Using Environmental Contexts to Model Restrictions on Sensor Capabilities

Martin Richter, Christine Jakobs, Theresa Werner, Matthias Werner
Operating Systems Group
Chemnitz University of Technology
09111 Chemnitz, Germany
email: {martin.richter, christine.jakobs, theresa.werner, matthias.werner}@informatik.tu-chemnitz.de

***Abstract*—Cyber-Physical Systems (CPS) incorporate the physical and digital worlds via sensors and actuators. The system devices may be heterogeneous, distributed in space, and mobile. This leads to new challenges in the design of applications that should utilize the full potential of the system. As devices are unreliable and may be arbitrarily moving away from locations of interest, there is a continuous necessity to replace them with alternatives during runtime. The handling of this task by the application programmer is error-prone and complex because the system might be heterogeneous and different sensors and actuators possibly possess varying means to observe and influence the environment. Therefore, a capability model has to be employed on the operating system level which provides a holistic view of the different devices. In this regard, an environmental context model has to be supplied as the devices' capabilities may be restricted depending on the contexts they are located in. This paper presents an environmental context model based on an abstract sensor capability model. In conjunction, the models provide a description of sensors concerning their ability to observe properties of physical objects of interest within their particular environmental contexts. The effect of contexts on the spatial interpretability of sensor measurements is therefore made explicit. Corresponding inferences are made with respect to the localization of physical objects or phenomena of interest.**

Keywords*—*cyber-physical systems; context awareness; heterogeneity; sensor virtualization.

I. Introduction

Through emerging trends like the Internet of Things, Industry 4.0 or Smart Home, devices like sensors and actuators are of increasing importance in our daily lives. This leads to the emergence of Cyber-Physical Systems (CPS) that create a link between the physical and digital worlds. Such systems consist of devices that may be large in number, distributed in space, heterogeneous, unreliable, and mobile. As the target of CPS is the observation and influence of the physical world, the respective applications are often bound to certain devices situated in locations of interest. This severely impairs the portability of applications between systems. Additionally, the unpredictable motion and failure of sensors as well as actuators become a challenge for properly executing the applications. Furthermore, the system's full potential may go unrecognized as certain tasks may only be executable when taking the aggregated capabilities of multiple devices into account instead of considering them individually. The coordination of such groups of devices heavily depends on their environmental contexts as they may limit the devices' abilities to cooperate (e.g., robots being close to each other but being located on opposite sides of a wall). The described challenges lead to a demand for new device capability models that abstract the heterogeneous hardware such that applications are made portable and the full potential of CPS is utilized.

This paper provides an abstract sensor capability model. It incorporates the influence of environmental contexts on the ability of sensors to observe their surroundings. Additionally, it allows to describe which combinations of the available devices are suitable for performing a task with regard to their respective locations and environmental contexts. This enables the virtualization of hardware resources, i.e., the transparent utilization of a common set of devices by multiple applications.

Existing approaches to sensor virtualization (such as [1] and [2]) focus on concurrently executing multiple applications on a fixed set of sensors. In our view, the dynamics of the different devices have to be recognized. Due to the motion and failure of sensors, it is necessary to alternately execute applications on a changing set of devices such that they are able to continuously observe and influence the environment as desired. Our model enables the Operating System (OS) to decide which sensor measurements are required for a given task based on the programmer's task description, the sensors' capabilities, as well as the devices' environmental contexts. As the task description is detached from controlling the sensors, the execution of an application on transparently alternating sets of devices is accomplished. This leads to new possibilities for the task planning of sensors. Their mobility and heterogeneity can be exploited such that a leaving or failing device may be replaced by one or possibly multiple other devices that provide similar types of measurements. In that way, the application's possibilities for sensing and influencing the environment remain the same or may even be enhanced. Therefore, it is possible to unlock the full potential of CPS and to port applications to other systems that employ unrelated devices.

As a running example, we use a street surveillance system near a highway. Its goal is the detection of environmental hazards close to the road, such as wildfires. It consists of immobile cameras on the roadside as well as cars driving down the street which incorporate a dashboard camera as well as two temperature sensors for measuring interior and exterior temperatures respectively. The back seat windows of the cars shall be tinted such that cameras from the outside are not able to observe the back seat area of the car. The front seats of the car are observable through untinted windows. We assume that all sensors provide information on their respective locations and are able to communicate wirelessly.

