Towards a model-based approach for context-aware assistance systems in offshore operations

Nils Koppaetzky
Database and Internet Technologies
Dep. of Computer Sciences
University of Oldenburg, Germany
nils.koppaetzky@uni-oldenburg.de

Daniela Nicklas
Database and Internet Technologies
Dep. of Computer Sciences
University of Oldenburg, Germany
dnicklas@acm.org

***Abstract*—With the effort to increase renewable energy production, offshore wind power plants become increasingly important. Thus, there is a growing necessity to carry out offshore operations like construction or maintenance of wind power plants. Offshore operations are complex, risky, and therefore prone to accidents. A context aware assistance system might help to increase the safety during these operations. It could observe the status of safety relevant entities with sensors and support crews by displaying status information and triggering warnings in occurrence of critical situations. As the safety relevant entities and the critical situations depend on the executed operation, the assistance system needs to be based on a dynamic and flexible context model. This paper introduces an approach to design an easy adaptable, extendable and deployable context model for an offshore operation assistance system.**

***Keywords*-context modeling; situation recognition; offshore; safety; assistance system;**

I. Introduction

Offshore operations, like construction or maintenance of wind power plants, are complex, risky, and therefore prone to accidents. Within the project "Safe Offshore Operation" (SOOP) [1] we develop tools and methods for risk analysis, human behavior modeling, operation planning [2], training, situation recognition, and sensor technology [3] to ease the planning and execution of offshore operations for carriers and crews. One particular goal of SOOP is to increase the safety of offshore operations by observing the status of safety relevant entities by a context aware assistance system. It assists the crew by displaying an overview of the situation and by triggering warnings in occurrence of critical situations.

The definition of safety relevant situations, the involved actors, and the state of environmental entities is the result of a planning phase that includes a formal risk analysis. The results are also used to identify critical and relevant non-critical situations for the assistance system. State-of-the-art assistance systems are highly specialized. They are designed and optimized for a certain fixed set of sensors, entities, and situations that should be recognized during a certain supported operation. As we want to support many operations with only one system, and to be extendable to new sensors and new situations, it is not feasible to hardwire the sensors and the software to derive situations. Instead, we define a context model that contains all relevant entities and their relations. Based on that, we describe and define the situations of interest for the assistance system.

Within the field of offshore operations, sensors, entities, and important situations depend on the executed operations and might vary between them. The same operation could be carried out, by different types of ships, by different crews, with different sensors, etc. As we do not want to design a separate context model for each possible configuration of entities and each operation, a system approach with a fixed, predefined context model cannot be used. During the actual operation the assistance system derives knowledge and situations from raw sensor data and prior knowledge by using a dynamic implementation of the context model (runtime context model), which is updated continuously. The context model itself is defined as a result of the planning phase and realized as ontology (ontology context model).

A typical offshore operation is a cargo loading operation. When lifting large or heavy loads, the crane operator is supported by a lift supervisor, who observes the loading procedure from the cargo deck and communicates with the crane operator. During the operation, the assistance system should warn, e.g., when

- a person is to close to the lifted cargo or under it
- the lift supervisor does not keep sufficient distance to the cargo

The relevant entities within the context model for these situations would be *LiftSupervisor, CraneOperator, Cargo* and *SafetyArea* with their parameters *absolutPosition, relativePosition, geometry* and *size*.

In this paper, we present an approach for flexible assistance systems for offshore operations. To mediate between the high level description of safety relevant situations and low level sensor data sources that may even change between operations, we introduce a context model that shall significantly reduce the development effort for such systems.

978-1-4673-5077-8/13/$31.00 ©2013 IEEE

Our multi-layered approach is based on an ontology (the ontology context model). For the recognition process of a given operation, ship, and sensor configuration, an adapted instance of the ontology context model is transformed and deployed as an executable query plan for a data stream management system. This executable query plan represents the runtime context model of the assistance system. The approach exploits both the advantages of ontologies (like a machine-readable knowledge representation, reasoning, and consistency checks) and data stream management systems (like easy configuration, easy adaptation, and highly dynamic data processing).Since our work is at an early stage, we present our multi-layered system approach, show examples for the ontology context model and the implemented runtime context model, and outline our evaluation plans.

II. Related work

In the last years, many approaches for context modeling, reasoning, and management have been developed [4] , [5]. While a multitude of frameworks and middleware systems have been proposed to support context awareness and manage context models (e.g., the CML framework [6], ACoMS [7], Nexus [8], to name a few) none of these approaches is designed to cope with highly-dynamic context models as needed in offshore operations. Their sensors are updated within seconds, and the application adaptation frequencies are typically much lower. This even holds for the work in [9], where the Care framework is used to target continuous streaming applications: the application is highly dynamic, the context model management is not. In SOOP, a typical sensor update frequency is 2 Hertz, and the assistance system has to react within milliseconds to increase safety by triggering warnings at present critical situations.

