SitRS XT – Towards Near Real Time Situation Recognition

Ana Cristina Franco da Silva, Pascal Hirmer, Matthias Wieland, Bernhard Mitschang

Institute of Parallel and Distributed Systems, University of Stuttgart,
Universitätsstr. 38, D-70569 Stuttgart, Germany
{franco-da-silva,lastname}@ipvs.uni-stuttgart.de

Abstract. Nowadays, the Internet of Things gains more and more attention through cheap, highly interconnected hardware devices that are attached with sensors and actuators. This results in an instrumented environment that provides sufficient context information to drive what is called situation recognition. Situations are derived from large amounts of context data, which is difficult to handle. In this article, we present SitRS XT, an extension of our previously introduced situation recognition service SitRS, to enable situation recognition in near real time. SitRS XT provides easy to use situation recognition based on Complex Event Processing, which is highly efficient. The architecture and method of SitRS XT is described and evaluated through a prototypical implementation.

Categories and Subject Descriptors: H.3 [**Information Storage and Retrieval**]: Miscellaneous; I.5 [**Pattern Recognition**]: Miscellaneous

Keywords: Complex Event Processing, Internet of Things, Situation-awareness, Situation Recognition

1. INTRODUCTION

The advances in sensor technologies, fast network communication, and cheap hardware led to a large number of so-called *smart things*, which are interconnected devices that continuously consume and exchange information about themselves and their surroundings [Aggarwal et al. 2013]. An environment containing one or more of these smart things is referred to as *smart environment*. Famous examples are smart homes, smart cities or smart factories [Vermesan and Friess 2013]. The volume of provided context information and how it is processed has become a challenging aspect in the paradigm known as *Internet of Things* (IoT) [Vermesan and Friess 2013; Ma et al. 2013].

The efficient processing of such low-level context information enables new advances through the recognition of occurring changes in the environment. These changes are referred to as *situations*, which are usually detected by sensors of distributed, interconnected smart devices. The integration of this low-level sensor data leads to so-called *high-level context*. Accomplishing a derivation of situations enables building situation-aware applications that are able to adapt to changing states and, thus, allow for self-organized smart environments. To enable such situation-awareness for IoT, we need a means to (i) extract context information, e.g. sensor data, in an efficient, timely manner, to (ii) continuously derive high-level context from a large amount of low-level context data, and to (iii) notify situation-aware applications about occurring situations. These aspects together ensure an effective, timely recognition of situations [Vermesan and Friess 2013].

In previous work, we proposed such a means by introducing the cloud-ready situation recognition service SitRS [Hirmer et al. 2015], which has been developed within the scope of the project SitOPT [Wieland et al. 2015]. SitRS integrates *things* into the internet by deriving their situational state based on sensor data. Furthermore, using so-called *Situation Templates*, a domain-specific model

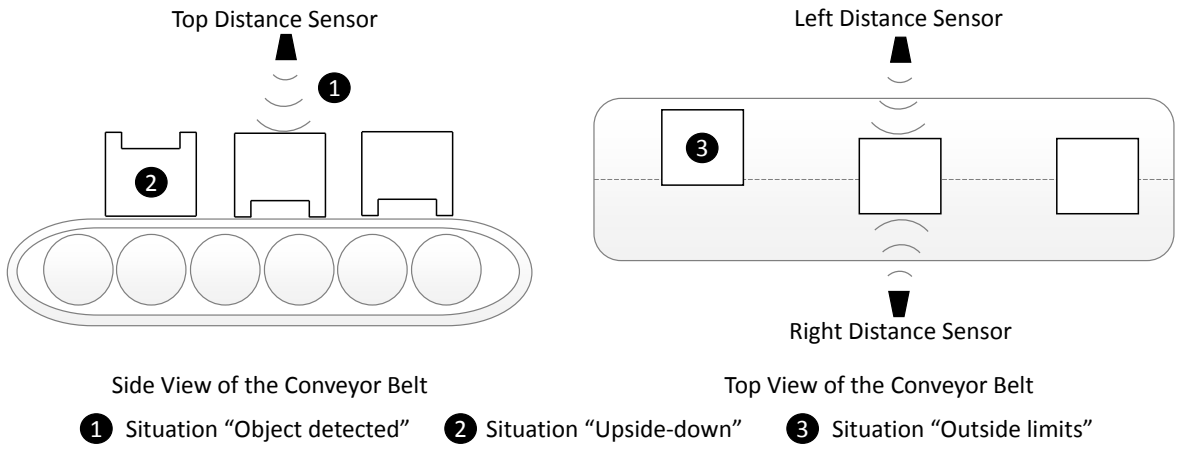


Fig. 1. Sensors for monitoring objects on a conveyor belt

that abstracts from complex, technical details, enables an easy means to define situations based on determining conditions. This model is automatically transformed onto executable representations, which are deployed into flow-based execution environments in order to recognize the occurrence of the modeled situation.

However, the current SitRS approach has revealed some shortcomings in respect of only supporting comparison of sensor data with fixed, predefined values. To suite more sophisticated scenarios, a means to recognize more complex situations that are time-based, require comparison of data from different sensors or require sensor data aggregation is needed. For example, to recognize situations that might occur when: a machine temperature is greater than 90 degrees for 10 seconds (time-based), the temperature of one sensor is greater than the temperature of another sensor (comparison of values from different sensors), or the average temperature is greater than 90 degrees (sensor data aggregation). Furthermore, the implementation of SitRS has revealed shortcomings in regard of efficiency, i.e., it does not support real-time situation recognition and scalability caused by the flow-based execution environment used for situation recognition. Moreover, in SitRS, context data is received based on a pull approach using an intermediate cache. This can lead to stale sensor data and can consequently decrease the quality of recognized situations.

In this article, we introduce *SitRS XT*, an extension of SitRS that tackles these issues (i) by supporting more complex situation recognition scenarios by enhancing the *powerfulness* of Situation Templates, (ii) by introducing a new, robust way to transform and execute the situation recognition using *Complex Event Processing* (CEP) [Luckham 2002; Bruns and Dunkel 2015], and (iii) by increasing *efficiency*, introducing stream-based sensor data provisioning, in contrast to the previous pull-based approach. Our main contribution is the new transformation and execution approach of Situation Templates combined with increased efficiency in order to achieve *near real-time situation recognition*. The concepts are evaluated through a prototypical implementation as well as corresponding runtime measurements. In the context of this article, *real time* stands for a guaranteed fast situation recognition time under all circumstances, whereas *near real time* stands for a fast situation recognition time without such guarantees.