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-106-0

The article is further structured as follows. Section II depicts the concept for an abstract sensor and environmental context model. Section III discusses existing approaches for virtualizing sensors and compares them to our model with respect to modeling the influence of environmental contexts on the capabilities of devices. Section IV presents the conclusion and possible directions for future work.

II. Concept

The following sections present the programming model on which our sensor capability model is based, followed by the capability model, as well as the environmental context model.

A. Programming Model

The presented capability model is created in the context of the programming model introduced in [3]. The programming model provides sensor and actuator virtualization, as well as transparency with respect to distribution, location, and motion at the application level while allowing location- and motion-aware control of devices at the OS level. This is achieved by reversing the view of the CPS developer such that he or she takes a physical perspective. In his application, the programmer describes the properties of a physical object and its behavior depending on internal dynamics as well as potential actuator actions. The set of properties and their values form the state of the object, which may change over time. Additionally, the developer specifies a target state for the object via constraints. These constraints are then solved based on the state and behavioral descriptions of the object. The capability and context model we propose in this work is used to determine whether the available sensor capabilities with respect to the sensor contexts are sufficient for observing the object of interest and if so, in which locations it is present.

As mentioned before, the programmer provides a property description for the physical object of interest. For the object o it takes the form of a state description vector $\vec{z}_o$.

$$\vec{z}_o = \left[(\tau_1, r_1), \ldots, (\tau_n, r_n)\right]^T \tag{1}$$

Each element of the vector consists of a tuple (τ_i, r_i). The first element denotes an object property type τ_i (e.g., color, shape, or temperature), which represents the domain of possible values for the corresponding property. The second element is a rule for specifying a range of values for the property (e.g., for a fire the color has to be red or yellow, and the temperature has to exceed 300 degrees Celsius). Each rule represents a logic formula that evaluates either to `true` or `false` for a given location based on the available sensor measurements. If all the rules for an object evaluate to `true` at a given location, the object is present there, otherwise it is assumed not to be.

B. Sensor Model

The provided programming model allows the developer to exclusively focus on the creation of the state description vector $\vec{z}_o$. The OS is responsible for utilizing the required sensors transparently. To achieve this, it requires knowledge of the sensors' capabilities as each of them may have different measurands and may observe varying physical phenomena. The following paragraphs provide an abstract model for the capabilities of a sensor.

A possibly mobile sensor at location $\vec{x} = \vec{x}(t)$ observes a physical quantity q (e.g., electromagnetic radiation or temperature). The measurement of the physical quantity is then transformed by the sensor into a digital signal $v = v(t)$ (e.g., an array of pixels) through a measuring process μ. The resulting signal is interpretable for different locations in space $X \subseteq X_\Sigma$, where X_Σ denotes all locations relevant to the system. The set of interpretable locations depends on the sensor's location $\vec{x}$ and possibly other sensor-specific parameters $\vec{p} = \vec{p}(t)$, which may also change over time (i.e., $X = X(\vec{x}, \vec{p})$). In conclusion, a sensor s is characterized by the following six-tuple.

$$s = (q, \vec{x}, v, \mu, \vec{p}, X(\vec{x}, \vec{p})) \tag{2}$$

To select sensors for given tasks efficiently it is necessary to group them into classes. The knowledge about the class of a sensor allows the OS to decide how to interpret the outputs of sensors (i.e., which actions to perform on them). It can then transform the results into instantiated physical object properties (e.g., shape or form). Sensors within a class measure the same physical quantity and their results can be utilized similarly. Therefore, the class of a sensor depends on its observed quantity q and its measurement process μ. We denote a sensor class γ as a boolean function that returns `true` for a sensor s_j belonging to the class and `false` otherwise.

$$\gamma_i(s_j) = \begin{cases} true, & \text{sensor } s_j \text{ belongs to class } \gamma_i \\ false, & \text{otherwise} \end{cases} \tag{3}$$

For example, digital dashboard cameras measure electromagnetic radiation within a certain wavelength and transform it into an array of pixels through their measurement process. On these arrays, methods like image recognition can be performed to extract desired object properties such as shape or color. The set of interpretable locations for a camera is described by the following formula for a cone, where $\vec{x} = \vec{x}(t)$ denotes the location of the sensor at time point t, $\vec{u}$ is the location vector for which the equation has to be solved, and the parameter vector $\vec{p}$ consists of the orientation of the camera $\vec{o} = \vec{o}(t)$ as well as its viewing angle ϕ.

$$X\left(\vec{x}, \begin{bmatrix}\vec{o} & \phi\end{bmatrix}^\top\right) = \left\{\vec{u} \in X_\Sigma : \frac{(\vec{u} - \vec{x}) \cdot \vec{o}}{|\vec{u} - \vec{x}||\vec{o}|} \leq \cos\phi\right\} \tag{4}$$

This equation could also be further constrained by considering the maximum range of the camera. A brightness sensor, in contrast, also measures electromagnetic radiation but through its measurement process its output can not be utilized for similar methods as the camera. Additionally, its measurement is solely interpretable for a small radius around its location.

