Layered Modeling Approach for Distributed Situation Recognition in Smart Environments

Mathias Mormul, Pascal Hirmer, Matthias Wieland, and Bernhard Mitschang

Institute of Parallel and Distributed Systems
University of Stuttgart, Universitätsstr. 38, D-70569, Germany
email: firstname.lastname@ipvs.uni-stuttgart.de

***Abstract*—In the last decade, multiple new paradigms changed the way IT works. Cloud and Edge Computing led to new approaches, such as Smart Factories and Smart Cities, but also to new challenges and opportunities. One of those challenges is the recognition of situations, e.g., machine failures. Especially in the domain of industrial manufacturing, several requirements have to be met in order to deliver a reliable and efficient situation recognition. One of these requirements is distribution. The main contribution of this paper is a layered modeling approach for modeling situations to enable the distribution of situation recognition based on distribution patterns that are introduced in this paper.**

Keywords–Industry 4.0; Edge Computing; Smart Factories; Smart Homes; Situation Recognition; Distribution Pattern.

I. Introduction

In recent years, Industry 4.0 (I4.0), the digitization of the manufacturing industry, emerges as a new paradigm enabling approaches, such as Smart Factories [1]. In I4.0, devices equipped with sensors and actuators communicate with each other through uniform network addressing schemes to reach common goals [2][3]. Oftentimes, this goal is situation recognition, which enables monitoring of I4.0 environments and, consequently, the timely reaction to occurring situations. For example, the occurrence of a traffic accident in a Smart City, recognized by sensors of a vehicle, could lead to an adaptation of traffic lights to control the affected traffic.

Situations are recognized through the aggregation of context data which, in I4.0, is usually provided by sensors. In current approaches, such as the one we introduced in our previous work [4][5], situations are recognized in a monolithic IT infrastructure in the cloud. Consequently, involved context data needs to be shipped to the processing infrastructure in order to recognize situations. However, especially in domains where efficiency is of vital importance, e.g., Smart Factories, this approach is not feasible. In order to fulfill important requirements, such as low network latency and fast response times, the situation recognition needs to be conducted as close to the context data sources as possible and, therefore, in a distributed manner. Processing data close to the sources is commonly known as Edge Computing [6].

In this paper, we introduce an approach to enable a distributed situation recognition. By doing so, we introduce so-called *distribution patterns*. These patterns represent common ways to distribute the recognition of situations, i.e., exclusively in the *edge*, in on-premise or off-premise cloud infrastructures, or based on a hybrid approach. We provide a layered approach for modeling and executing the situation recognition based on these distribution patterns. Our approach builds on a set of requirements we derive from a use case scenario in the manufacturing domain. We validate the approach by applying it to our previous non-distributed situation recognition [4][5] that is based on the modeling and execution of so-called Situation Templates [7].

The remainder of this paper is structured as follows: Section II describes related work and foundational background. In Section III, we introduce a motivating scenario, which is used to derive requirements for our approach. In Section IV, we present the main contribution of our paper. Finally, Section V concludes the paper and gives an outlook to future work.

II. Related Work and Background

In this section, we describe related work, as well as foundational concepts of our previous work that are necessary to comprehend our approach.

A. Related Work

In related work, approaches exist for distributed situation recognition using ontologies, e.g., Fang et al. [8]. These approaches do not achieve the latency required in real-time critical scenarios, such as Industry 4.0 [1], due to time-consuming reasoning. The goal of our approach is to offer a low latency for distributed situation recognition in the range of milliseconds. Many approaches using ontologies are in the range of seconds to minutes, even without distribution [9][10]. Using machine learning leads to similar limitations regarding latency [11].

In the area of distributed Complex Event Processing (CEP), Schilling et al. [12] aim at integrating different CEP systems using a common meta language. This allows to use different CEP systems and integrate the results. This could be beneficial for our distribution because we would not be limited to one execution environment. However, in [12], the queries have to be hand-written and distributed. This is difficult, especially for domain experts, e.g., in Industry 4.0, who do not have computer science knowledge. In our approach, we provide an abstraction by Situation Templates that can be modeled using a graphical user interface. Furthermore, the users are supported in splitting up the template and in the distribution decision. Other approaches in distributed CEP, e.g., by Schultz-Moller et al. [13], follow the concept of automatic query rewriting. Here, CEP queries are split up using automated rewriting and are distributed on different operators based on a cost model, which is mostly based on CPU usage in the different nodes. In our approach, we want to support the user to select the desired