The article is structured as follows: Section 2 introduces a motivating scenario. Section 3 presents an overview of our previous work SitRS. After that, Section 4 and Section 5 present the main contribution of our article, the SitRS XT system architecture and method. In Section 6, we present our prototypical implementation and in Section 7 related work. Section 8 gives a summary of the article, a conclusion, and an outlook on future work.

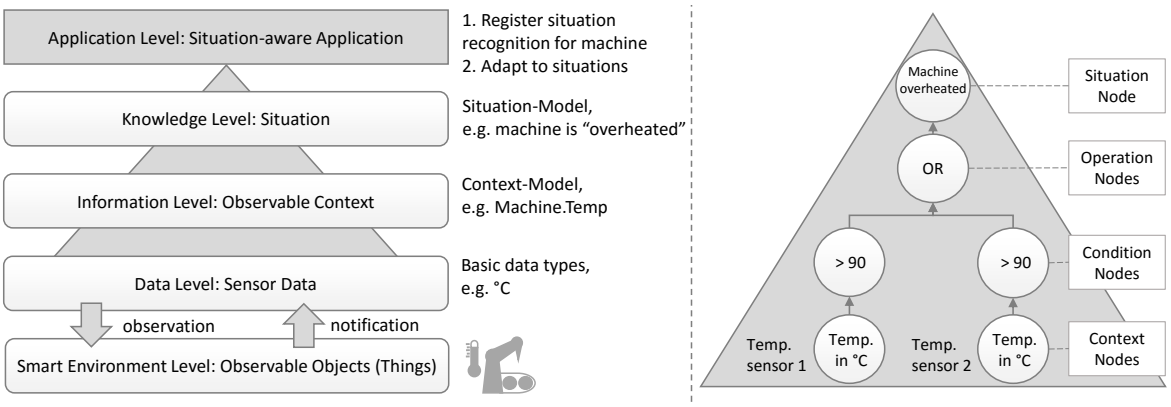


Fig. 2. Data processing levels based on [Hirmer et al. 2015]

2. MOTIVATING SCENARIO

This section introduces a motivating scenario in a production environment that is used throughout the article to explain our approach. Our contribution can also be applied to many other fields besides production environments. The goal of this scenario is the monitoring of objects on a conveyor belt in order to recognize if objects are positioned wrongly (cf. Figure 1). The recognition is based on the data generated by several distance sensors, which are attached to the conveyor belt. This scenario is abstracted from real production scenarios, where the main challenges are domain-specific situation modeling, continuous processing of streamed sensor data, and timely recognition of situations [Lucke et al. 2008]. Coping with these challenges allows us to timely notify situation-aware applications, which can immediately react to recognized situations in order to avoid, for example, further disturbances in the process. We define four situations for this scenario, which are depicted in Figure 1: (i) "Object detected" indicates the presence of an object on the conveyor belt, (ii) "Upside-down" indicates that an object is upside-down, (iii) "Outside limits" indicates that an object is positioned outside the allowed limits either left or right, and (iv) "Wrongly positioned" indicates that an object is either upside-down or positioned outside the allowed limits. The situation "Object detected" is a requirement for the occurrence of the other situations. Furthermore, this situation uses a time-based condition to avoid erroneous sensor values to be detected as a situation, i.e., to avoid mistakenly detecting an object. In Figure 6, we depict a Situation Template combining all these situations. The motivating scenario is implemented in our prototype and can be seen in action as a slightly adapted version in a video[1].

3. BACKGROUND – SITRS

In this section, we present the concepts of our previous work SitRS [Hirmer et al. 2015], which serves as basis for this article. In an IoT environment, sensors generate huge amounts of low-level data, which is difficult to handle. For this reason, this data needs to be derived into high-level situations, which can be better understood and processed by IoT applications. In order to provide situation-awareness to such applications, SitRS processes sensor data on different levels (cf. Figure 2). On the first level, called *data level*, only raw sensor data (e.g., temperature values) is available. This data is pushed to the *information level*, to be enhanced with information about relations of sensor data to real-world things (e.g., a production machine), becoming in this way information about the smart environment. Based on this information, i.e., the observable context, sensor data is aggregated and interpreted in order to derive situations, which leads to knowledge about the smart environment. This high-level knowledge is crucial for situation-aware applications, since it can be processed on a

[1]http://www.bit.ly/1Uod4Mq

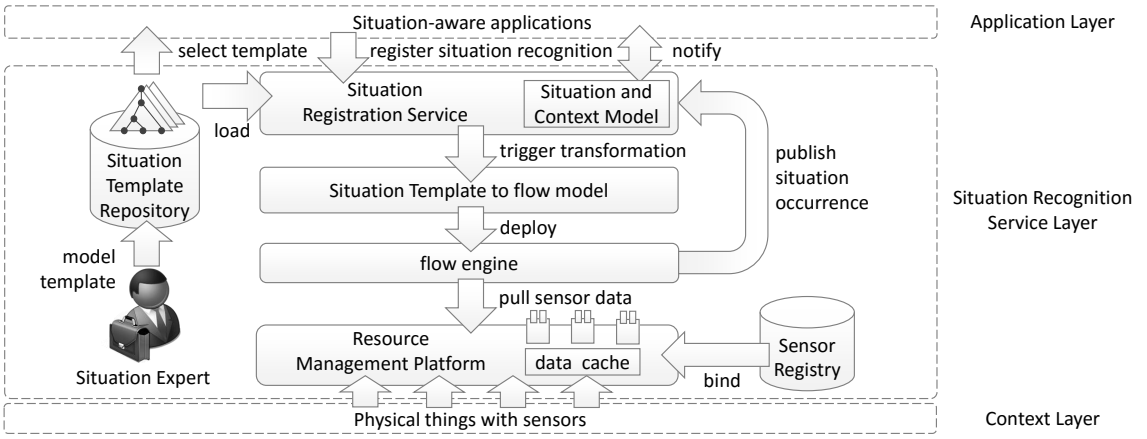


Fig. 3. SitRS Architecture [Hirmer et al. 2015]