C. Sensor Identification

Depending on the object description presented in Section II-A, multiple different sensors may have to be utilized to gather data on an object of interest. The information on which methods may be used to extract object properties from sensor

outputs and which classes of sensors are required to utilize them is managed in a dictionary $\mathcal{D}$. The dictionary takes as inputs:

1) the set of all sensors $S(t)$, which may change over time due to motion or failure,
2) the set of all available methods M, which may be applied to different classes of sensors, and
3) the type τ of the physical property z_i.

As a result of these inputs, the dictionary returns a vector of different methods $\vec{m}$. They can be applied to the outputs of sensors of the corresponding classes to calculate a value for the physical property.

$$\mathcal{D}(S(t), M, \tau) = \vec{m} \tag{5}$$

For example, for determining the shape of an object image recognition methods may be used on the outputs of multiple cameras or sophisticated laser scanning systems may be utilized.

Each method m_j in $\vec{m}$ is a tuple consisting of a set of sensor classes Γ_j^τ and a function ψ_j for calculating the physical property's value from the outputs of sensors of the corresponding classes.

$$m_j = (\Gamma_j^\tau, \psi_j) \tag{6}$$

For each of the sensor classes in Γ_j^τ an individual sensor has to be chosen for providing the inputs to the corresponding calculation function ψ_j. All possible sets of input sensors are determined by a function g. It maps the currently available sensors $S(t)$ and the required classes of sensors Γ_j^τ to the corresponding sets of sensors Θ_j, which are a subset of the power set of $S(t)$.

$$g(S(t), \Gamma_j^\tau) = \Theta_j, \Theta_j \subset \mathcal{P}(S(t)) \tag{7}$$

The dimensions of each set of sensors $\theta_i \in \Theta_j$ depend on how many sensors are required by the method m_j. For example, for the calculation of the position of an object either one distance sensor with known location and orientation is required or several cameras may be utilized through the method of triangulation.

When the function ψ_j is applied to the set of chosen sensors θ_i a value v of type τ is created.

$$\psi_j(\theta_j) = v \tag{8}$$

This value v may correspond to the value of a physical object property depending on whether the corresponding rule r in the state description vector is satisfied or not.

D. Environmental Context Model

Not all sensors that belong to the required classes for performing a method can be utilized for the method. Different sensors may be located in different environmental contexts such that their measurements do not stand in any relationship with each other. An example of this is an interpolation method that can not be used simultaneously for temperature sensors in a car and temperature sensors on the roadside.

An environmental context constrains the area for which the sensors' output can be interpreted. Each context c is defined with respect to its influence on a physical quantity q (e.g., a closed window influences the interpretability of temperature sensors but not of cameras). A context possesses an anchor location $\vec{x}(t)$ which may change over time. This location is continuously updated, e.g., by the use of a positioning sensor. Depending on its anchor location the context additionally consists of a set of surrounding locations $X(\vec{x}(t))$ which describe the spatial extent of the context. A context might inhibit the interpretation of sensor measurements of the corresponding physical quantity q in two ways:

1) It may impede the interpretation of a measurement from outside the context for a location inside the context.
2) It may impede the interpretation of a sensor measurement within the context for a location outside the context.