 ISBN: 978-1-61208-653-8

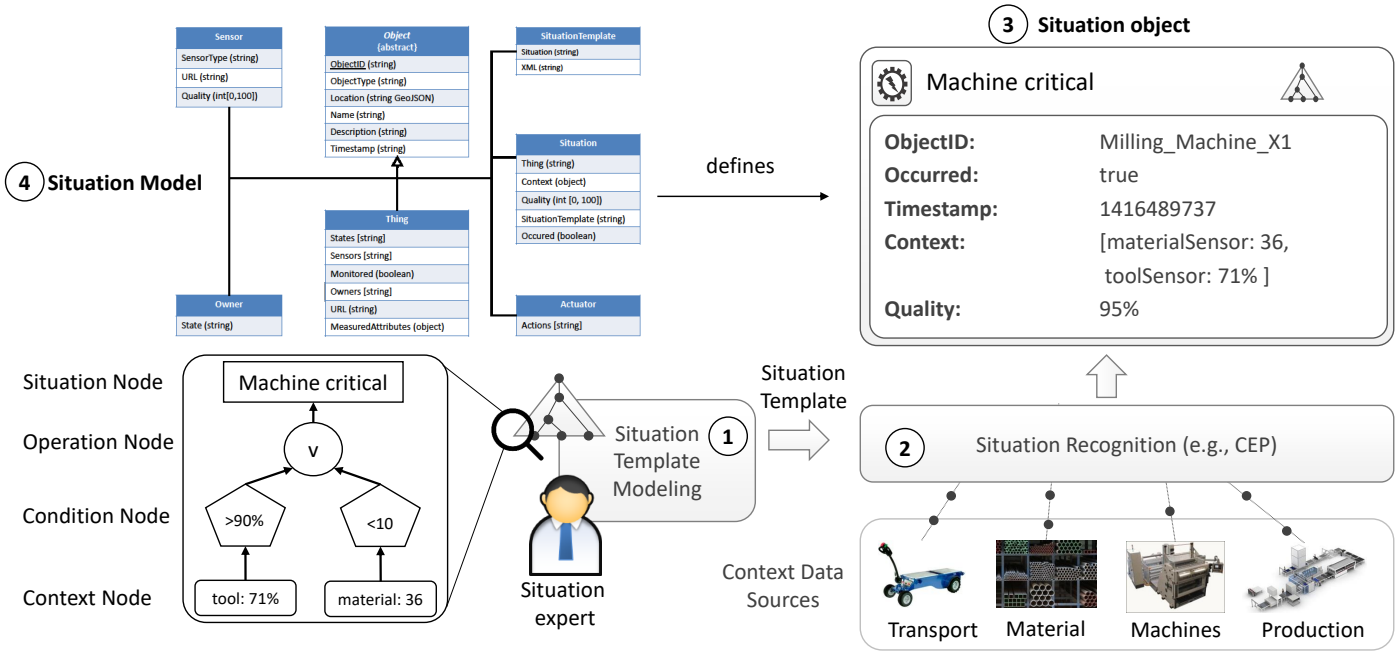

Figure 1. Previous approach for situation recognition

distribution type. Since there are many aspects, such as data protection or security, that play a role in distributing the CEP queries correctly, this only can be known by a responsible user.

Furthermore, approaches exist that enable a massive distribution of sensors, e.g., by Laerhoven and Gellersen [14] in cloths, to detect activities of the person wearing the cloth. This is similar to detecting the situation in the edge, but there is no concept presented in [14] to integrate the activities with other activities from different edges or create a global situation involving different locations.

B. Background

In this section, we describe our previous work. Our first approach for situation recognition, this paper builds on, is depicted in Figure 1. This approach is a result of the issues of related work, as discussed in the previous section.

An important fundamental concept are Situation Templates (ST), introduced by Häussermann et al. [7]. We adapted the STs in [5] to model and recognize situations. Situation Templates (see Figure 1 on the buttom left) consist of *context, condition* and *operation* nodes, which are used to model specific situations. Context nodes describe the input for the situation recognition, i.e., the context data, based on the definition of Dey et al. [15]. Context nodes are connected to condition nodes, which define the conditions that have to apply for a situation to be valid. Operation nodes combine condition and operation nodes and represent the logical operators *AND, OR*, or *XOR*. Operation nodes are used to aggregate all condition nodes of the ST into a single node, the situation node.

After modeling of a ST (Figure 1, Step 1), we developed a transformation into an executable representation (not depicted) was realized using CEP or light-weight execution languages, such as Node-RED. The advantage of this transformation is that it provides a flexible means to recognize situations. These transformations can be found in [16][17]. Consequently, we are not limited to specific engines or data formats. Once the transformation is done, the executable Situation Template is handed over to the corresponding execution engine.

On execution (Figure 1, Step 2), context data originating from the context sources is validated against the conditions defined by the Situation Template, for example, through pattern recognition in CEP. On each validation, we create a so-called situation object [18], defining whether the situation occurred and containing the involved context data (Figure 1, Step 3). We created a Situation Model [18](Figure 1, Step 4) to define the attributes of those situation objects. This leads to a better understanding of how context data led to the situation.