higher-level of abstraction. In SitRS, situations are modeled as *Situation Templates*, a domain-specific model that abstracts from complex, technical details. This model contains the monitored sensors and the conditions that have to match for a certain situation to be recognized. Situation Templates are based on situation aggregation trees (SAT), which are directed, cohesive graphs as introduced by Zweigle et al. [2009]. In this model, sensors correspond to leaf nodes called context nodes and branches are aggregated bottom-up through a combination of so-called condition nodes and operation nodes until the root node is reached. A simple example of a Situation Template model is depicted in Figure 2, which defines the conditions to recognize when a production machine overheats, i.e., when the machine temperature passes the threshold of 90 degrees. The nodes of a Situation Template reflect the aforementioned processing levels: *context nodes* represent the monitored sensors of a certain thing, which correspond to the data level. Context nodes are connected to *condition nodes*, which establish the relations of sensor data to things (information level). Furthermore, condition nodes filter sensor data based on the defined conditions. Condition nodes can be aggregated by *operation nodes* using logical operations until the situation node is reached, which represents the situation to be recognized. The combination of condition, operation and situation nodes corresponds to the knowledge level, where sensor data is aggregated, interpreted and derived to situations. Defining situations with such easy means releases domain users of defining situations using complex representations (e.g., flows, CEP queries), which are required for the deployment in execution environments. Manually creating such complex representations requires expert knowledge and it is, thus, time-consuming and error-prone. SitRS reduces this complexity by automatically transforming Situation Templates onto the required executable representations. Furthermore, this approach also enables the support of different execution environments, avoiding dependency on a specific execution engine (vendor lock-in).

The architecture of SitRS for realizing the explained situation recognition approach is depicted in Figure 3. It is composed of three layers: (i) the Context Layer containing physical things and sensors, (ii) the Situation Recognition Service Layer, and (iii) the Application Layer containing situation-aware applications. Sensors can be registered through the *Resource Management Platform*, which provisions context information (i.e., sensor data) for the situation recognition through a pull-based approach. Situation-aware applications register their interest on a certain situation through the *Situation Registration Service*. Situations are modeled as Situation Templates and are available for registration through the *Situation Template Repository*. If a situation was not registered yet, the corresponding Situation Template first needs to be transformed onto an executable representation. The situation recognition starts by deploying the corresponding executable representation into the flow-based execution engine. Finally, situation-aware applications are notified about the occurrence of a situation, as soon as it occurs, through the Situation Registration Service. Details of SitRS can be found in [Hirmer et al. 2015]. The extensions of SitRS are explained in details in Sections 4 and 5.

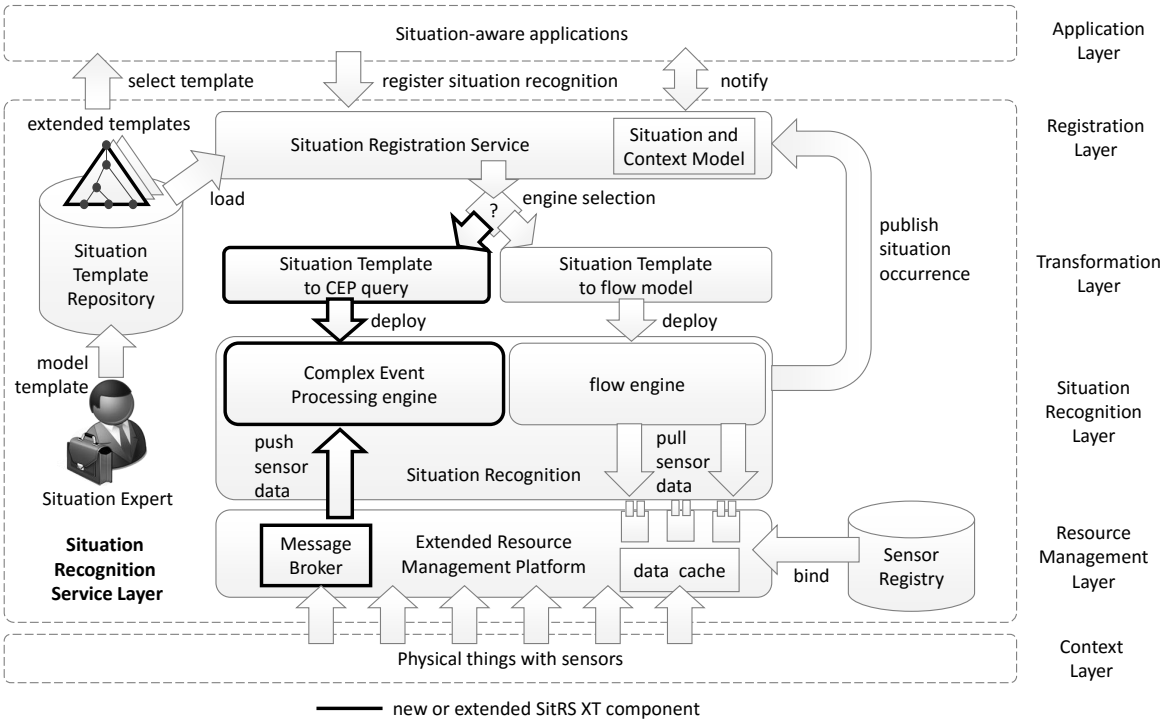


Fig. 4. SitRS XT Architecture based on [Hirmer et al. 2015]

4. SITRS XT ARCHITECTURE

Figure 4 depicts the system architecture of SitRS XT. The marked elements represent the extensions we present in this article. The focus of this article is the Situation Recognition Service Layer, which is divided into several sub-layers that cooperate to enable situation recognition:

The *Resource Management Layer* contains the *Sensor Registry* for registration and binding of sensors and the *Extended Resource Management Platform*, which receives context information from physical things and provisions this information to the execution engines. This article extends the existing Resource Management Platform of SitRS with a *Message Broker* in order to enable stream-based data provisioning. The Message Broker uses a publish/subscribe model [Eugster et al. 2003], in which physical things publish context information to the Message Broker. In this approach, context information can be processed timely, in contrast to a periodically triggered pull-based approach. This context information is forwarded to the *Situation Recognition Layer*. This layer contains the execution engines for situation recognition, which is extended in this article with a CEP engine. In this way, the SitRS XT architecture supports both flow engines and CEP engines. Furthermore, this layer contains the *Situation Template Repository* to store Situation Templates modeled by *Situation Experts* that define the conditions for situations to be recognized. In this article, we improve the powerfulness of Situation Templates for the modeling of more sophisticated situations, e.g., involving time-based conditions. The next layer, the *Transformation Layer*, provides: (i) a transformation of *Situation Templates to flow models*, and (ii) a transformation of *Situation Templates to CEP queries* as newly introduced in this article. Which kind of execution is chosen depends on the requirements of the scenario, i.e., on the modeled Situation Template. On the *Registration Layer*, situation-aware applications can register on situations of interest through the *Situation Registration Service* and get notified as soon as those situations occur. Notifications about situation occurrences are given to the applications through the Situation Registration Service. Due to the modular structure of the

