additional CPS concerns

When the ES is extended with server side functionality accessed over shared networks, the following technical concerns are added:

- *Server resource utilization*: How much computational resources are needed on the server? This relates directly to the investment and operational cost, but may also affect scalability.
- *Robustness*: How well does the system function in the presence of invalid input or stressful environmental conditions? In the example, the fact that the system now includes resources which are shared with unknown other users means that latencies and availability cannot be analyzed statically. In traditional ES architectures, all the resources are dedicated to the system, and hence under full control.

4.3 Lifecycle concerns

So far, the discussion has mainly been about technical concerns. However, there are many factors related to the lifecycle of the system that differ between traditional ES and CPS architectures:

- *Development support*: Development on embedded hardware is much more complicated than on desktop or server systems using standard hardware and operating systems. This includes all stages of the development cycle, such as programming where compilers and libraries are specific; debugging, where it is hard to see what is going on; etc.
- *Software updates*: One area where the difference becomes dramatic with the server based solution is software updates. If the sensor algorithms are updated, in the server based solution the new version only needs to be deployed in one place and all users instantly get access. If it is instead allocated in the ES part, it has to be deployed to all system instances which can take a very long time and possibly not happen at all for some instances. This leads to problems with many different variants being used.
- *Operational feedback*: In ES, it is very difficult to get access to data from the operating system, and developers often lack insight into how the systems actually work in practice. With a CPS solution, important data is transferred to the server, and can be used for both off-line and on-line analysis. This opens up the possibility to create a learning system, which applies advanced data processing to discover improvements to the algorithms, which can then be deployed through a software update on the server. Approaches such as A/B testing commonly used for web based systems can now also be applied in the CPS domain, leading to improved quality.

4.4 Analysis of alternatives

To summarize the analysis, a Pugh analysis [9] has been performed, as shown in Table 2. In the table, each alternative is rated qualitatively against the concerns. Alternative 1, the traditional ES solution, is considered a baseline, and is hence given a rating of 0 for each concern. The other alternatives are rated + if they are considerably better than the baseline, - if they are worse, and 0 if they are equivalent.

Since the table does not show any weights between concerns, and has a very qualitative analysis scale, it is not meaningful to sum columns and compare them to find the best alternative. Still, the table clearly indicates that alternatives 2 and 3 provide few benefits compared to alternative 1, and thus the main choice is between alternatives 1 and 4.

The table can be used for identifying important trade-offs that need to be studied further to determine if alternative 4 is superior. One such trade-off is that this alternative gives a shorter response time at the cost of increasing jitter and decreasing robustness. Another factor is that alternative 4 trades ES resources for server resources, and the total cost for this can be calculated further.

5. DISCUSSION

Having analyzed the different alternatives with respect to concerns, we will now broaden the discussion to address research question 3. In this, we generalize from the example and discuss patterns related to quality of service, extended services, architecture, and operations, that we believe will recur also in other systems.

Table 2. Evaluation of alternatives against concerns.

Concern	Alt. 1	Alt. 2	Alt. 3	Alt. 4
Response time	0	-	-	+
Jitter	0	+	+	-
ES resource utilization	0	0	0	+
Server resource utilization	0	0	0	-
Robustness	0	0	0	-
Development support	0	-	-	+
Software updates	0	0	0	+
Operational feedback	0	0	0	+

5.1 Application quality of service

An important aspect is the quality of service (QoS) that can be achieved, and how this differs between a traditional ES solution and a CPS one. The term QoS is often used for describing properties of communication networks, but here the focus is on application QoS, involving the combination of computation and communication. This means that a lower service level in the communication can be compensated by alternative computation algorithms that may differ in their quality but also resource usage. We will now discuss three complementary approaches to increasing the QoS in CPS.

5.1.1 Redundant computation

One possibility that comes to mind in the example is to combine alternatives 1 and 4, and implement both the on- and off-board sensor processing. When sensor data has been captured, it is sent over the network to the server, and at the same time, a local processing is initiated. The response that arrives first is used, thereby putting an upper bound on the response time (even in the case of a complete network failure), while getting the possibility of a much lower average response time.

