Towards Flexible and Secure End-to-End Communication in Industry 4.0

Silia Maksuti[1,3], Ani Bicaku[1,3], Markus Tauber[1], Silke Palkovits-Rauter[1], Sarah Haas[2] and Jerker Delsing[3]

[1]*University of Applied Sciences Burgenland* - Eisenstadt, Austria
[2]*Infineon Technologies Austria AG* - Graz, Austria
[3]*Luleå University of Technology* - Luleå, Sweden

***Abstract*—The digital transformation of industrial production is driven by the advance of cyber-physical production systems (CPPS) within which raw materials, machines and operations are interconnected to form a sophisticated network. Making such systems self-adaptable is a priority concern for the future implementation of Industry 4.0 application scenarios. In this position paper, we design a meta-model and use it as a tool to describe an end-to-end communication use case from an ongoing research project. Based on this use case we develop a business process performance and security trade-off model, which shows that maximazing both parameters at the same time is not possible, thus an efficient balance between them has to be achieved. Motivated by the result, we propose self adaptation as a solution towards a flexible and secure end-to-end communication in Industry 4.0. To identify and document the self-adaptation points in a structured methodological and lightweight way we use the bespoken meta-model.**

I. Introduction

The fourth industrial revolution, referred to as Industry 4.0, is often understood as the application of the generic concept of cyber-physical systems (CPS) to industrial production systems [1]. In cyber-physical production systems (CPPS) all machines are properly equipped with sensors, actuators and communication technology and connected to each other within an Industrial Internet of Things (IIoT). The stream of data collected and processed from these systems is the new raw material in such connected industry. Innovations based on collecting, evaluating, and using this data can improve existing processes and create new business models. At this early development phase, there is an urgent need for a clear description of flexible and secure CPPS. Thus, based on RAMI 4.0 reference model [2], we derive a meta-model that is used as a tool to support the understanding of such systems, to reduce the complexity and to provide a consistent terminology.

Despite the benefits, there are still a lot of challenges to address when dealing with CPPS in Industry 4.0. CPPS have high security demands, so even if security aspects have been considered at the design and the implementation phase, the continuous change of environment conditions and requirements of the system itself can affect them later during the operation. Furthermore, security can not be seen as indipendent from, for example performance, legal or safety aspects of such system. Improving security without risking to negatively affect other aspects is a main concern for complex systems handling a large number of interconnected components.

To identify this research need we consider a use case from an ongoing research project, addressing the communication from the edge devices to the backend system via IIoT gateways. We map the components of this use case in the corresponding objects of the meta-model to give an overview how it can be used to describe such systems. Based on this use case we develop a business process performance and security trade-off model to identify the need for flexible and secure end-to-end communication in Industry 4.0. We consider traffic characteristics, such as offered load, and delivery time, to evaluate the business process performance of such systems (e.g., effectiveness) for different security levels of the communication protocol. The trade-off model results have shown that performance tends to decrease when improving security of such systems and vice versa.

Therefore, we propose self-adaptation based on autonomic computing approach [3] as a solution for achieving an efficient balance between performance and security. Self-adaptable systems mitigate the risk of negatively affecting one parameter while improving the other since they have the ability to change with the environment and requirements, making these systems flexible. As mentioned by [4], self-adaptation in CPPS is one of the five areas that have priority for the future implementation of Industry 4.0 application scenarios.

Our main contributions and initial findings in this position paper include the following:

- a CPPS meta-model derived based on RAMI 4.0 reference architecture model, which can be easily adopted and scaled to the needs of the project
- an end-to-end communication use case described using the derived meta-model,
- self-adaptation as a solution for improving the trade-off between business process performance and security, thus providing a flexible and secure end-to-end communication in Industry 4.0, and
- using the meta-model to identify and document the self-adaptation points in a structured methodological and lightweight way.

The paper is structured following the contributions above.

II. Related Work

To derive the CPPS meta-model presented in this paper, we have investigated existing architecture models such as the Industrial Internet Reference Architecture (IIRA) [5], the Reference Architecture Model for Industry 4.0 (RAMI 4.0) [2], and communication architectures of distributed automation systems [6]. However the main foundation of the proposed meta-model is based on RAMI 4.0 reference model.