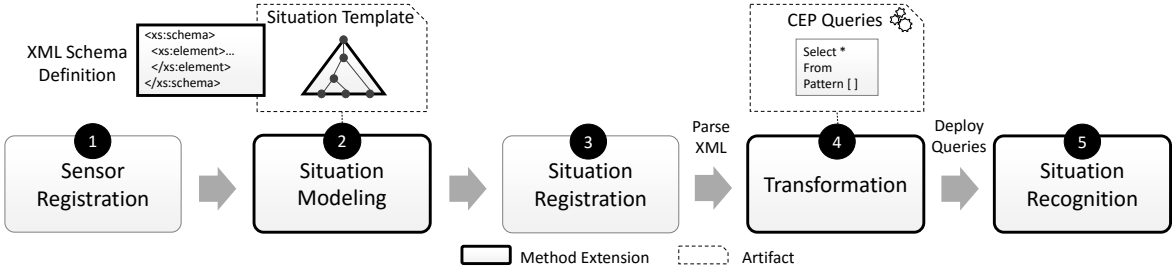


Fig. 5. Extended method for situation recognition (extends [Hirmer et al. 2015])

architecture, the components can be deployed in cloud computing environments and, thus, can be replicated, scaled, and distributed separately. Consequently, this enables the aforementioned cloud-readiness of our approach.

5. USAGE METHOD AND IMPLEMENTATION CONCEPT FOR SITRS XT

Figure 5 depicts the method how to use SitRS XT. This method extends the existing SitRS method by adding new concepts to the steps. There are two kinds of actors in this method: the *Situation Expert* and the *Context Expert*. The Situation Expert has the domain knowledge about the smart environment and about specific situations that need to be recognized. The Situation Expert models situations as Situation Templates and stores them into the Situation Template Repository. In contrast, the Context Expert has the technical knowledge about devices (e.g., things, sensors) of the environment and is responsible for registering the sensors to be used for situation recognition. The separation of technical and domain experts leads to a clear *separation of concerns*. The overall method consists of the following steps: (i) sensor registration, (ii) situation modeling, (iii) situation registration (iv) Situation Template transformation, and (v) situation recognition.

Step 1 – Sensor Registration

The first step of our extended method is the registration of available sensors in the Sensor Registry (cf. Figure 4). To register, for example, a distance sensor of the conveyor belt in our motivating scenario, the identifier of the *thing*, i.e., the conveyor belt, and unique sensor identifiers have to be specified. For that, an entry in the registry is created, which stores this information. Once a sensor is registered, the Resource Management Platform [Hirmer et al. 2015; Hirmer et al. 2016b; 2016a] is notified, which creates adapters to connect to the sensor and provisions its data to the situation recognition system either through the new Message Broker component (cf. Figure 4) or through REST resources. Details of this step can be found in [Hirmer et al. 2016a].

Step 2 – Situation Modeling

In this step, a situation is modeled as a Situation Template, which contains the monitored sensors and the conditions for the situation to be recognized. This model is based on situation aggregation trees (SAT), which are directed, cohesive graphs as introduced in [Zweigle et al. 2009]. Figure 6 depicts a tree representation of the situation "Wrongly positioned" (cf. Section 2) modeled as Situation Template. Possible types of condition nodes are *greater than*, *less than*, *equals*, *not equals* and *between*, while operation nodes can assume the logical operations *AND*, *OR*, *XOR* or *NOT*. A recognized situation is represented by this root node, called *situation node*. In our motivating scenario, the situation "Wrongly positioned" occurs if (i) an object is detected on the conveyor belt, and the object is either (ii) upside-down or (iii) positioned outside limits left or right. An object is detected on the conveyor belt if the measured distance from the *top sensor* is lower than the fixed distance between the top