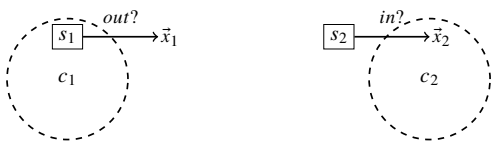
For example, a camera located inside a vehicle may be able to observe locations outside through a tinted window but a camera located outside the vehicle is not able to observe its inside through the same window. This is accounted for by introducing the boolean attributes *in* and *out* for each context. If *in* is `false`, the context constrains the interpretation of sensor measurements from outside the context for locations inside the context. If it is `true` an interpretation is possible. The attribute *out* describes whether the interpretation of a measurement from inside a context is feasible for a location outside in a similar fashion. Figure 1 depicts this situation. In conclusion, a context c related to a physical quantity q is defined by the following quintuplet.

$$c = c(t) = \{q, \vec{x}(t), X(\vec{x}(t)), in, out\}\,,\ X(\vec{x}(t)) \subseteq X_\Sigma \tag{9}$$

The locations for different contexts may overlap arbitrarily. Therefore, a sensor may be influenced by multiple contexts at once. As the sensor's location and the contexts' locations may change over time, the set of contexts that influence a sensor may also change over time.

E. Impact of Contexts on Sensor Measurements

For a given sensor s, the set of interpretable locations $s.X$ is created without taking its environment into account (see Section II-B). The set of locations the sensor is actually able to observe may be smaller because it is constrained by the set of environmental contexts C. The measurement of a sensor can be restricted by a context in two ways, as described in Section II-D. Therefore, the locations X_{obs}^s a



(a) Observability of $\vec{x}_1$ outside c_1 with s_1 being located within c_1 (*out*).

(b) Observability of $\vec{x}_2$ within c_2 with s_2 being located outside c_2 (*in*).

Figure 1. Depiction of the *in* and *out* attributes of the contexts c_1 and c_2 regarding the sensors s_1 and s_2 that may or may not measure the locations $\vec{x}_1$ and $\vec{x}_2$ respectively.

sensor observes from within its current contexts depend on two sets of locations $X^s_{\neg out}$ and $X^s_{\neg in}$. The set $X^s_{\neg out}$ denotes the locations of contexts a sensor is located in which do not allow the measurement of the sensor to be interpreted for locations outside (i.e., $\neg c.out$).

$$X^s_{\neg out} = \bigcap_{\substack{c \in C, c.q = s.q, \\ \neg c.out, s.\vec{x} \in c.X}} c.X \tag{10}$$

Thus, the locations $s.X$ a sensor may be able to measure have to be intersected with $X^s_{\neg out}$. This removes all locations from $s.X$ which lie outside of the corresponding contexts. From the resulting set the locations $X^s_{\neg in}$ have to be removed. They refer to contexts in which the sensor is not located and which prohibit the interpretation of the sensor's output from outside the context for a location within the context (i.e., $\neg c.in$).

$$X^s_{\neg in} = \bigcup_{\substack{c \in C, c.q = s.q, \\ \neg c.in, s.\vec{x} \notin c.X}} c.X \tag{11}$$

In conclusion, the resulting set of locations X^s_{obs} a sensor is able to observe is defined by:

$$X^s_{obs} = (s.X \cap X^s_{\neg out}) \setminus X^s_{\neg in}. \tag{12}$$

Figure 2 shows an example for this. The camera sensor s is able to measure a cone-shaped area in front of it and is located within the contexts c_1 and c_2. Therefore, the *out* attributes of these contexts are relevant for the spatial interpretation of the output of s. Only c_2 constrains the sensor's measurement locations in this regard. The sensor is not located within c_3 and c_4. Thus, their respective *in* attributes are of relevance as both contexts consist of locations that may intersect with the sensor's observable locations $s.X$. The locations X^s_{obs} for which its measurements are interpretable are represented by diagonal stripes.

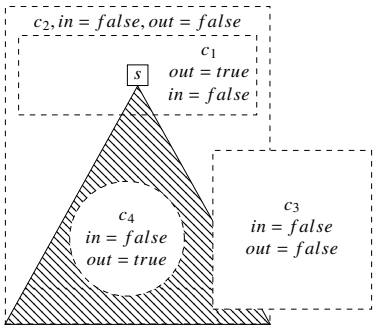


Figure 2. Depiction of observable locations X^s_{obs} for a sensor s with respect to the contexts c_1, c_2, c_3, and c_4 that relate to the same physical quantity s is measuring.

The set of context regions X^s_{con} for a sensor s is defined similarly to X^s_{obs}.

$$X^s_{con} = (X_\Sigma \cap X^s_{\neg out}) \setminus X^s_{\neg in} \tag{13}$$

It denotes the entirety of the sensor's context without taking its observable locations $s.X$ into account. This set allows to analyze whether the measurements of different sensors stand in relation to each other, i.e., whether their context regions overlap.