This previous approach for situation recognition works well, however, there are still some limitations this paper aims to solve. First, the current approach was built to monitor single things (e.g., devices). However, as the complexity of nowadays IT infrastructure arises, means need to be enabled to monitor more than one thing using the introduced Situation Templates. Furthermore, currently, the STs are executed in a monolithic manner because in former scenarios, distribution was not necessary. In current approaches, e.g., involving Industrie 4.0, however, this is necessary. In this paper, we aim for enhancing our approach in order to be more fitting to recent scenarios.

III. Scenario and Requirements

In this section, we introduce a practical motivating scenario from the I4.0 domain, which is used throughout the paper to explain our approach. In the scenario, depicted in Figure 2, a specific part of the supply chain of a production company should be monitored. As depicted, there are several entities involved: (i) production machines, assembling products based on parts, and (ii) trucks, delivering the parts to be assembled. The monitoring should detect critical situations that could occur, for example, the failure of at least one of the machines, or a delivery delay of parts, e.g., caused by issues with

 ISBN: 978-1-61208-653-8

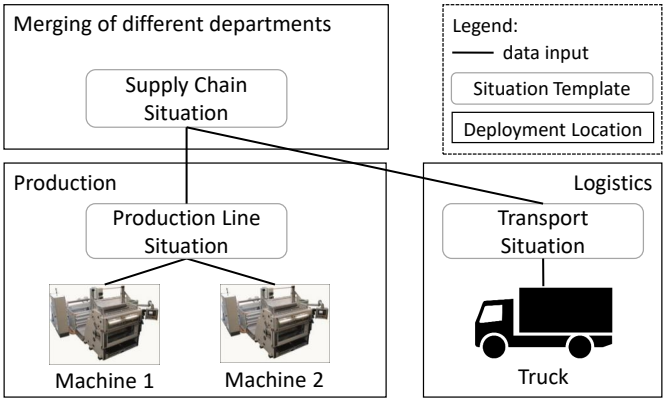

Figure 2. Motivating scenario for distributed situation recognition

trucks or with the supplier. Situations that could occur are: (i) *Production Line Situation*, indicating that one of the production machines is in an erroneous state, (ii) *Transport Situation*, indicating a problem with the truck, and (iii) *Supply Chain Situation*, indicating a problem with either the production line or the truck.

When applying our previous approach, described in Section II, to this scenario, new requirements arise that need to be coped with. We divide these requirements into ones that concern the modeling of STs and ones that concern the execution of the situation recognition. We derived 8 requirements R_1 to R_7 for this scenario.

Modeling Requirements
Three requirements focus on the modeling of the situation recognition using STs.

- R_1 **- More powerful Situation Templates:** With our previous approach (cf. Section II-B), single machines can be monitored in an efficient way as evaluated in [5], which was sufficient for previous scenarios. However, in recent scenarios involving Industry 4.0, the requirements are increasing. In our motivating scenario, it is important to model dependencies between multiple entities within a single ST, e.g., to recognize the *Production Line Situation*.
- R_2 **- Low modeling complexity:** In our previous approach, modeling STs involving a lot of context data has led to a cumbersome task and, consequently, to a high complexity and error-prone modeling. To cope with this issue, a new modeling approach is required that enables the reutilization of already existing STs to lower the modeling complexity of new STs.
- R_3 **- Domain-independence:** A consequence of the issue described in R_2 is domain-dependence. Large STs usually consist of a wide range of context data sources, e.g., Trucks or the Production Line of our scenario. However, these context data sources require domain experts of these specific areas. Consequently, STs need to be modeled by these experts together. This leads to high costs due to the communication overhead. Hence, our goal is to enable domain-independence for ST modeling.

Execution Requirements

- R_4 **- Low latency:** In many domains, latency plays a crucial role. Especially in Smart Factory environments, the industrial automation layer has strong requirements regarding end-to-end latency up to 1 ms or even lower [19]. Therefore, the execution of the situation recognition needs to adapt to those requirements, so that critical situations like machine failures can be recognized in a timely manner.
- R_5 **- Low network traffic:** In modern scenarios, large amounts of data are produced that need to be stored and processed in order to recognize situations. For example, an autonomous car produces about 35 GB/hour of data [20]. In comparison, Budomo et al. [21] conducted a drive test and recorded a maximum and minimum upload speed of 30Mbps (13.5 GB/hour) and 3.5Mbps (1.58 GB/hour), respectively, using the current mobile communication standard LTE-A. Therefore, transferring all data of an autonomous car to the cloud is currently impossible. Consequently, reducing the network traffic is an important issue when recognizing situations.
- R_6 **- Data security & privacy:** Especially the processing of company data needs to be secure and, furthermore, privacy needs to be ensured. However, especially when processing data in the Public Cloud, companies need to trust the Cloud providers that they provide the security they require. Alternatively, companies can keep their data close, i.e., in a trusted environment.
- R_7 **- Cross-company situation recognition:** Modern products and its components are rarely built completely by one company. Therefore, most actual scenarios are very complex, involve multiple companies, and require a cross-company situation recognition. Our motivating scenario in Figure 2 can be regarded as such an example, in which a manufacturing company cooperates with a logistics company. For example, a delayed delivery caused by a failure of the truck must be communicated to the manufacturing company. Consequently, our situation recognition approach needs to enable a cross-company situation recognition.