A major task that should be addressed in CPPS is to enable self adaptation, to allow these systems to adjust their behaviour in response to their perception of the environment and the requirements of the system itself, that may be unknown at the design phase. Previous research work have already discussed possible solutions for self adaptable systems. Tauber et al, [7] propose a generic autonomic management framework for self-adaptation based on the autonomic computing approach [3]. Georgas and Tayler [8] provide a knowledge-based approach to develop systems that are able to autonomously adapt in the phase of change. They also extend their work by applying this approach to the development of self-adaptive robotic systems [9]. Kit et al., [10] present an architecture, which takes into account self-adaptation by providing a holistic view to combine the goals of a system, the system's operational model and a realistic communication model. This model allows large-scale simulations of complex CPS, but without considering self-adaptation in the cloud backend and the sensors alike.

Recent projects have also addressed self - adaptation. The FEDerAteS (A Foundation for Engineering Decentralized Self-Adaptive Software Systems) project overall goal is to study and develop a scientific foundation for engineering decentralized self-adaptive systems. The main project results include foundations of self-adaptation, based on which they have provided evidence that external feedback control loops (FCL) improve the design of self-adaptive systems [11], an architecture framework for collective intelligent systems that include humans in the loop [12], and validation of the results in different application domains, such as robotic system [13].

The MORISIA (Models@run.time for Engineering Self-adaptive Software Systems) project provides concepts for engineering self-adaptive software systems, especially the adaptation logic, with runtime models. The project is mainly focused on runtime models for feedback loops by means of models that are causally connected to the running system, reflecting the system's environment, and specifying the adaptation steps (monitoring, analysis, planning, and execute) [14].

The SALTY (Self-Adaptive Very Large Distributed Systems) project has also published results on innovative self-managing software framework at run-time for VLSDS. Nzekwa et al., [15] investigate a model-driven approach for the engineering of FCL whose architecture is based on the Service Component Architecture (SCA) model and argue that the use of a data-oriented model for designing self-adaptive systems significantly increases FCL visibility. Further, they propose a tool approach that enables engineers to design and integrate adaptation mechanisms into software systems [16].

However, none of the above mentioned works addresses explicitly how self-adaptation can be applied to CPPS in an Industry 4.0 application scenario, and how it can be used to improve the trade-off between business process performance and security. Thus, in this paper we develop a trade-off model as a mean to identify the need for flexible and secure communication between edge devices and cloud backend in an Industry 4.0 application scenario and propose self-adaptation as a solution to it.

III. The CPPS meta-model

Since CPPS are in the initial stage of the development, there is an urgent need for a clear description of such systems. To meet such a demand, considering RAMI 4.0, we derive a CPPS meta-model as shown in figure 1. The CPPS meta-model can be adopted and scaled to the needs of the project. As different use cases and viewpoints should be considered and researched, the chosen meta-model allows incremental enhancements that can be proven by different research activities. We design the CPPS meta-model using the ADOxx meta modelling tool and the UML notation. The following modelling objects are used for the presentation of the meta-model: (i) modelling object *class*, a classifier which describes a set of objects that share the same features, constraints or semantics, (ii) relationship *dependency*, a directed relationship that is used to show that some UML elements or a set of elements requires, needs or depends on other model elements for specification or implementation, and (iii) modelling object *aggregation*, a graphical modelling object to visualize levels of the CPPS.

As described in RAMI 4.0 the process axis is vital for the success of describing and planning the real world of Industry 4.0. Processes allow a one-to-one description of the reality and the measurement of, for example, security aspects in a predictive way. Using the CPPS meta-model, is possible to model all relevant entities and their relations to each other on different levels. The CPPS meta-model is composed of five main levels. The business or governance level comprises objects like processes, products, IT-services, requirements as well as contracts/SLA agreements. These objects are interlinked with the objects of the architecture and services level with the entities application, application group, service, application component, interface and data. On the technology level the objects technology, technology package, infrastructure component, database and network element can be found. An additional level for the risk architecture with the objects risk and safety/security prevention is defined. The organizational view and its interrelations to the CPPS is demonstrated within the level of the organization with the objects role, actor, organizational unit and site. Each of these objects has a set of attributes that describes the features of the underlying CPPS. In the context of this research work we use the CPPS meta-model as a tool to describe an end-to-end communication use case, explained in section IV-A, and to identify and document in a structured methodological and lightweight way the self-adaptation points.