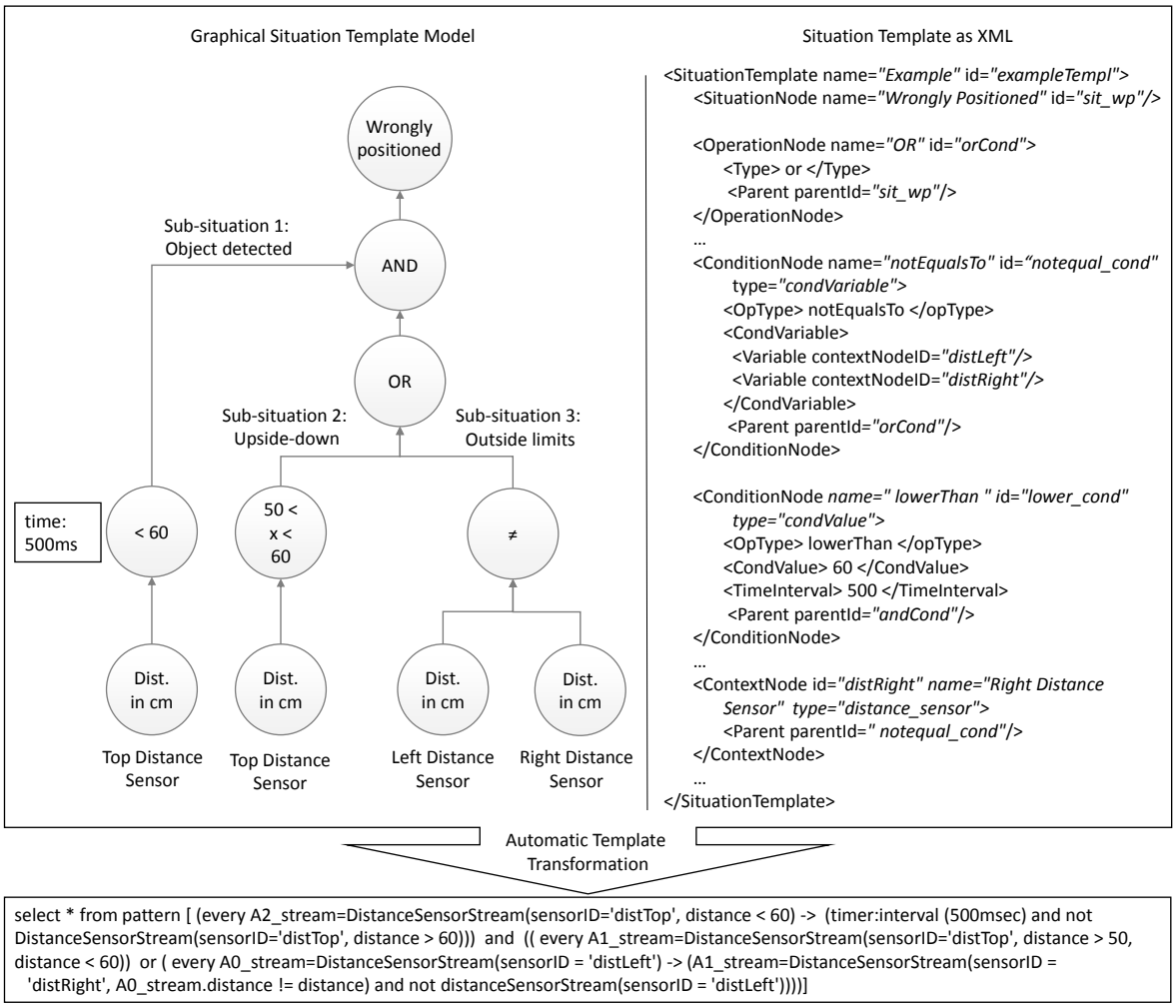


Fig. 6. Motivating scenario modeled as Situation Template and its transformation to an Esper CEP query

sensor and the conveyor belt. The object is *upside-down* if the distance between the top sensor and the object is greater than the usual distance for an upright object. The object is recognized to be outside limits if the measured distances from the *left sensor* and the *right sensor* are not the same. Note that the physical sensor deployment is not covered in this article. We assume that sensors are already deployed and therefore Situation Templates are modeled accordingly to the physical reality (e.g., object size, distance of sensors and breadth of belt).

The combination of graphical situation modeling and the automated template transformation releases the user from creating CEP queries him/herself (cf. Figure 6), which is difficult and error-prone since queries can become verbose depending on the complexity of the situation to be recognized. In our previous work [Hirmer et al. 2015], only situations composed of simple conditions can be modeled, i.e., only a comparison of sensor data with fixed, predefined values is possible.

In this work, we further extend this step by enabling the modeling of more sophisticated situations, which can (i) consist of time-based conditions, (ii) contain conditions comparing data from different sensors, and can (iii) aggregate sensor data already on the context level, for example, by computing the *maximum* value of multiple sensors. The extensions of the schema definition of condition nodes in Situation Templates are depicted in Figure 7. These changes only affect the condition nodes. A

New SitRS XT concepts	XML schema definition
New element for comparison of sensors	<xs:element name="CondVariable" minOccurs="0"> <xs:complexType> <xs:sequence> <xs:element name="Variable" type="tVariableInput" minOccurs="2" maxOccurs="2"/> </xs:sequence> </xs:complexType> </xs:element>
New element for time-based conditions	<xs:element name="TimeInterval" type="xs:integer" minOccurs="0" maxOccurs="1"> </xs:element>
New attribute for aggregation of values	<xs:attribute name="aggregation"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:pattern value="avg | min | max"/> </xs:restriction> </xs:simpleType> </xs:attribute>

Fig. 7. Extension of the condition node XML schema definition

time-based condition, i.e., a condition that is valid for a specific period of time, enables the modeling of situations that depend on a time window. For example the situation "Object detected" in the motivating scenario avoids erroneous sensor readings to be detected as a situation by using a time-based condition. Assuming that the distance between the top sensor and the conveyor belt is 60 cm, an object will only be detected if the measured distance is lower than 60 cm *for at least 500 milliseconds*. Furthermore, we enable the modeling of conditions comparing data from different sensors, such as the condition to detect if the object is positioned outside limits. Finally, the last extension enables the aggregation of different sensors and the common processing of their data. For example, if a machine has many temperature sensors and the situation recognition only needs to know if the average temperature is greater than 90 degrees. In this case, the sensor data can be aggregated by an *average function*, and it is checked if the result is greater than 90 degrees. Aggregating sensor data on the context level has the advantage of reducing efforts in modeling and processing condition nodes for each sensor.

Step 3 – Situation Registration

Situation-aware applications can register on situations of interest through the situation registration service in order to be notified when the registered situations occur. If there are no registrations for a situation yet, the situation registration triggers a transformation of the corresponding Situation Template onto an executable representation and the situation recognition is processed by the execution engine. If a situation is already being monitored, the registered application is notified once the situation occurs, and an additional transformation is not necessary. In case the monitoring of a situation is not needed anymore, it can be deregistered. If there are no more registrations for a situation, the recognition is stopped to save resources.

Step 4 – Transformation

We extended the transformation from Situation Templates onto an executable representation provided in our previous work by introducing a transformation onto CEP queries. To enable a CEP engine to monitor situations based on Situation Templates, the templates need to be transformed

onto executable event-based representations that can be executed by the engine. In our prototypical implementation, for example, the executable representation is defined using a CEP query language such as Esper EPL that can be executed by the CEP engine Esper[2]. Note that a wide range of CEP-based and non-CEP-based execution formats exist that could be supported by our approach. The interested reader is referred to [Hirmer et al. 2015; Hirmer and Mitschang 2016] for further information regarding a user-requirement-dependent selection of a suitable execution engine.