IV. Distribution of Situation Recognition

In our previous work, we already solved challenges regarding sensor registration and management [22], efficient solutions for a situation recognition [17], and the management of recognized situation [18]. Now, we concentrate on extending our previous approach by introducing a distribution of the situation recognition to fulfill the above-mentioned requirements R_1-R_7. For this, we first present (i) the modeling improvements for our approach to support the distribution we aim for. On this basis, we present (ii) the execution improvements to enable the distribution based on three distribution patterns including a decision support for each of those patterns.

The distributed situation recognition was implemented based on the existing prototype of our previous work, introduced in [17][18] by following adaptations: (i) the modeling for STs was extended, (ii) the transformation was enhanced to accept multiple things, and (iii) the communication between the distributed locations is enabled by messaging systems.

 ISBN: 978-1-61208-653-8

A. Modeling Improvements

In the following, we present the improvements regarding the modeling of STs to fulfill the requirements R_1-R_3. The extension of the STs, i.e. its schema, comprises (i) the modeling of multiple things within a single ST, and (ii) a layered modeling by reutilizing already modeled STs. These extensions are depicted in Figure 3. Requirement R_1 describes the need for the modeling of more powerful situations, e.g., *Production Line critical*. However, a production line itself does not contain any sensors but rather describes the coherence and arrangement of multiple machines. Therefore, to model a situation describing the production line, we need to model all machines of the production line into a single ST. By extending the Situation Template Schema to allow the modeling of multiple things, therefore, we fulfill requirement R_1.

It is obvious that from a certain amount of things in a single ST and each thing having multiple sensors, the complexity of modeling such a ST is becoming a problem. An excessive complexity restricts the usability of our modeling approach, hence, the reduction of the modeling complexity is required (cf. R_2). To cope with the increasing complexity of STs, we introduce the layered modeling approach. Instead of modeling everything within a single ST, we use situations as context input for further situation recognition. Thereby, we implicitly reuse already modeled STs. These situations can be divided into three classes: *local situations*, *hybrid situations*, and *global situations*. This classification is based on the context input of the respective situations and describes the hierarchy of situations. Local situations only receive context input by one or multiple things. Hybrid situations receive at least one local situation as input and context input of at least one thing. Global situations receive at least two situations, local or/and hybrid, as input. An equivalent modeling of the situation *Production Line critical* using the layered modeling approach is shown in Figure 3 (right side). Based on this comparison, we show the benefits of this approach:

- **Reusability:** By using situations as input, we reutilize existing STs. When modeling a global situation, we only need to model the relation between the already modeled local/hybrid situations similar to putting together building blocks. A further advantage is that the local/hybrid STs possibly were already used and tested for correctness, which lowers the error-proneness for modeling global situations.
- **Reduce complexity:** The reusability directly leads to less complex STs, since the modeling is based on the Divide and Conquer paradigm. By using the layered modeling approach, we fulfill the requirement R_2.
- **Distribution:** Since we do not have one single and complex ST, but instead, multiple smaller ones, we already have a beneficial starting point for the distribution of the situation recognition as we can simply execute the different STs at different locations.
- **Support for specific domains:** Having multiple things within a single ST could lead to the problem that knowledge from different domains is required. For example, motivating scenario contains three domains - manufacturing, logistics and their dependencies. Using the layered modeling approach, different domains can model STs independently. Requirement R_3 is fulfilled.

As a result, by introducing an extended Situation Template Schema to enable the modeling of multiple things within a single ST and the layered modeling approach, we fulfill all modeling requirements R_1-R_3.

B. Execution Improvements

The modeling improvements we presented in the last section serve as the foundation for the distribution of the situation recognition. As mentioned above, in our previous approach, the situation recognition was executed centralized in the cloud. Hence, all context data was sent to this cloud and was used as input for the situation recognition. However, lately, the term *Edge Computing* gains more and more attention. Shi et al. [6] define *the edge* as *"any computing and network resources along the path between data sources and cloud data centers"*. Therefore, in our context, Edge Computing refers to the processing of context data close to the data sources.

By introducing Edge Computing to our approach, a distribution of the situation recognition to the cloud and the edge can be performed. In the scenario of Figure 2, the distribution of the situation recognition seems obvious. Using the layered modeling approach, we can model the local situations *Production Line Situation* and *Transport Situation* and the global situation *Supply Chain Situation*. The situation recognition for the local situation is executed at the edge, i.e., locally in the factory or truck, respectively. The global situation is executed in the cloud and receives the local situations as input. However, based on the execution requirements R_4-R_7, this distribution might not always be ideal.

Therefore, in the following, we present the execution improvements resulting from the distribution of the situation recognition. First, we present the concept of context stripping and its benefits. Afterwards, we introduce three distribution patterns and a decision support for choosing the most suitable distribution pattern for a certain scenario.