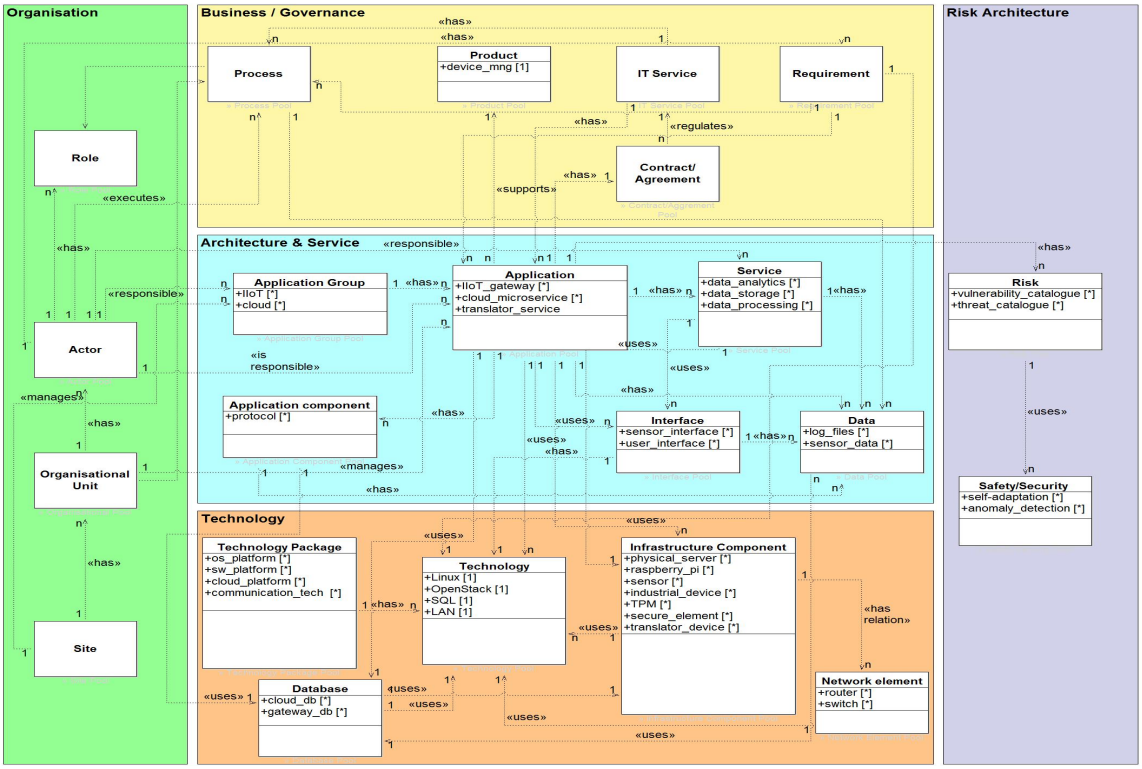

Fig. 1. CPPS meta-model composed of five main levels and the corresponding objects, addressing the RAMI 4.0 axes, used to describe an end-to-end communication use case for an Industry 4.0 application scenario

IV. Applying Self-Adaptation to Industry 4.0 - A Use Case

In this section, we introduce an end-to-end communication use case for an Industry 4.0 application scenario and discuss some related security and performance considerations. Based on these considerations, we develop a trade-off model, considering traffic characteristics, such as offered load, and delivery time, to evaluate the effectiveness of such systems for different security levels of the communication protocol. Additionally, we propose self-adaptation based on autonomic computing approach [3] as a solution to improve the trade-off between performance and security.

A. CPPS end-to-end communication use case

The CPPS are integrated and built on many existing technologies and components such as industrial production environment, including industrial devices equipped with sensors and actuators, industrial IoT (IIoT) components, and backend systems, such as cloud platforms.