The inputs of the transformation are the identifier of the monitored thing (e.g., the conveyor belt) and the Situation Template identifier. In this step, we retrieve the Situation Template from the Situation Template Repository by using its identifier. We traverse its nodes in order to formulate a complex event pattern, which is composed of pattern expressions combined through logical operators (e.g., or, and). In our approach, the complex event pattern is built based on the set of condition nodes, which are aggregated by operation nodes. Each condition node corresponds to a pattern expression, while an operation node corresponds to a logical operator. The following example depicts such a complex event pattern in pseudo query language, which is roughly based on the Esper Event Processing Language (EPL):

```
select * from pattern [
  <conditionNode_patternExpression>
  <operationNode_type>
  <conditionNode_patternExpression>
]
```

To build the pattern expression for a condition node, we need the monitored sensor identifier and sensor type. This information is retrieved from the sensor registry by using the identifier of the monitored thing. In case there are more than one registered sensors of the same type, it is necessary to know, which sensor is used for a specific condition. For that, the *sensor role*, e.g., "Top Distance Sensor" in our motivation scenario, is specified in the context node and the transformation finds the registered sensor with this role. The structure of a pattern expression is shown below:

```
conditionNode_patternExpression = <sensor_type_stream>(
  sensor_id = '<monitored_sensor_id>',
  sensor_role = '<contextNode_sensor_role>',
  <conditionNode_condition>)
```

Figure 6 shows the CEP query to recognize the situation "Wrongly positioned" from our motivating scenario. This CEP query is complex and therefore difficult to create manually. By providing automatic transformation from Situation Templates onto CEP queries, we release users from the burden of creating such complex CEP queries themselves.

Step 5 – Situation Recognition

In the last step of our method, the recognition is started by deploying the CEP queries of the previous step into the execution engine. In our prototypical implementation, we deploy CEP queries into a CEP engine, which automatically starts the situation recognition. Through our introduced sensor data stream-based approach, the CEP engine receives input events as soon as sensor data is available. This enables a continuous monitoring of situations and a prompt recognition. On recognition, situation-aware applications that have registered for this situation are notified automatically through the Situation Registration Service.

[2]http://www.espertech.com/esper/

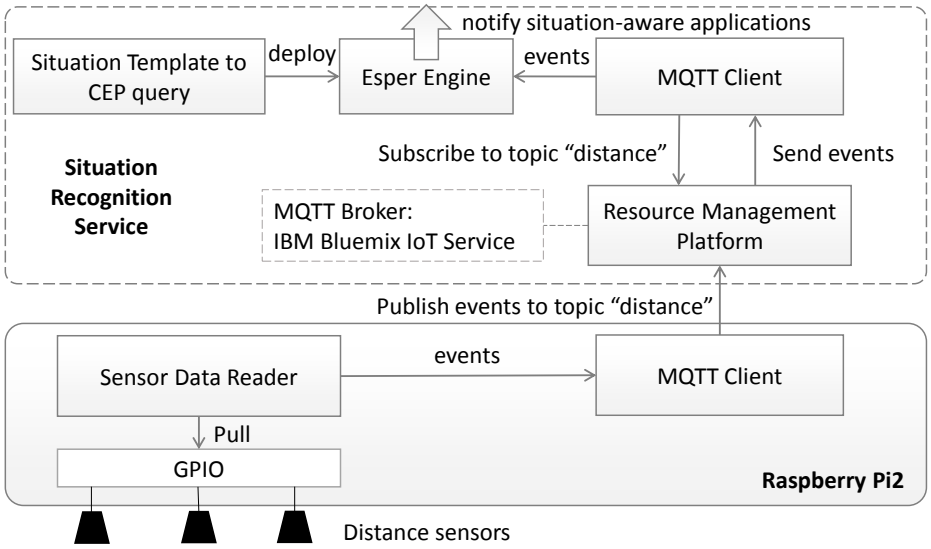


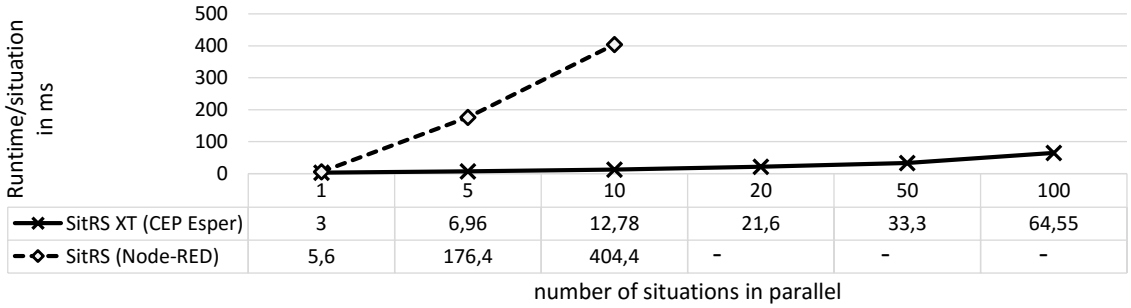
Fig. 8. SitRS XT prototype

6. IMPLEMENTATION AND PROTOTYPE

This section describes our prototypical implementation of the cloud-ready situation recognition service *SitRS XT* and provides the evaluation of our approach by presenting runtime measurements and a load test based on our prototype. The architecture of the SitRS XT prototype is depicted in Figure 8. For our experiments, it is sufficient to start with Situation Templates given in XML, because the modeling of Situation Templates can only be evaluated with extensive user studies, which are out of scope of this article. These templates are transformed to CEP queries. The transformation has been implemented as a Java library using the Java Architecture for XML Binding (JAXB) to parse the Situation Templates to Java objects. After the transformation, the CEP queries are added to the Esper engine to execute the situation recognition. In this prototype, the sensor data is published to SitRS XT through the IBM Bluemix IoT service[3], which provides a Message Broker based on the MQTT (Message Queue Telemetry Transport)[4] protocol.

To conduct the runtime measurements, a machine with 8 GB RAM and an Intel(R) Core(TM) i5-4300U CPU processor and a Windows 8 operation system was used. The Situation Template depicted in Figure 6 was used for our measurements. We measured (i) the transformation time of the Situation Template to a CEP query, and (ii) the situation recognition time, i.e., the time to recognize a situation. To measure the situation recognition time by the Esper engine, the situation "Wrongly positioned" was induced by sending events locally to the Esper engine that satisfy the condition "the object is outside limits". The transformation time of a single Situation Template was *145.6 milliseconds* on average based on *5* measurements. The situation recognition time based on one CEP query running in the Esper engine was *3 milliseconds* on average also based on *5* measurements. These measurements show that both the transformation and situation recognition steps are executed in a reasonable time. We conducted a load test as well to find out how many situations can be monitored in parallel. Furthermore, we compared the results to the measurements of SitRS [Hirmer et al. 2015]. Figure 9 shows the load test result for the same Situation Template as above. The parallelization degree was varied from 5 to 100 situations. It shows that the time to recognize a situation slightly increases with the number of monitored situations in parallel, i.e., the runtime of 3 milliseconds for one situation