In the model-based approach of [6], a graphical notation called Context Modeling Language (CML) is used to design a context model. On top of this context model, situations are defined using rules. Out of these models, the context management is generated by the framework, namely the database schema and queries to retrieve context information. This is very similar to our approach; however, instead of a database management system, we build upon a data stream management system which supports active data sources, temporal patterns and context information with high update rates. In the past years, data stream management systems have evolved from pure research prototypes (like Aurora [10], or PIPES [11]) to more and more commercial products [12], [13]. Other related work covers the processing of location-based queries [14] or spatio-temporal queries [15] using data stream management systems. Recently, the Nexus project has extended their middleware to cope with context data streams [16]. However, none of these approaches can mediate a dynamic context model between fused data streams.

The approach of model-based development of context-aware applications introduced in [17] is comparable to our approach. Like in [17], one of our goals is the joint development between domain experts, and we assume the end user not to be involved in the development process. However in SOOP, the model-based approach is used to reduce the effort in developing and configuring an assistance system based on results of planning and risk analysis. Our aim is to provide the assistance system with a customized runtime context model, not to develop a multitude of context-aware applications. The approaches discussed in [18] provide solutions that allow to involve end users with different experience levels in the development process. In SOOP, the development is done by a team of domain experts, the end users only use the developed tools and models to customize the assistant system or they use a already customized assistant system.

III. Approach

In offshore operations, we face the problem that every supported operation commonly needs its own assistance system observe the status of the operation, including the recognition of critical situations. These assistance systems would have to be customized to the specific sensor configurations, ship type, sensor technologies, crew configuration, and many other entities included in a certain operation. The resulting heterogeneity and effort in development and configuration, plus the inflexibility in using new sensor technology would make these systems impractical and unusable. To cope with the heterogeneity in applications, operations, entities, and sensor technology, and to reduce the effort of development, we propose an approach with a multi-layer middleware between sensors and applications, which is separated into an ontology context model and a runtime context model. All assistance systems like a cargo operation assistance system, a personal transfer operation assistance system, etc., can be realized by one configurable generic assistance system for offshore operations.

A. System design

Figure 1 shows the layers of our approach and the relation between the ontology context model and the runtime context model.

Ontology context model: The ontology context model is defined as a result of the development and planning phase, which includes a formal risk analysis. It contains all known entities and their parameters, like the ship types, sensor descriptions, actors, or processes. Its layers are described in detail below. For a certain operation a customized and adapted instance of the model is transformed into a runtime context model and deployed to the assistance system.

Assistance system: The assistance system contains all components which are necessary to recognize and react on context and situations. It consists of sensors, context recognizers which partly belong to the runtime context

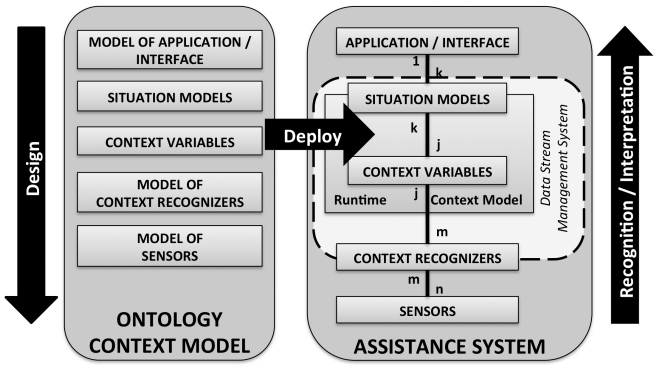

Figure 1. System diagram: Abstraction of data increases over the layers

model, the runtime context model, and the context and situation consuming application and its interfaces.

We design the system layers displayed in figure 1 following a situation-centered context modeling approach. Each layer appears in the ontology context model as well as in the assistance system. Note that some real world components an only appear as modeled entities in the ontology context model. The layers are described as follows:

Application / Interface: The application and its Interfaces are software components or stand alone programs that consume the derived context and situations and react on it. Together they realize the assistance system for a given operation. One example would be a user interface which displays the overall situation and warnings on a large screen. If an operation shows abnormal situations or changes to a critical situation it should warn the crew, for we observe both, the normal flow of operations as well as critical situations that should be avoided. Other interfaces could be realized by mobile devices or ambient devices.

Situation description: The situation models consist of a situation description and a situation definition. Analyzing the cargo loading operation described in chapter I, we find three *k* situations that are safety relevant, described such as:

PersonToCloseToLiftedCargo: A person is to close to or under the lifted cargo.

LiftSupervisorToFarFromCargo: The LiftSupervisor who has to follow the cargo does not keep sufficient distance to it.