For our investigation, we consider a use case from an ongoing research project illustrated in figure 2, which addresses a flexible and secure end-to-end communication in an Industry 4.0 application scenario. We describe the use case in details, including the technologies, to show how the CPPS components can be mapped in the corresponding objects of the meta-model. To provide a smart service, such as device management, data is transmitted between devices, processed and

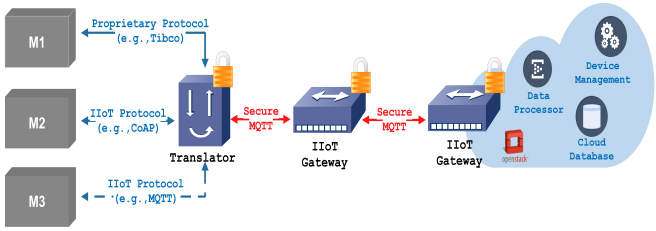

Fig. 2. CPPS end-to-end communication use case for an Industry 4.0 application scenario

stored throughout the network, and sent to private clouds for further processing and analysis. The communication protocol used between the industrial devices, the IIoT components, e.g., the IIoT gateways, and the cloud backend system is the MQTT (Message Queue Telemetry Transport) protocol. MQTT is a lightweight protocol widely used to accommodate constrained devices with low power and bandwidth requirements [17]. In the industrial production environment, the new industrial devices (M2, M3) are already able to communicate using state of the art protocols, such as MQTT or CoAP. However, this is not the case if a legacy device (M1) wants to establish a connection with the IIoT gateway. In this case, a translator device is needed to translate the device protocol into MQTT. Relevant research projects have already proposed solutions for the translation systems, such as Arrowhead protocol translation system [18].

The industrial devices need to be secured, and particularly

the transmitted data needs to be encrypted as this data often contains highly confidential data such as industrial recipes. In principle, the data could be encrypted by simply using available security APIs. Unfortunately, software-based encryption is prone to attacks that reveal the used encryption key. To overcome this issue, we propose to integrate a dedicated hardware, often referred to as "Secure Element", in the translator device. The secure elements provide tamper resistant storages for holding and protecting the key from any kinds of attacks, even including physical access to the device. The industrial devices send the data (e.g., device lifetime data) to an IIoT gateway, which is build on a Raspberry Pi equipped with macchina.io [1], which is an open source software toolkit for quickly building embedded applications that run on Linux-based devices. These gateways are used to distribute data between various industrial devices and the cloud storage, however, they might not be able to store certificates or perform cryptographic calculations. A potential attacker could easily clone such a gateway and insert it into the network. The attacker's device could then manipulate or steal the data sent between industrial devices and the cloud. Therefore other solutions have to be applied to overcome this issue, such as TPMs (Trusted Platform Module) as they provide integrity protection and a root of trust for the IIoT gateway. Furthermore, TPMs provide a standardized interface, which makes it very easy to integrate them in any gateway. The data are than sent to a backend system, in this case an OpenStack[2] cloud platform. The microservices in the cloud backend than analyse and process this data to provide tailor-made individualized services.

The components of this use case are mapped in the CPPS meta-model to give an idea how such a model can be used to describe Industry 4.0 application scenarios and to show the dependences between different levels and objects as shown in figure 1. For example, in the business/governance level the product (sales unit with a certain price that can be either product or service) resulting from the described use case is the device management service. This service is supported by the underlying levels, such as architecture and technology level, where the components of the CPPS are mapped to the corresponding objects of the meta-model. For example, one application supporting device management service is the IIoT gateway, which has the MQTT protocol as an application component. The application component is interconnected with the data object, in this case sensor data. The data object uses a database object, in this case a gateway database, from the technology level, which uses SQL as technology, and Raspberry Pi as infrastructure component.

B. Security and performance considerations

In this subsection we discuss performance and security considerations related to the MQTT protocol, as an asset of the considered end-to-end communication use case.

MQTT over TCP: The MQTT protocol by default relies on TCP (Transmission Control Protocol) as transport protocol, so the connection does not use encrypted communication. In this case security remains a serious concern.

MQTT over TLS session start-up: For a secure end-to-end communication, the MQTT should be used over TLS (Transport Layer Security) protocol instead of plain TCP. TLS is a protocol used to establish an authenticated communication channel and can also accomodate other application layer protocols if needed. However, this security improvement comes at a cost in terms of the communication overhead. The overhead of a TLS connection can be divided up into: (i) the session start-up overhead (OH_{TLS_ss}), required for establishing the TLS handshake, and (ii) the per-packet overhead (OH_{TLS_pp}), computational overhead of cryptographic functions. The TLS handshake is initiated by the MQTT client to start a secure communication session by using asymmetric algorithms, such as RSA, for server identification. The TLS handshake typically adds 4-7 kilobytes of communication overhead, whilst the per-packet overhead depends on the ciphersuite used. In this model, we have considered AES-256-CBC-SHA. AES is a symmetric key encryption technique that has a fixed block size of 128 bits and uses different key lengths, 128, 192, and 256 bit, and is used for encrypting application layer specific messages. Although encryption guarantees privacy, to ensure message integrity TLS uses also message digits generations and verification algorithms such as MD5, SHA, etc. For a high level of security, SHA is recommended compared to MD5. Thus, the per-packet overhead of the chosen ciphersuite is approximately 40 bytes, as the packet is protected by a TLS header (5 bytes) and a TLS trailer, which includes the encryption algorithm padding (0-15 bytes) and the MAC tag that comes from the SHA authentication algorithm (20 bytes).