1) Context Stripping: As presented in Section II-B, when a situation recognition is executed, situation objects are created that are defined by the Situation Model [18]. This situation object contains all context data that were used for the evaluation of this specific situation. In [18], we approximated the data volume of situation objects based on the amount of used context data. The results showed that the appended context data presents the majority of the data size of a situation object. Now, when using the layered modeling approach, we may use local situations that we recognized at the edge as input for the recognition of global situations in the cloud. That causes us to send all context data to the cloud again within the situation object. However, based on the scenario, we might not be interested in the context data of a situation object but only if the local situation occurred or not, so we can evaluate the global situation. Therefore, we introduce the concept of *context stripping*. By using context stripping, the context used for the situation recognition is not sent within the situation object. It only contains the most vital data for a further situation recognition in the cloud. Therefore, content-wise, a local situation only contains a boolean value, which describes if the local situation occurred or not and the required meta data for further processing.

This leads to a trade-off the user has to decide based on his requirements. By using context stripping, the data size of a situation object can be strongly reduced. However, the context

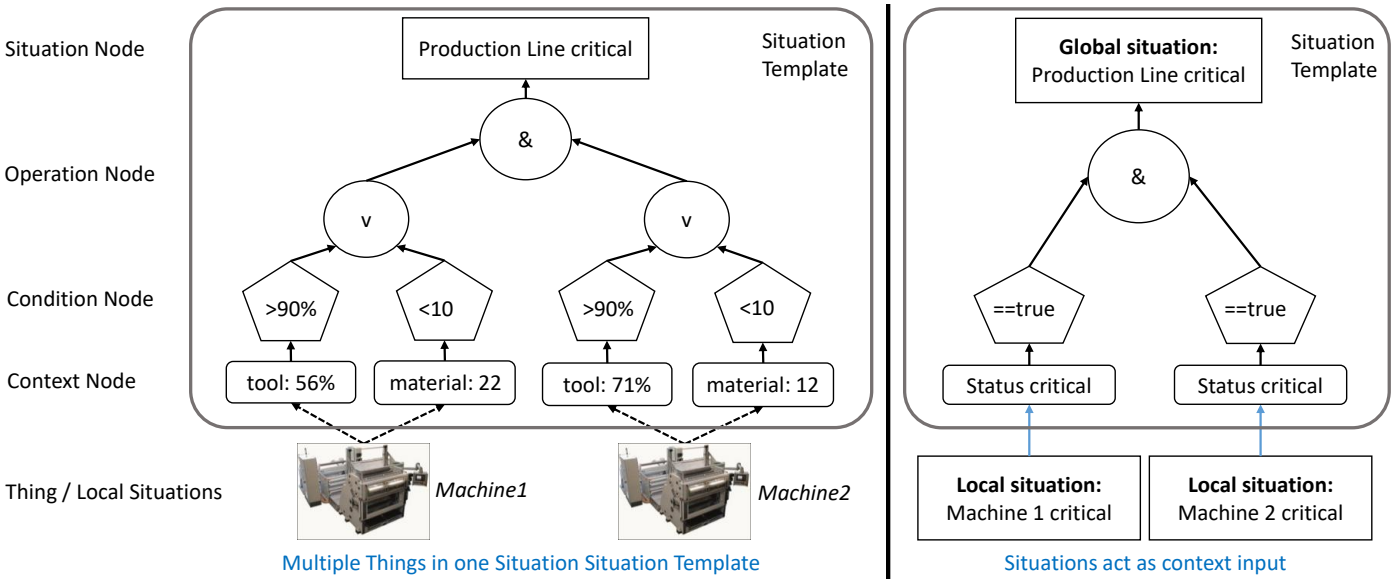

Figure 3. Modeling improvements for STs (legend see Figure 4)

data that led to the evaluation of a specific situation object is discarded after processing. In our first approach, we explicitly wanted to store the context data within situation objects for a detailed historization of situations. This historization, for example, can be used afterwards for a root cause analysis of detected situations based on the involved context data.

2) Distribution Patterns: As mentioned above, the distribution of the situation recognition is dependent on the execution requirements R_4-R_7. Therefore, a general solution for the distribution of the situation recognition is not possible. Instead, we introduce three different distribution patterns, depicted in Figure 4 based on the scenario shown in Figure 2. The *Type I* distribution pattern describes our previous approach. All context data, i.e., in this scenario, context data from a truck and two machines, is sent to the cloud. The situation recognition is executed in the cloud and all context data is available. In contrast, the *Type II* pattern describes the execution of the situation recognition at the edge, close to the data sources. In this case, it is often impossible to gather all context data from all sources, e.g., from the truck, since it is not part of the local network of the factory, where the machines are located. Therefore, only parts of the situation recognition may be executed at the edge. The *Type III* pattern is a hybrid solution based on both the *Type I* and *Type II* pattern and enables the execution of situation recognition at the edge, which results in local situations (i.e., *Production Line* and *Transport*) and the execution of situation recognition in the cloud, where the local situations are used to evaluate the global situation.