The set $X^{\theta^{m_i}}_{obs}$ denotes the sets of observable locations for a chosen set of sensors $\theta_{m_i} \in \Theta_i$ utilized by method m_i.

$$X^{\theta^{m_i}}_{obs} = \{s \in \theta_{m_i} : X^s_{obs}\} \tag{14}$$

The set $X^{\theta^{m_i}}_{con}$ depicts the sets of context regions for a chosen set of sensors similarly to $X^{\theta^{m_i}}_{obs}$.

$$X^{\theta^{m_i}}_{con} = \{s \in \theta_{m_i} : X^s_{con}\} \tag{15}$$

The scope $\mathcal{S}_{m_i}$ for the result of a method m_i is determined by a function α.

$$\mathcal{S}_{m_i} = \alpha(X^{\theta_{m_i}}_{obs}, X^{\theta^{\theta_{m_i}}}_{con}, \psi) \tag{16}$$

The scope describes which locations in the system can be covered by the corresponding method by utilizing varying sets of sensors. If the scope is an empty set, the method is not applicable to the available sensors in the given contexts. It depends on the following three parameters:

1) the different sets of observable locations $X^{\theta^{m_i}}_{obs}$ of the utilized sensors as they determine for which locations valid sensor outputs are available,
2) the different sets of context regions $X^{\theta^{m_i}}_{con}$ of the utilized sensors because they describe which sensor measurements stand in relation to each other and may be used for methods like interpolation for example, and
3) the function ψ for calculating the result of the method as varying calculation functions may lead to different results, e.g., interpolation between temperature sensors increases the scope of a method and triangulation between cameras reduces it.

F. Object Identification

As described in Section II-A, a physical object property z is computable by a set of different methods. Each of these methods provides a result for a given scope based on which sensors are chosen for its execution. Depending on the programmer's target, it may be feasible to utilize multiple different methods to increase the area in which an object's property can be observed. For a chosen set of methods $\vec{m}_\tau$ for determining the physical object property z of type τ, z is observable in the set of locations X^z which is a union of the utilized methods' scopes.

$$X^z = \bigcup_{m_i \in \vec{m}_\tau} \mathcal{S}_{m_i} \tag{17}$$

The physical object o (i.e., all its properties) is observable at the intersection of the observed locations of all its properties.

$$X^o_{obs} = \bigcap_{z_i \in \vec{z}} X^{z_i} \tag{18}$$

The actual spatial scope X^o of the object is determined by the rules r_i provided by the state description $\vec{z}_o$ (see Section II-A).

$$X^o = \bigcap_{\substack{(\tau_i, r_i) \in \vec{z}_o, \vec{m} = D(S, M, \tau_i), \\ m_j \in \vec{m}, v = \psi_{m_j}(\theta_{m_j}), \\ r_i(v) = true}} X^{z_i} \tag{19}$$

The scope of an object may encompass multiple regions of space which are detached from each other (e.g., multiple fires being localized by the street surveillance system). They are interpreted as distinct instances o_i of the same object type with a set of multiple unconnected regions $X^{o_i} \subseteq X^o$ as a result.

For each of the objects o_i resulting from this interpretation, an instance vector $\vec{v}$ is calculated for the corresponding scope X^{o_i} by applying the according methods, i.e., applying their calculation functions $m_j.\psi$ to a chosen set of sensors θ^{m_j}.

$$\vec{v}_{X^{o_i}}(t, \vec{x}) = \vec{\psi}(\vec{\theta}) = \begin{bmatrix} m_1.\psi(\theta^{m_1}) \\ \dots \\ m_q.\psi(\theta^{m_q}) \end{bmatrix}, \ \vec{x} \in X^{o_i} \qquad (20)$$

Each element v_i of this vector provides a value of type τ_i for the corresponding element z_i of the state vector in the object description of an object instance o_i in the scope X^{o_i}. The property value may also change within the scope over time and space. Therefore, it depends on time t as well as space $\vec{x}$. This vector allows monitoring the state of the physical object of interest and changes in its location within its observable scope.

III. Related Work and Discussion

This section discusses related work on incorporating environmental contexts into sensor capability models. Thereafter, the presented approaches are compared to our model.