[3]http://www.bluemix.net
[4]http://www.mqtt.org



	1	5	10	20	50	100
SitRS XT (CEP Esper)	3	6,96	12,78	21,6	33,3	64,55
SitRS (Node-RED)	5,6	176,4	404,4	-	-	-

Fig. 9. Load test of SitRS XT and comparison to SitRS

increases only to *64.55 milliseconds* on average for monitoring 100 situations in parallel. It shows that the Esper engine takes circa *6.4 seconds* to recognize 100 situations, i.e., the situation recognition using the Esper engine scales better than the prototypical implementation of our previous work SitRS, which uses the Node-RED[5] engine for situation recognition. Furthermore, Node-RED can only monitor up to 10 situations in parallel before it reaches its scalability limit. Note that we used the default setting of Esper and did not make manual optimizations for fair comparability with Node-RED.

Assessing the usability of Situation Template modeling is a difficult task, which still requires work to be done. However, the modeling of Situation Templates is quite similar to the modeling of business processes, which usability has been proven and become well-established. In particular, Situation Template modeling is an easier task because the resulting graphs are tree representations, which are much smaller than the huge, complex graphs in the case of business process models.

In summary, our extended approach for situation recognition overcomes the limitations in regard of scalability, revealed by the implementation of our previous work SitRS. In contrast to the other approaches mentioned in Section 7, SitRS XT shows efficient performance in order to be used in time-critical, real-world scenarios. Further evaluation in respect to the quality of the situation recognition is part of our future work.

7. RELATED WORK

In previous work [Hirmer et al. 2015], we presented SitRS, a cloud-ready situation recognition service that enables situation recognition based on raw sensor data. In this approach, sensors are bound dynamically and the sensor data is received in a pull-based manner. The situation recognition is executed in fixed time intervals by pulling the sensor data and deriving the situations. Currently SitRS only fits simple situation recognition scenarios, comparing sensor data with fixed, predefined values, which results from limitations in situation modeling using Situation Templates. In this article, we extend this approach by enhancing the power of Situation Templates, and by enabling a continuous processing of sensor data by a stream-based approach using CEP technologies. The SitRS architecture serves as basis. SitRS and SitRS XT are both based on Situation Templates as introduced by Häussermann et al. [2010] defining conditions that have to be satisfied for situations to be recognized.

Many similar approaches exist that use ontologies for situation recognition [Wang et al. 2004]. However, these approaches are either focused on specific use case scenarios [Brumitt et al. 2000] or cannot provide the efficiency required by real-time critical scenarios [Wang et al. 2004; Dargie et al. 2013], e.g., in smart factories. These limitations regarding efficiency occur in machine learning approaches [Attard et al. 2013] too. In contrast, our approach offers high efficiency by recognizing situations in milliseconds instead of seconds or even minutes as reported in [Wang et al. 2004; Dargie et al. 2013].

[5]http://www.nodered.org

This enables applicability in time-critical real-world scenarios, such as smart factories [Lucke et al. 2008], in which fast recognition times are of vital importance.

Several situation recognition systems using CEP were proposed in [Taylor and Leidinger 2011; Hasan et al. 2011; Glombiewski et al. 2013]. Taylor and Leidinger [2011] propose the use of ontologies to specify and recognize complex events, whose occurrence can be detected in digital messages streamed from multiple sensor networks. The developed ontology is accessed through an user interface, where the user specifies the events of interest. The specification of an event of interest is then processed in order to generate configuration commands for a CEP system. The CEP system monitors the specified data streams and generates notifications, which can be delivered to clients when the event occurs [Taylor and Leidinger 2011]. In this article, complex events are not directly specified, but rather abstracted as situations. Our approach also realizes transformations of the event specifications (i.e., the modeled situations) into commands for a CEP system. The difference is that we use Situation Templates to model situations of interest instead of an ontology. This enables not only the employment of CEP systems but also other technologies for situation recognition.

Hasan et al. [2011] propose to use CEP along with a dynamic enrichment of sensor data in order to realize situation awareness. In this approach, the situations of interest are directly defined in the CEP engine, i.e., the user formulates the situations of interest using CEP query languages [Hasan et al. 2011]. A dynamic enrichment component processes and enriches the sensor data before the CEP engine evaluates them against the situations of interest. This approach and the one in this article differ in following: No complex dynamic enrichment of the sensor data is done in this article. The necessary information about the sensor for the situation recognition (e.g., the sensor identification) is kept at a minimum. This information together with the sensor reading are made available directly to the CEP engine, dispensing any further processing step of the sensor data. Furthermore, instead of defining situations directly as CEP queries, which can be long and complicated depending of the situation, this work defines the situations of interest as Situation Templates. The abstraction provided through Situation Templates enables the employment of CEP engines as well as the use of other technologies for the situation recognition. Besides that, the use of Situation Templates also facilitates the modeling step of situations for the user, so that the user does not have to deal with the complexity of formulating CEP queries. The formulation of CEP queries is taken care of by transformations, which automatically create the necessary CEP queries for a given Situation Template.

Glombiewski et al. [2013] present a similar approach integrating context from a wide range of sources for situation recognition using event processing technologies as well. However, they do not provide any abstraction, i.e., the users of the situation recognition have to create CEP queries themselves. This proves difficult, especially for domain experts, e.g., in factories, who do not have extensive computer science knowledge. In our approach, we provide an abstraction by Situation Templates [Häussermann et al. 2010] and a graphical interface (cf. Figure 6), which enables the usage by domain experts without necessary knowledge of technical details such as event processing queries.

8. SUMMARY, CONCLUSION, AND OUTLOOK

In this article, we present *SitRS XT*, an extension of the situation recognition service SitRS, which supports Complex Event Processing (CEP) based situation recognition. By introducing CEP for the situation recognition in SitRS, we are now able to monitor more situations in parallel, improve the situation recognition speed and are able to process continuously sensor data. For realizing SitRS XT, we first had to introduce a data stream-based approach for the provisioning of sensor data to the situation recognition service. Based on that, the extended situation recognition service processes the sensor data using automatically generated CEP queries. These queries are generated using transformations derived from predefined Situation Templates. Furthermore, in this article we present a motivating scenario in a production environment that shows the advantages of our method: (i) modeling and recognizing complex situation recognition scenarios, (ii) an efficient situation recognition

suitable for time-critical scenarios, e.g., smart factories, and (iii) an enhanced scalability in contrast to other approaches as reported in Section 7. Since the number of deployed sensors and smart environments is increasing fast, the number of context/situation-aware applications is increasing as well. As a consequence, the pressure on the involved middleware (CEP systems, situation recognition systems, etc.) is high. This motivated us to extend our situation recognition system SitRS towards enhanced expressiveness and increased performance, as shown by a comparison to related work (cf. Section 7).