In the following, the different distribution patterns are described in more detail with regard to the execution requirements R_4-R_7. Each pattern comprises advantages for certain use cases and might not fulfill every execution requirement. Additionally, the presented distribution patterns are applicable to the distribution of data processing in general.

3) Type-I: Cloud-only (Figure 4, left): Despite many advantages of Edge Computing, the Type-I distribution pattern still is a viable option. Introducing Edge Computing is no trivial task and comprises multiple challenges [6]. Companies with low IT experience or no IT department benefit from outsourcing IT infrastructure and expertise to third-party cloud providers. This oftentimes is the case for SMEs, which then can solely focus on their products and the pay-as-you-go model provides a cost-effective and scalable infrastructure.

- R_4 **- Low latency:** Currently, when using an off-premise cloud, the requirement of 1 ms is already violated by the network latency itself. Therefore, requirement R_4 cannot be fulfilled.
- R_5 **- Low network traffic:** Since all context data must be sent to the cloud first, network traffic cannot be reduced. Requirement R_5 is not fulfilled.
- R_6 **- Data security & privacy:** Since all context data is sent to the cloud, new security risks are introduced. Furthermore, company policies might prohibit sending sensitive or personal context data to the cloud. Therefore, requirement R_6 is not fulfilled.
- R_7 **- Cross company situation recognition:** Since all data is available in the cloud, companies can work together to execute a collaborative situation recognition. Requirement R_7 is fulfilled.

As shown, the Type-I pattern does not fulfill most requirements. Still, in non-critical scenarios where high latency is acceptable, the network traffic is low or fluctuating and the data is allowed to be sent to the cloud by the companies' policies or government regulations, the Type-I pattern is a sensible option.

4) Type-II: Edge-only (Figure 4, middle): In comparison, the Type-II distribution pattern describes the execution of the whole situation recognition at the edge. As already mentioned, this is only possible if all context data is available at the edge. Therefore, the situation recognition of local situations is best-suited for an edge-only execution.

- R_4 **- Low latency:** Yi et al. [23] show that latency can be reduced by 82% by moving an application to the edge of the network. As the situation recognition is