MQTT over TLS session resume: The MQTT client needs to establish a connection once per session, thus the TLS session once established can be resumed. Once connected to the MQTT broker, the client can send and receive packets without any additional handshake overhead, reducing it to the session resume overhead (OH_{TLS_sr}), approximately 330 bytes. The per-packet overhead remains the same.

MQTT over TLS protected by TPM: However as discussed above, the security of the MQTT over TLS can be further improved by integrating a TPM in the IIoT gateway that acts as a trust-boundary for secure data exchange, but adds an additional overhead (OH_{TPM}).

C. The trade-off model

Based on the security and performance considerations, we can argue that by using ***MQTT over TLS protected by TPM*** the level of security in CPPS end-to-end communication can be significantly improved, whereas the level of business process performance depends on whether TLS session resume is used or not. The TLS session resume offers a better performance compared with TLS session start up but has a major limitation, the servers are responsible for remembering negotiated TLS sessions only for a given period of time, known as session interval. To visualize the effect of session interval in effectiveness and security level, we develop a simplified trade-off

[1]https://macchina.io/

[2]https://www.openstack.org/

model. In the TLS protocol standard [19], an upper limit of 24 hours is suggested for session interval, since an attacker who obtains a master secret may be able to impersonate the compromised party until the corresponding session interval is retired. Thus, we evaluate the effectiveness and security level for different number of sessions (s) within a given period of time, meaning that the bigger the number of sessions (s) the shorter the session interval. First, we define the delivery time of N packets when using ***MQTT over TCP*** (d_t_{TCP}), as the ratio between the packet size (p_s) and the offered load. In the same way we define the delivery time of N packets when using ***MQTT over TLS protected by TPM*** ($d\_t_{TLS+TPM}$) for different number of sessions (s), where $N \geq s$, which depends on the packet size (p_s) increased by the additional overhead coming as a result of security mechanisms and offered load. In this case n is the number of packets delivered per TLS session start-up.

$$d\_t_{TCP} = \frac{N * p\_s}{offered\_load} \tag{1}$$

$$d\_t_{TLS+TPM}(s) = \frac{n * (p\_s + \sum OH_1) + (N - n) * (p\_s + \sum OH_2)}{offered\_load} \tag{2}$$

where,

$$\sum OH_1 = OH_{TLS\_ss} + OH_{TLS\_pp} + OH_{TPM} \tag{3}$$

$$\sum OH_2 = OH_{TLS\_sr} + OH_{TLS\_pp} \tag{4}$$

In terms of business process performance, we define effectiveness as the ratio between the expected delivery time, delivery time when using MQTT over TCP, and the achieved delivery time for different number of TLS sessions within a given period of time.

$$effectiveness(s) = \frac{d\_t_{TCP}}{d\_t_{TLS+TPM}(s)} * 100\% \tag{5}$$

In terms of security, the shorter the session interval, the lesser the time that an attacker gets to guess the session ID [19]. To visualize this we define the security level as reverse of the effectiveness level.

$$security(i+1) = effectiveness(s - i) \tag{6}$$

for i=0,1,2,...,s-1

For the trade-off model analysis we consider the following as a representative example: (i) number of delivered packets $N = 50$, (ii) packet size $p\_s = 500bytes$, (iii) offered load $offered\_load = 0.1; 0.5; 1; 5; 10Mbps$, and (iv) number of sessions $s = 1, 2, ..., s_{max}; s_{max} = 50$.