5.1.2 Load balancing

The second approach is to include a control functionality that analyzes performance, and uses this to choose the best approach from a set of alternatives. In situations with low communication bandwidth, this could include choosing a faster but less performant algorithm; compressing the sensor information using a lossy algorithm; etc.

5.1.3 Jitter compensation

A control algorithm is often built on the assumption of periodic inputs, and to achieve that in a situation where input is subject to considerable jitter, one approach is to create a local estimator that can predict future values. Once a new value arrives, the estimates are compensated to use the best available knowledge. Kalman filters [10] are typically used for such tasks.

5.2 Extended services

An intriguing opportunity provided by a centralization of data processing is the possibility to use the data to build new or improved services. By having multiple ES delivering data, the server can build a much more complete picture of the world. In the example, the different vehicles driving around will report data from different places, and by comparing this, it might be possible to extrapolate what the world looks like in between the samples. This can be used to provide an overall world view useable for other users, but can also be used to raise the quality of the responses from the sensor processing algorithms, by combining the sensor data with *a priori* information.

5.3 Architecture

One of the fundamental shifts in moving from ES to CPS is the change of system border. Previously, the system-of-interest used to be very clear and easy to identify, by simply looking at the electronic components of the product where the ES is included. Now, there are also server or cloud components inside the system border, components which may be shared also with other systems dynamically. Similar aspects apply for the communication networks, where sharing also starts to occur with other unknown applications. It is no longer evident what is inside or outside the system border.

To achieve properties such as QoS, robustness, scalability, architectural solutions are needed which include redundancy, and self-protective mechanisms, including but not limited to security. The range of aspects architects need to deal with expands significantly. Most likely, layered or hierarchical architecture patterns with service based interfaces will become even more important to correctly implement functionality and mechanisms on different levels of the system.

5.4 DevOps competency

Organizations that develop ES typically have a large competency in electronics and embedded software. When the system border is expanded to also include server side functionality, architects and developers need to broaden this to also understand the trade-offs and implementation details concerned with cloud computing, internet protocol based communication, etc.

However, these organizations additionally become involved with operating IT systems, ensuring that they maintain a high availability level while still being able to encompass rapid changes in functionality. In essence, the organizations will face a close co-operation between development and IT operations (DevOps), which also includes embedded system developers.

6. RELATED WORK

There are a number of fairly recent papers that discuss different aspects of CPS architecture with a bearing on our work. A first group considers the overall logical structure of a system consisting of both on-board and off-board components. Most of these papers advocate a layered approach, similarly to the one sketched in Section 5.3. In the domain of automotive CPS, [11] proposes the V-cloud architecture consisting of three interdependent layers, namely in-car, vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) systems. The V2V and V2I layers are also connected to cloud environments. [12] suggests a similar architecture, but call the levels Micro, Meso, and Macro layers. They also indicate on a very high level how different functions can be allocated to the layers, including computation off-loading, but do not provide any more detailed analysis. In [13], yet another similar architecture is presented, and it is discussed how the availability of data in the cloud can be used for data mining of warranty data, and for parking assistance.

A second group of papers look at a more logical structuring. [14] applies techniques from service oriented architecture (SOA) on CPS, arriving again at a three layer architecture, with an environmental tier containing the physical nodes, a control tier for controlling access to the devices, and a service tier which makes interfaces available to other systems. SOA is also used by [15] in the context of industrial automation, with the inclusion of cloud based functionality.

A third group discusses performance issues for both computation and communication, bringing up properties like the ones discussed in Section 3. [16] present different communication techniques for CPS, looking at both the low-level technologies for wireless communication including performance aspects, as well as its application in different industry sectors. [17] discusses various aspects of swarm computing, i.e., utilizing the devices' computational power, and identifies different trade-off points, such as computation vs. communication. Finally, [18] evaluates, through a small case study, performance aspects of cloud computing when applied to time-critical CPS applications, concluding that platform-as-a-service is the most suitable cloud

approach, but also that even small details in the implementation of the cloud services can have a large impact on performance.

All these aspects are relevant for the kind of systems discussed in this paper. However, they do not address the analysis of different allocation alternatives for functionality, and the trade-offs it leads to for different properties. Also, lifecycle consequences are not in the focus of these references.