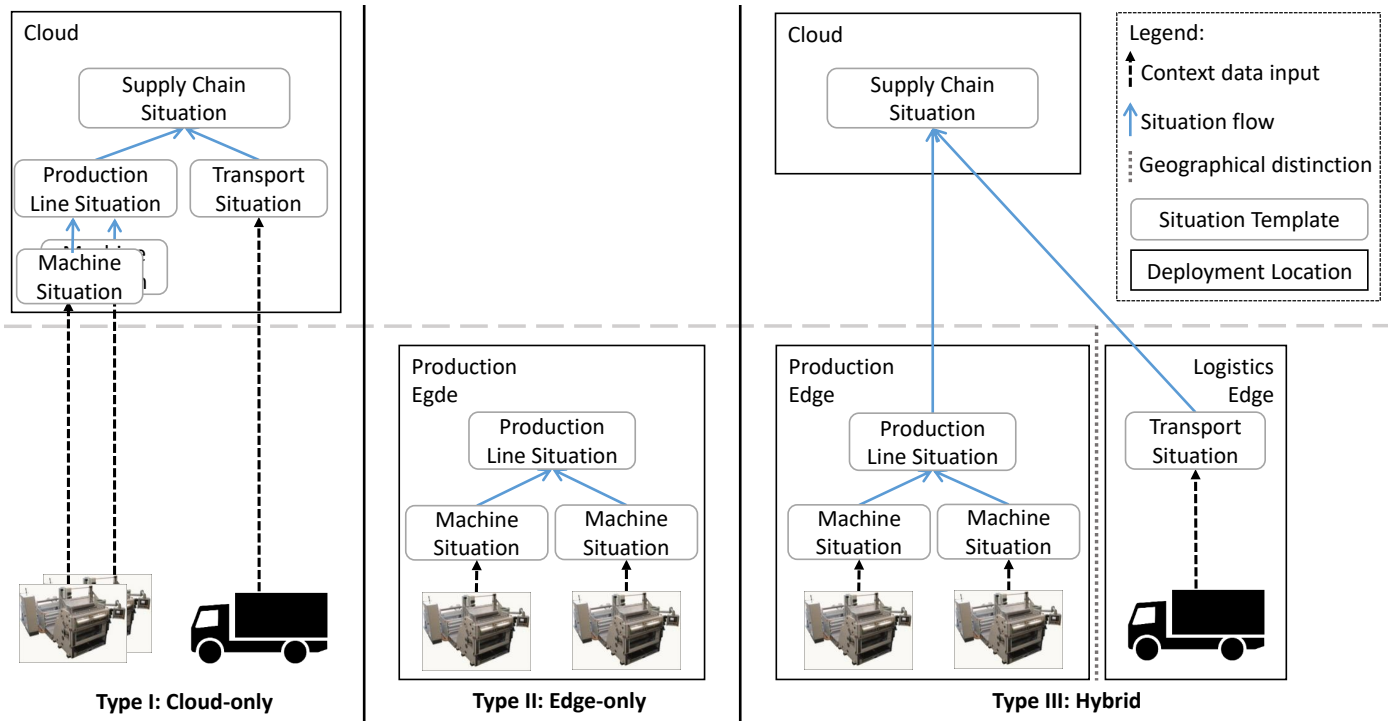

Figure 4. Distribution patterns

executed as close as possible to the data sources the requirement R_4 is fulfilled. With an execution time of 3ms for our situation recognition [17], the overall latency is kept comparably low.

- R_5 **- Low network traffic:** No context data is sent to the cloud, therefore, network traffic stays low and requirement R_5 is fulfilled.
- R_6 **- Data security & privacy:** One of the main concerns regarding the adoption of Cloud Computing still is security, especially in companies with few experience with Cloud Computing. Security and privacy of data is increased, since all context data and situations remain at the edge, i.e., a local network controlled by its company. Requirement R_6 is fulfilled.
- R_7 **- Cross company situation recognition:** In general, the data sources of different companies are geographically distributed and not in the same local network. Therefore, a cross company situation recognition is not possible. Requirement R_7 is not fulfilled.

Most requirements are fulfilled. However, more complex scenarios (cf. Figure 2) cannot be mapped to this pattern because of geographically distributed data sources. Therefore, the Type-II distribution pattern is best suited for company-internal situation recognition that fulfills critical requirements like latency and security. Especially in mobile environments, e.g., an autonomous truck, with high-volume data the Type-II pattern is a good option.

5) Type-III: Hybrid (Figure 4, right): Neither a Type-I nor a Type-II distribution pattern presents a viable option for our motivating scenario, since the truck produces too much data for a cloud-only solution and the geographical distribution of the data sources prevents an edge-only solution. Therefore, in the Type-III distribution pattern, the situation recognition is distributed to both the cloud and the edge. This leads to the recognition of local situations at the edge and global situations in the cloud and their advantages.

- R_4 **- Low latency:** The latency for local situations is reduced as described in Type-II. However, global situations are evaluated in the cloud and the latency is as described in Type-I. Therefore, the requirement R_4 is fulfilled only for local situations.
- R_5 **- Low network traffic:** As in Type-II, network traffic can be saved by shifting the situation recognition to the edge. The situation objects of the local situations must be sent to the cloud for the evaluation of global situations, thereby increasing network traffic. However, by using context stripping, the data size of situation objects can be massively reduced and still enable further processing of global situations. Therefore, requirement R_5 is fulfilled.
- R_6 **- Data security & privacy:** Security and privacy of local situations match the Type-II pattern. Again, when using context stripping for local situations, we support complex scenarios and do not have to send sensitive context data within situation objects to the cloud. Therefore, R_6 is fulfilled.
- R_7 **- Cross company situation recognition:** As in the Type-I distribution pattern, a collaborative situation recognition is possible. However, a big advantage is gained by using context stripping. Possibly sensitive context data of each company remains at their respective edge. Only context-stripped local situations are sent to the cloud for the collaborative evaluation of the global situation. Therefore, requirement R_8 is fulfilled.

 ISBN: 978-1-61208-653-8

TABLE I. FULFILLMENT OF EXECUTION REQUIREMENTS BY THE DISTRIBUTION PATTERNS

	R_4	R_5	R_6	R_7
Type-I: Cloud-only	X	X	X	✓
Type-II: Edge-only	✓	✓	✓	X
Type-III: Hybrid	✓	✓	✓	✓

Except reducing the latency for the evaluation of global situations, all requirements are fulfilled by this hybrid approach. Especially the usage of context stripping presents multiple advantages when transferring local situations to the cloud. The Type-III distribution pattern is best-suited for complex scenarios with multiple data sources that require a fast reaction to local situations and a centralized situation recognition of global situations without increasing the network traffic. Multiple companies can collaborate without sharing sensitive data or infringing government regulation.

Table I summarizes the analysis of the different distribution patterns. As shown, the Type-III hybrid approach fulfills all execution requirements.

V. Summary and Future Work

In this paper, we present an approach for distributed situation recognition. To support the distribution, we extend the Situation Template Schema so that multiple things and situations can be used for context input using a layered modeling approach. Furthermore, we present the concept of context stripping to reduce network traffic by removing the associated context of situation objects. We examine three distribution patterns based on execution requirements that are important for a situation recognition in complex environments.

In future work, we intend to create a sophisticated cost model, since choosing a suitable distribution pattern is very use-case dependent. This way, users can receive a more detailed decision support based on their specific properties and requirements, which can lead to a faster adoption of new technologies like Edge Computing.

Acknowledgment This work is partially funded by the BMWi project IC4F (01MA17008G).