We achieved near real time through the new transformation approach and hence increased efficiency in contrast to previous work and other approaches. However, to fulfill real-time requirements from domains like smart production environments, i.e., to guarantee predefined recognition times, additional means are required such as scalability and availability through distribution and infrastructure virtualization. This will be part of our future work. Furthermore, we will conduct user studies for further evaluation.

Acknowledgment

This work is funded by the DFG project SitOPT (610872) and the BMWi project SmartOrchestra (01MD16001F).

REFERENCES

Aggarwal, C. C., Ashish, N., and Sheth, A. The Internet of Things: a survey from the data-centric perspective. In C. C. Aggarwal (Ed.), *Managing and Mining Sensor Data*. Springer, Boston, pp. 383–428, 2013.

Attard, J., Scerri, S., Rivera, I., and Handschuh, S. Ontology-based Situation Recognition for Context-Aware Systems. In *Proceedings of the International Conference on Semantic Systems*. Graz, Austria, pp. 113–120, 2013.

Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. EasyLiving: Technologies for Intelligent Environments. In *Handheld and Ubiquitous Computing*. Springer Berlin Heidelberg, 2000.

Bruns, R. and Dunkel, J. *Complex Event Processing: komplexe analyse von massiven datenströmen mit CEP*. Springer-Verlag, 2015.

Dargie, W., Eldora, E., Mendez, J., Möbius, C., Rybina, K., Thost, V., and Turhan, A.-Y. Situation Recognition for Service Management Systems Using OWL 2 Reasoners. In *Proceedings of the IEEE Workshop on Context Modeling and Reasoning*. San Diego, USA, pp. 31–36, 2013.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. The Many Faces of Publish/Subscribe. *ACM Computing Surveys* 35 (2): 114–131, 2003.

Glombiewski, N., Hossbach, B., Morgen, A., Ritter, F., and Seeger, B. Event Processing on Your Own Database. In *Proceedings of the Datenbanksysteme für Business, Technologie und Web Workshopband*. Magdeburg, Germany, pp. 33–42, 2013.

Hasan, S., Curry, E., Banduk, M., and O'Riain, S. Toward Situation Awareness for the Semantic Sensor Web: complex event processing with dynamic linked data enrichment. In *Proceedings of the International Conference on Semantic Sensor Networks*. Aachen, Germany, pp. 69–81, 2011.

Häussermann, K., Hubig, C., Levi, P., Leymann, F., Simoneit, O., Wieland, M., and Zweigle, O. Understanding and Designing Situation-Aware Mobile and Ubiquitous Computing Systems. *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 4 (3): 562–571, 2010.

Hirmer, P. and Mitschang, B. FlexMash - Flexible Data Mashups Based on Pattern-Based Model Transformation. In *Rapid Mashup Development Tools*. Springer International Publishing, pp. 12–30, 2016.

Hirmer, P., Reimann, P., Wieland, M., and Mitschang, B. Extended Techniques for Flexible Modeling and Execution of Data Mashups. In *Proceedings of the International Conference on Data Management Technologies and Applications*. Colmar, France, pp. 111–122, 2015.

Hirmer, P., Wieland, M., Breitenbücher, U., and Mitschang, B. Dynamic Ontology-Based Sensor Binding. In J. Pokorný, M. Ivanović, B. Thalheim, and P. Šaloun (Eds.), *Advances in Databases and Information Systems*. Lecture Notes in Computer Science, vol. 9809. Springer, pp. 323–337, 2016a.

Hirmer, P., Wieland, M., Breitenbücher, U., and Mitschang, B. Automated Sensor Registration, Binding and Sensor Data Provisioning. In *Proceedings of the International Conference on Advanced Information Systems Engineering*. Ljubljana, Slovenia, pp. 81–88, 2016b.

Hirmer, P., Wieland, M., Schwarz, H., Mitschang, B., Breitenbücher, U., and Leymann, F. SitRS - A Situation Recognition Service Based on Modeling and Executing Situation Templates. In *Proceedings of the Symposium and Summer School On Service-Oriented Computing*. Crete, Greece, pp. 113–127, 2015.

Lucke, D., Constantinescu, C., and Westkämper, E. Smart Factory - A Step towards the Next Generation of Manufacturing. In *Proceedings of the CIRP Conference on Manufacturing Systems*. Tokyo, Japan, pp. 115–118, 2008.

Luckham, D. *The Power of Events*. Vol. 204. Addison-Wesley Reading, 2002.

Ma, M., Wang, P., and Chu, C.-H. Data Management for Internet of Things: challenges, approaches and opportunities. In *Proceedings of the IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. Beijing, China, pp. 1144–1151, 2013.

Taylor, K. and Leidinger, L. Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks. In G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. D. Leenheer, and J. Pan (Eds.), *The Semanic Web: research and applications*. Lecture Notes in Computer Science, vol. 6644. Springer, pp. 285–299, 2011.

Vermesan, O. and Friess, P. *Internet of Things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

Wang, X., Zhang, D. Q., Gu, T., and Pung, H. Ontology Based Context Modeling and Reasoning Using OWL. In *Proceedings of the IEEE Annual Conference on Pervasive Computing and Communications Workshops*. Orlando, USA, pp. 18–18, 2004.

Wieland, M., Schwarz, H., Breitenbücher, U., and Leymann, F. Towards Situation-Aware Adaptive Workflows. In *Proceedings of the Workshop on Context and Activity Modeling and Recognition*. St. Louis, USA, pp. 32–37, 2015.

Zweigle, O., Häussermann, K., Käppeler, U.-P., and Levi, P. Supervised Learning Algorithm for Automatic Adaption of Situation Templates using Uncertain data. In *Proceedings of the International Conference on Interaction Sciences: information technology, culture and human*. Seoul, Korea, pp. 197–200, 2009.