Figure 3 shows the effectiveness and security level using different number of TLS sessions in a given period of time, thus different TLS session intervals. When using MQTT over TLS protected by TPM with a statically configured session interval the following unsatisfactory situations can be identified depending on the system requirements: (i) a low effectiveness level for short session intervals because the TLS handshake will be repeated for each session start up, and (ii) a low security level for long session intervals because the attacker will get more time to guess the session ID.

Thus, we propose self-adaptation as a solution to improve the trade-off between effectiveness and security. The integration of autonomic managers in the corresponding CPPS components makes possible to adapt the session interval based on the system requirements.

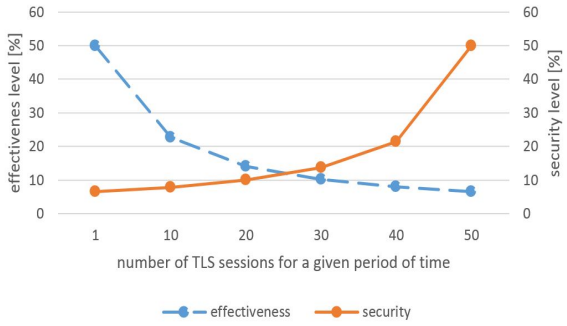

Fig. 3. Effectiveness and security level for different number of TLS sessions in a given period of time

V. Self-adaptation

From the trade-off model result we can argue that maximizing the security and business process performance at the same time is not possible and an efficient balance between these two parameters has to be achieved. Thus, we propose self-adaptation as a solution to improve the trade-off between business process performance and security. As illustarted in figure 4, a self-adaptable system consists of two parts: the target system, which supports functions that are specific to the domain, and the autonomic manager. A well-known approach to develop an autonomic manager is through feedback loops [3], consisting of four phases: (i) monitor phase, during which target system specific events are generated, (ii) analyze phase, during which metrics are extracted from the generated events in order to represent the current situation of the target system, (iii) plan phase, during which policies driven by the metrics are defined, and (iv) execute phase, during which the planned actions are carried out. These four phases are supported by a knowledge base that provides a level of abstraction of the activity, target system, environment, and requirements. To allow the interaction between the target system and the autonomic manager two additional components are included, the sensors and the effectors. Sensors are used to gather information from the target system, in order to support the tasks performed by the autonomic manager. Likewise, the effectors are used to execute the action decided by the autonomic manager to the target system. In our previous work [7], we have proposed GAMF (Generic Autonomic Management Framework) as a solution that allows programmers to develop an autonomic manager for any target system without having to (re)implement generic control mechanisms. We intend to extend our previous work to develop autonomic managers for CPPS components and to investigate how to make them interact with each other. To identify and document in a structured methodological and lightweight way the self-

adaptation points we use the CPPS meta-model, as illustrated in figure 4, where self-adaptation approach is a model of the security object in the risk architecture level of the meta-model.

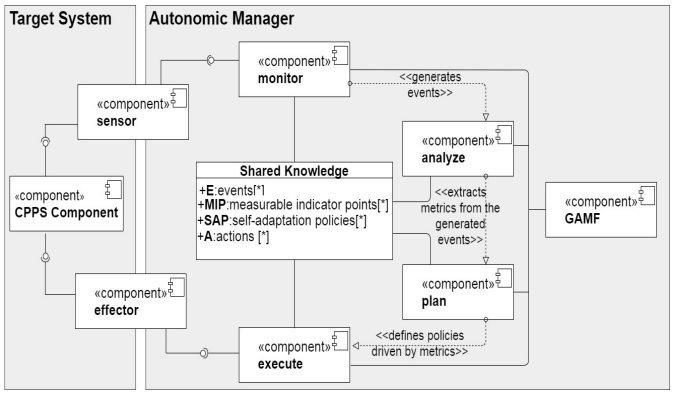

Fig. 4. Self-adaptation components documented in the CPPS meta-model

Considering our use case, in order to provide self adaptation for business process performance and security concerns, first it is necessary to *monitor* the target system. Thus, *sensors* are used to collect data in the IIoT gateway regarding delivery time and session interval. The sensors must have access to read this data from the MQTT protocol. Another target system which is of interest to be considered for monitoring in such a scenario is the translator system. It is a crucial point of the use case, since it makes possible the communication of different components that use different messaging protocols. Second, it is important to *analyze* the target system behaviour based on the shared knowledge. For example, it is important to analyze the effectiveness and security level, defined as Measurable Indicator Points (MIP) in the model. Third, a *plan* needs to be defined using self-adaptation policies (SAP) to mitigate the detected issues. A possible SAP is to adapt the session interval, for example by extending the session interval when high performance is required, or by shortening the session interval when high security is required. These actions (A) are *executed* by the *effectors* to the target system.