LineOfSightInterrupted: The cargo interrupts the line of sight between CraneOperator and LiftSupervisor.

Situation definition: The *k* safety relevant situations have been identified by a predated risk analysis. Such a situation is defined by a set of values of real world parameters of a set of *j* real world entities. For the cargo loading operation the situation PersonToCloseToLiftedCargo is critical and would be defined as follows:

$$\begin{aligned} \text{cargolifted} &= \text{"true"} \\ \text{distance}(\text{cargo}, \text{person}) &< \text{safety distance} \end{aligned}$$

Context model of cargo operation: To define the situations as a condition over a fixed set of measurable parameters of real world entities, we describe these entities, their possible parameters, and their relations. In this first step, we use the Unified Modeling Language 2.0 (UML 2.0) to illustrate the ontology context model (Figure 2). We allow multi-inheritance to generalize our entities. For example, the cargo is both a resource and a physical entity, which means an entity with physical representation. In our example, all entities are physical entities and the sensors can gather information referring to the physical character of the entities. The set of relevant parameters of a physical entity are *absolutPosition, relativePosition, geometry* and *size*, with the relative position referring to another physical entity (self-relation arrow).

Context variables: The *j* context variables are relevant parameters of an entity in our system, which are measureable by sensors or derivable from prior knowledge. They are used to define the situations described above.

Context recognizers: The *m* context recognizers are programs or executable data stream processing plans that derive values of context variables from raw sensor data, or from values of context variables that are already derived. As the value derivation of context variables can be very complex, the context recognizer can vary highly in their complexity. Conceptually context recognizers are comparable to context widgets introduced in [19].

Sensors: The *n* sensors are used to perceive and observe the environment by measuring data. In our case a sensor network using wireless ultra wide band technology is used to determine the positions of entities.

B. Recognition and interpretation

The process of runtime recognition of situations with the introduced system follows Figure 1 bottom up. It increases the degree of abstraction, compression, and interpretation of data with every layer. Raw data, delivered from the sensors is used to derive the context variables with the help of context recognizers. A set of values of context variables is abstracted into situations, which are transmitted to the assistance system and could trigger warnings. Hence, a large number of raw sensor data are mapped to few domain concepts during the recognition process.

IV. Implementation

The ontology context model is implemented as a result of a planning phase, where domain experts provide information on entities, situations, etc. and there relations. To enable highly dynamic runtime recognition of situations during operations, we transform an adapted instance of the ontology context model into an executable query plan within a data stream management system (DSMS). A DSMS is a middleware that allows to manage and process continuous

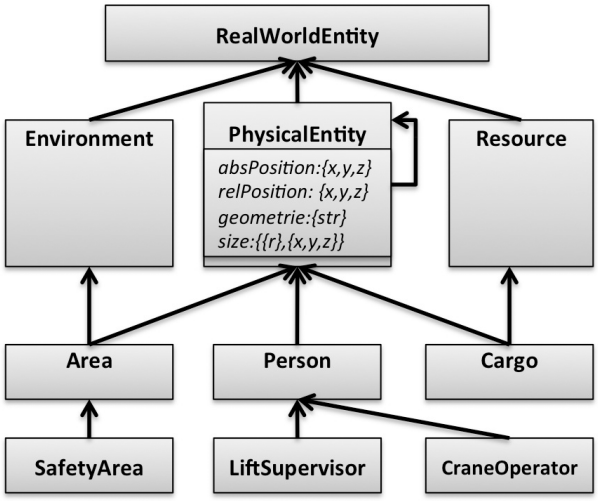

Figure 2. Context model for the cargo loading operation described in chapter I, modeled as a class diagram with UML 2.0

data streams from heterogeneous data sources (e.g., sensors). Similar to a data base management system, queries can be stated and mathematical expressions applied to manage, filter, and process the data. However, in a DSMS these queries are registered in the system and executed continuously. After registration, a declarative query is transformed into a so-called query plan. In many cases the query languages for DSMS are extensions or adaptations of the Structured Query Language (SQL). The query plan resulting from the transformation is the runtime context model for a particular operation with a particular set of entities (sensors, ship, etc.). Its sources are the sensors and it sinks represent the recognizable situations. A situations is recognized when data, which are processed and filtered by the DSMS, appear at a sink. This event can be used to trigger warnings on the interface or to trigger other actions or applications.