7. CONCLUSIONS AND FUTURE WORK

Not so many years ago, the idea to send sensor data from an embedded system to a server for processing, and then feeding back the result for usage in control loops would have been very far fetched. With the rapid development of communication technology, which is still ongoing, it can be expected that such solutions will become increasingly attractive. In this paper, the consequences of such allocations have been studied, and it is clear that there are many benefits resulting from such an approach, but also challenges.

The challenges are primarily technical, including how to deal with increasing variation in response time and availability due to the use of shared communication and computation resources. The solutions to this touch all domains of CPS, including control engineering, ES engineering, communications, and algorithms. There are also additional competency challenges related to the use of technology not traditionally associated with ES, such as cloud systems, which leads to a DevOps situation including also IT operations.

The opportunities are many, and in particular include improved development process, continuous software upgrades, operational feedback, and extended services based on the data from many systems.

In the future, we plan to look at more cases to reach a deeper understanding of recurring patterns in this kind of solutions, and also further analyze potential solutions for increased robustness and for supporting DevOps of CPS.

8. ACKNOWLEDGMENTS

This work is supported in part by The Knowledge Foundation in Sweden, as part of the ORION project (Grant No. 20140218), and in part by VINNOVA, the Swedish Agency for Innovation Systems (Grants No. 2013-03492, 2014-05599).

9. REFERENCES

[1] Broy, M. and Schmidt, A. (2014). Challenges in Engineering Cyber-Physical Systems. *Computer*. 47(2):70–72.

[2] Maier, M.W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*. 1(4):267–284.

[3] Axelsson, J., and Kobetski, A. (2014). Architectural Concepts for Federated Embedded Systems. In *Proc. 2nd Intl. Workshop on Software Engineering for Systems-of-Systems*.

[4] ISO/IEC/IEEE Std. 42010 (2011). *Systems and software engineering — Architecture description*.

[5] Hevner, A.R. et al. (2004). Design Science in Information Systems Research. *MIS Quarterly*. 28, 1 (2004), 75–105.

[6] Axelsson, J. et al. (2014). MOPED : A Mobile Open Platform for Experimental Design of Cyber-Physical Systems. *Euromicro SEAA*.

[7] AUTOSAR consortium. www.autosar.org.

[8] Banks, A. and Gupta, R. (Eds.) MQTT Version 3.1.1. *OASIS Standard*, 2014.

[9] Pugh, S. (1991). *Total Design: Integrated Methods for Successful Product Engineering*. Addison-Wesley, New York.

[10] Faragher, R. (2012). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. *IEEE Signal Processing Magazine*, pp. 128-132.

[11] Abid, H., et al. (2011). V-Cloud: vehicular cyber-physical systems and cloud computing. In *Proc.4th Intl. Symposium on Applied Sciences in Biomedical and Communication Technologies*.

[12] Wan, J. et al. (2014). VCMIA: A novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing. *Mobile Networks and Applications* 19(2): 153-160.

[13] He, W., Yan, G., and Da Xu, L. (2014). Developing vehicular data cloud services in the IoT environment. *IEEE Trans.on Industrial Informatics*, 10(2), 1587-1595.

[14] La, H. J., and Kim, S. D. (2010). A service-based approach to designing cyber physical systems. In *Proc. IEEE/ACIS 9th International Conference on Computer and Information Science*, pp. 895-900.

[15] Karnouskos, S. et al. (2012). A SOA-based architecture for empowering future collaborative cloud-based industrial automation. In *38th Annual Conference on IEEE Industrial Electronics Society*, pp. 5766-5772.

[16] Chen, M., Wan, J., and Li, F. (2012). Machine-to-Machine Communications. *KSII Transactions on Internet and Information Systems*, 6(2), 480-497.

[17] Rabaey, J. M. (2011). The swarm at the edge of the cloud-a new perspective on wireless. In Proc. *IEEE Symposium on VLSI Circuits*, pp. 6-8.

[18] Olson, M., and Chandy, K. M. (2011). Performance issues in cloud computing for cyber-physical applications. In *IEEE International Conference on Cloud Computing*, pp. 742-743.