VI. Conclusion

In this position paper we have derived a CPPS meta-model from RAMI 4.0 and we have used it as a tool to describe an end-to-end communication use case. We have developed a business process performance and security trade-off model, which shows that by adapting system parameters, such as the TLS session interval of the communication protocol, the trade-off between effectiveness and security can be improved. Thus, we have proposed to use self-adaptation based on autonomic management approach to allow such system to adapt their behaviour in response to changes of the environment and the system requirements, providing a flexible and secure end-to-end communication. Defining and categorizing events, metrics, adaptation policies and effectors relevant for self-adaptable CPPS in the Industry 4.0 domain will be one of the main tasks in our future work.

Acknowledgment

The work has been performed in the project Power Semiconductor and Electronics Manufacturing 4.0 (SemI40), under grant agreement No 692466.

References

[1] R. Drath and A. Horch, "Industrie 4.0: Hit or hype?[industry forum]," *IEEE industrial electronics magazine*, vol. 8, no. 2, pp. 56–58, 2014.

[2] M. Hankel and B. Rexroth, "The reference architectural model industrie 4.0 (rami 4.0)," *ZVEI, April*, 2015.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[4] P. I. 4.0, "Digitization of industrie - plattform industrie 4.0," *Federal Ministry for Economic Affairs and Energy (BMWi), Berlin*, April 2016.

[5] I. I. Consortium *et al.*, "Industrial internet reference architecture," *Industrial Internet Consortium, Tech. Rep., June*, 2015.

[6] T. Hadlich, C. Diedrich, K. Eckert, T. Frank, A. Fay, and B. Vogel-Heuser, "Common communication model for distributed automation systems," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*. IEEE, 2011, pp. 131–136.

[7] M. Tauber, G. Kirby, and A. Dearle, "Self-adaptation applied to peer-set maintenance in chord via a generic autonomic management framework," in *Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on*. IEEE, 2010, pp. 9–16.

[8] J. C. Georgas and R. N. Taylor, "Towards a knowledge-based approach to architectural adaptation management," in *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*. ACM, 2004, pp. 59–63.

[9] J. Georgas and R. Taylor, "Policy-based self-adaptive architectures: a feasibility study in the robotics domain," in *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. ACM, 2008, pp. 105–112.

[10] M. Kit, I. Gerostathopoulos, T. Bures, P. Hnetynka, and F. Plasil, "An architecture framework for experimentations with self-adaptive cyber-physical systems," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2015, pp. 93–96.

[11] J. Musil, A. Musil, D. Weyns, and S. Biffl, "An architecture framework for collective intelligence systems," in *Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on*. IEEE, 2015, pp. 21–30.

[12] M. U. Iftikhar and D. Weyns, "Activforms: Active formal models for self-adaptation," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2014, pp. 125–134.

[13] S. Shevtsov, M. U. Iftikhar, and D. Weyns, "Simca vs activforms: comparing control-and architecture-based adaptation on the tas exemplar," in *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*. ACM, 2015, pp. 1–8.

[14] T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker, "Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems," in *Proceedings of the 6th IEEE/ACM International Conference on Autonomic Computing and Communications (ICAC 2009), Barcelona, Spain*. ACM, June 2009, pp. 67–68. [Online]. Available: http://dx.doi.org/10.1145/1555228.1555249

[15] R. Nzekwa, R. Rouvoy, and L. Seinturier, "Modelling feedback control loops for self-adaptive systems," in *Third International DisCoTec Workshop on Context-Aware Adaptation Mechanisms for Pervasive and Ubiquitous Services*, 2010, pp. 1–6.

[16] F. Křikava, P. Collet, and R. B. France, "Actress: Domain-specific modeling of self-adaptive software architectures," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 2014.

[17] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[18] H. Derhamy, J. Eliasson, and J. Delsing, "Iot interoperability-on-demand and low latency transparent multi-protocol translator," *IEEE Internet of Things Journal*, 2016.

[19] E. Rescorla, "The transport layer security (tls) protocol version 1.1," *Transport*, 2006.