References

[1] D. Lucke, C. Constantinescu, and E. Westkämper, Manufacturing Systems and Technologies for the New Frontier: The 41st CIRP Conference on Manufacturing Systems May 26–28, 2008, Tokyo, Japan. London: Springer London, 2008, ch. Smart Factory - A Step towards the Next Generation of Manufacturing, pp. 115–118.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer Networks, vol. 54, no. 15, 2010, pp. 2787 – 2805.

[3] J. S. He, S. Ji, and P. O. Bobbie, "Internet of things (iot)-based learning framework to facilitate stem undergraduate education," in Proceedings of the SouthEast Conference. ACM, 2017, pp. 88–94.

[4] M. Wieland, H. Schwarz, U. Breitenbücher, and F. Leymann, "Towards situation-aware adaptive workflows: SitOPT – A general purpose situation-aware workflow management system," in Pervasive Computing and Communication Workshops (PerCom Workshops). IEEE, 2015, pp. 32–37.

[5] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, S. G. Sáez, and F. Leymann, "Situation recognition and handling based on executing situation templates and situation-aware workflows," Computing, 10 2016, pp. 1–19.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, 2016, pp. 637–646.

[7] K. Häussermann, C. Hubig, P. Levi, F. Leymann, O. Simoneit, M. Wieland, and O. Zweigle, "Understanding and designing situation-aware mobile and ubiquitous computing systems," in Proc. of intern. Conf. on Mobile, Ubiquitous and Pervasive Computing. Citeseer, 2010, pp. 329–339.

[8] Q. Fang, Y. Zhao, G. Yang, and W. Zheng, Scalable Distributed Ontology Reasoning Using DHT-Based Partitioning. Springer Berlin Heidelberg, 2008, pp. 91–105.

[9] X. Wang, D. Q. Zhang, T. Gu, and H. Pung, "Ontology based context modeling and reasoning using OWL," in Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, 2004.

[10] W. Dargie, J. Mendez, C. Mobius, K. Rybina, V. Thost, A.-Y. Turhan et al., "Situation recognition for service management systems using OWL 2 reasoners," in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on. IEEE, 2013, pp. 31–36.

[11] J. Attard, S. Scerri, I. Rivera, and S. Handschuh, "Ontology-based situation recognition for context-aware systems," in Proceedings of the 9th International Conference on Semantic Systems. ACM, 2013, pp. 113–120.

[12] B. Schilling, B. Koldehofe, U. Pletat, and K. Rothermel, "Distributed heterogeneous event processing: Enhancing scalability and interoperability of cep in an industrial context," in Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems. ACM, 2010, pp. 150–159.

[13] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch, "Distributed complex event processing with query rewriting," in Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, ser. DEBS '09. New York, NY, USA: ACM, 2009, pp. 4:1–4:12. [Online]. Available: http://doi.acm.org/10.1145/1619258.1619264

[14] K. V. Laerhoven and H. W. Gellersen, "Spine versus porcupine: a study in distributed wearable activity recognition," in Eighth International Symposium on Wearable Computers, vol. 1, Oct 2004, pp. 142–149.

[15] A. K. Dey, "Understanding and using context," Personal and ubiquitous computing, vol. 5, no. 1, 2001, pp. 4–7.

[16] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, and F. Leymann, "SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates," in Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing, 2015, Konferenz-Beitrag, pp. 113–127.

[17] A. C. Franco da Silva, P. Hirmer, M. Wieland, and B. Mitschang, "SitRS XT-Towards Near Real Time Situation Recognition," Journal of Information and Data Management, 2016.

[18] M. Mormul, P. Hirmer, M. Wieland, and B. Mitschang, "Situation model as interface between situation recognition and situation-aware applications," Computer Science - Research and Development, November 2016, pp. 1–12.

[19] O. N. Yilmaz, Y.-P. E. Wang, N. A. Johansson, N. Brahmi, S. A. Ashraf, and J. Sachs, "Analysis of ultra-reliable and low-latency 5g communication for a factory automation use case," in Communication Workshop (ICCW), 2015 IEEE International Conference on. IEEE, 2015, pp. 1190–1195.

[20] O. Moll, A. Zalewski, S. Pillai, S. Madden, M. Stonebraker, and V. Gadepally, "Exploring big volume sensor data with vroom," Proceedings of the VLDB Endowment, vol. 10, no. 12, 2017.

[21] J. Budomo, I. Ahmad, D. Habibi, and E. Dines, "4g lte-a systems at vehicular speeds: Performance evaluation," in Information Networking (ICOIN), 2017 International Conference on. IEEE, 2017, pp. 321–326.

[22] P. Hirmer, M. Wieland, U. Breitenbücher, and B. Mitschang, "Automated Sensor Registration, Binding and Sensor Data Provisioning," in CAiSE Forum, 2016.

[23] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on. IEEE, 2015, pp. 73–78.

All links were last followed on May 22.

 ISBN: 978-1-61208-653-8