A. Related Work

As mentioned in [4], the current state of the art is to utilize ontologies for describing the abilities of a sensor and the environmental context it observes (i.e., what it is measuring). In [5] and [6], different ontologies for the semantic specification of sensors are surveyed. They allow interpreting the sensor measurements with respect to their measurands and in which contexts their observations are made. These approaches are static in the sense that the measurands and environmental contexts are directly bound to sensors which does not take the motion of sensors into account. Additionally, the influence of contexts on the sensor capabilities is not made clear as the corresponding measurements are tagged but no adjustments to the specifications of a sensor's capabilities are made. Therefore, the developer has to manually infer the capabilities of the sensors depending on which contexts they are located in. Potential operating systems employing these techniques would therefore be obstructed in their ability to dynamically schedule devices suitably for the tasks at hand.

In [1], an approach to sensor virtualization is described which allows a set of applications to concurrently utilize a common set of sensors. The environmental contexts and capabilities of sensors are not taken into account in this model. Therefore, the choice of which devices to utilize during the execution of the application still lies with the programmer. This introduces room for errors as some sensors may be located in contexts that do not allow to measure the desired environmental entities.

In [7], a sensor virtualization method is presented. It allows the developer to declaratively specify a virtual sensor with respect to which behavior it has to implement. Based on their capabilities, changing physical devices are chosen during runtime for the execution of the application's tasks. This approach allows utilizing possibly changing sets of sensors and also accounts for the motion of devices. The contexts of sensors are not taken into consideration. Thus, an improper selection of devices may lead to an application utilizing data that does not suit the context for which it was developed.

In [2], virtual sensors are created such that each one wraps a physical device. A virtual sensor provides a service that can then be used by multiple different applications. Therefore, the sensors are bound to their respective applications such that taking a changing set of devices transparently into account is not feasible. Additionally, contexts are not considered which leads to similar challenges as described above.

In [8], an ontology is provided which allows the developer to create individual capability models for each sensor. These models are utilized to generate code for the distinct devices which makes their functionalities available to the system. This approach is most suited for systems with a static set of sensors as individual capability models have to be created and the corresponding code needs to be generated for each sensor. Transparently scheduling a changing set of devices is therefore only feasible in systems with an a priori known set of sensors. The importance of taking the sensor contexts into account is mentioned but not further elaborated on.

In [9], an abstraction for sensors and actuators is presented, called resource, which allows the programmer to declarative describe which devices are required by his application. If a device is not available, a new resource may be created by coupling a set of different devices which in combination are able to perform similar actions as the unavailable resource. Therefore, a transparent concurrent utilization of different sensors based on the task at hand is possible. The contexts of the devices are taken into account for the querying process but they are statically bound to sensors and actuators. Thus, the mobility of devices (i.e., changing contexts) is not considered which may lead to an erroneous allocation of mobile sensors or actuators depending on their change of position.

In [10] and [11], two approaches are presented which integrate the notion of contexts into the programming model. The developer binds parts of his application to different contexts based on which devices are available within them. Their models allow to transparently utilize different sets of sensors or actuators based on where they are located. Neither sensor capabilities nor how their environmental contexts influence them are taken into account, which leads to similar drawbacks as described above.

In [12]–[14], varying approaches for modeling contexts of sensor systems are surveyed. The examined propositions' primary focus lies on annotating sensor data such that they can be interpreted with respect to their environment. None of these approaches describe the influence of contexts on the capabilities of sensors. Rather, they enrich the sensor data with

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-106-0

context information such that the programmer has to make the correct choice of which contexts are of relevance and which sensor measurements have to be gathered within them. This introduces room for errors as the effect of contexts on the capabilities of sensors is not handled explicitly. Additionally, the developer may not be able to utilize all devices as a sensor located within a given context may still be able to provide measurements for locations outside of a context.

In [15], an approach to defining contexts onthologically is presented. The focus lies on documenting the impact of different contexts (i.e., developmental, behavioral, structural, and functional contexts) on the development of the CPS. This allows the programmers to consider challenges emerging from changes of the system's contexts during runtime. While the presented model is aiding developers in designing the system, it does not provide runtime support for transparently managing devices based on the context descriptions.

B. Discussion

None of the presented approaches discuss the influence of environmental contexts on the capabilities of sensors. Therefore, performing tasks like sensor fusion on measurements that may not relate to each other due to environmental constraints may create erroneous results. The virtualization of devices is affected similarly. The replacement of a failed sensor by one or more other devices is only feasible if their measurements are related.

Our model describes a solution to this challenge by taking the influence of environmental contexts into account. They are defined as regions in space that restrict the interpretable locations of sensor measurements. This allows to decide which measurements are related to each other. A context-aware virtualization of the devices is therefore enabled with respect to which devices possess the capabilities to perform a task.

For performance reasons, it may be required to restrain the definition of contexts in our model to discrete sets of locations in the form of predefined geometric shapes instead of arbitrary continuous regions in space (as in [11] for example). This allows the efficient computation of the required operations on the different sets of locations as described in Section II-D. The approaches discussed in [7], [10], and [11] provide performant implementations which enable sensor virtualization without taking contexts into account. Therefore, they are suitable as a starting point for implementing our model.

IV. Conclusion and Future Work

This paper presents a model for describing the influence of environmental contexts on the capabilities of sensors. This is necessary since a sensor's ability to observe its environment is strongly impacted by its surroundings. Contexts are defined as sets of locations such that they describe regions in space. They are viewed as constraints on the ability to interpret the measurements of a sensor for a given location. These constraints are effective at the edges of contexts, such that they influence the observable locations of sensors within or outside of the context. Our sensor capability and context model makes the influence of environmental contexts on the available sensors explicit.

The model is presented in the context of identifying physical objects based on an object description provided by the programmer. It allows to reason about the influence of choosing different sets of sensors on the coverage of the system space with respect to the object properties to be measured. This allows the OS to choose devices according to their ability to observe locations of interest. Therefore, a virtualization of sensors is achieved. This allows to transparently execute an application on alternating sets of heterogeneous devices while ensuring that locations of interest are observed.

For future work, we intend to enrich the model with further metrics for allowing an optimal choice of sensors based on their capabilities. The adaptation of contexts via exploration during runtime based on available sensor measurements is also a future goal. Additionally, an implementation and integration of the model into the presented programming model is intended.

References

[1] C. Mouradian *et al.*, "Network functions virtualization architecture for gateways for virtualized wireless sensor and actuator networks," *IEEE Network*, vol. 30, no. 3, pp. 72–80, 2016.

[2] P. Evensen and H. Meling, "Sensewrap: A service oriented middleware with sensor virtualization and self-configuration," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2009, pp. 261–266.

[3] M. Richter, T. Werner, and M. Werner, "A Programming Model for Heterogeneous CPS from the Physical Point of View," in *The Sixteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2022, pp. 1–6.

[4] N. Sahlab, N. Jazdi, and M. Weyrich, "Dynamic context modeling for cyber-physical systems applied to a pill dispenser," in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 1435–1438.

[5] G. M. Honti and J. Abonyi, "A review of semantic sensor technologies in internet of things architectures," in *Complexity*, 2019, pp. 1–21.

[6] M. Compton, C. A. Henson, L. Lefort, H. Neuhaus, and A. P. Sheth, "A survey of the semantic specification of sensors," in *CEUR Workshop Proceedings*, 2009, pp. 17–32.

[7] S. Kabadayi, A. Pridgen, and C. Julien, "Virtual sensors: abstracting data from physical sensors," in *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2006, pp. 586–592.

[8] O. Lemaire, K. Ohba, and S. Hirai, "Dynamic integration of ubiquitous robotic systems through capability model processing," in *SICE-ICASE International Joint Conference*, 2006, pp. 1207 – 1211.

[9] V. Tsiatsis *et al.*, "The sensei real world internet architecture," in *Towards the Future Internet: Emerging Trends from European Research*, 2010, pp. 247–256.

[10] Y. Ni, U. Kremer, and L. Iftode, "Spatial views: Space-aware programming for networks of embedded systems," in *Lang.s and Compilers for Parallel Comput.*, 2004, pp. 258–272.

[11] C. Borcea, C. Intanagonwiwat, P. Kang, U. Kremer, and L. Iftode, "Spatial programming using smart messages: design and implementation," in *24th Int. Conf. on Distrib. Comput. Syst.*, 2004, pp. 690–699.

[12] C. Bettini *et al.*, "A survey of context modelling and reasoning techniques," in *Pervasive and Mobile Computing*, 2010, pp. 161–180.

[13] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," in *IEEE Communications Surveys & Tutorials*, 2014, pp. 414–454.

[14] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp*, 2004, pp. 34–41.

[15] M. Daun and B. Tenbergen, "Context modeling for cyber-physical systems," *Journal of Software: Evolution and Process*, vol. 35, no. 7, p. e2451, 2022.

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-106-0