A. Implementation within Odysseus

For our implementation, we use the open source DSMS "Odysseus" [20] and its query language PQL, which both were developed at the University of Oldenburg. As an example, we show the implementation of the situation *PersonToCloseToLiftedCargo* (see chapter III-A) for the cargo loading operation (see chapter I) in PQL with the following set up:
Two mobile and at minimum four ultra wide band (UWB) sensors observe the locations in the field (for details see [3]). One of the mobile sensors is attached to the lift supervisor the other to the cargo, both at the center of the entities. The cargo is a container (box) of 10 m length, 3 m width and 3 m height. The safety distance between lift supervisor and container should be at least 3 m. The cargo is lifted the entire time. The inaccuracy of the sensors is 0.5 m radial. All physical units are SI system units. With these assumptions the PQL query is defined as follows:

Listing 1. Code example of a situation recognizer implemented in PQL

```
1  #PARSER PQL
2  #TRANSCFG Standard
3  #DEFINE Sensor1 'LiftSupervisor'
4          ...
5  #DEFINE SensorError 0.5
6  SensorData = ACCESS({source='SensorData', adapter='
       GenericPull', input='StringSocket', transformer
       ='LineToString', dataHandler='ScaiTuple',
       options_old=['host:localhost','port:7000', '
       charset:UTF-8','delimiter:;','keepDelimiter:true
       '], schema=[['PosX1','Double'],['PosY1','Double
       '],['PosZ1','Double'], ['PosX2','Double'],['
       PosY2','Double'],['PosZ2','Double'],] })
7  relPos = map({expressions = ['PosX1-PosX2','PosY1-
       PosY2','PosZ1-PosX2']}, SensorData)
8  newPos = RENAME({aliases = ['PosXRC', 'PosYRC','
       PosZRC']}, relPos)
9  dist = map({expressions = ['sqrt(PosXRC^2+PosYRC^2)
       ']},newPos)
10 distance = RENAME({aliases = ['distLSC']}, dist)
11 filter = SELECT({ predicate=RelationalPredicate('
       distLSC <=sqrt(2)*sqrt(CargoLength^2+CargoWidth
       ^2)/2+CargoSafetyDistance+SensorError*2') },
       distance)
```

Line 1 defines the parser language as PQL, for Odysseus can also process other query languages. Line 2 sets a standard for internal data handling, e.g., which meta data are attached to the incoming data. From line 3 to line 5, prior knowledge is defined as values of variables. Note that this block has been shortened in the code, other definitions here would be e.g. (the size of the cargo, CargoLength, CargoWidth, CargoHeight, ...). The statement in line 6 represents a context recognizer for the location of the sensors in the same reference system in co-coordinates. Line 7 registers a context recognizer for the relative location of one sensor to another. It works for all co-coordinates within the same reference system.
Line 8 renames the variables. That is necessary in Odysseus for further processing after applying the map operator. Line 9 contains the calculation of the distance between lift supervisor and cargo. A renaming is done in line 10 again for same reasons as in line 8. The statement in line 11 filters the values where the distance of the lift supervisor and the cargo fall below the safety distance. As one can see this is the part of the query that recognizes the critical situation *PersonToCloseToLiftedCargo*. Due to our assumptions we only need to account for the distances between lift supervisor and the cargo, for they are the only observed entities. The z-coordinate is irrelevant for detecting the situation. The safety distance has to be modified by adding the radial sensor errors as well as the half diagonal over the cargo length and cargo width due to the inability of detecting the cargos orientation. Note that this is an example of an implementation for a simple situation only based on location observation. More complex situations with large numbers of entities to derive complex relations between them would result in a more complex query plan.

B. Reconfiguration and maintenance

One advantage of using a DSMS to implement the runtime context model is that queries can be installed and de- or reinstalled while the system operates. This allows an easy adaptation of the system to the overall situation, making it highly flexible, for its installed instance depends on the situational recognition itself. We show the ease of adaptation and extension of queries by two examples:
1. If our sensor network would deliver GPS (Global Positioning System) coordinates instead of co-coordinates, we only need to add a context recognizer with the ability to transform GPS coordinates into a co-coordinate reference system to the query.
2. To add the situation *LiftSupervisorToFarFromCargo* to the example query above, we just add

```
#DEFINE MaxDistLiftSupervisorCargo 7
```

to the "DEFINE" block and to replace line 11 with the following statement:

```
filter = SELECT({ predicate=RelationalPredicate('distLSC
    <= sqrt(2)*sqrt(CargoLength^2+CargoWidth^2)/2+
    SecurityDistance+ Sensorerror*2') AND distLSC >=
    MaxDistLiftSupervisorCargo}, distance)
```

Then we register the resulting query to the system. Odysseus would re-use the shared parts of the query plan, i.e., the context recognizer. Note that such extensions could also be done semi-automatically depending on the derived situations during the assistance system operates.

V. Evaluation Plans

To evaluate our approach, some prior field-tests with the sensor technology [3] and simulations have already been made. These tests, although in an early stage support our approach. Different modeling languages will have to be evaluated against one another (e.g. UML 2.0, OWL or CML) for the capability of fitting all the requirements of the assistance system. One focus here will be the connection of the planning- and risk-analysis tools with the ontology and the context model. More sophisticated field test and simulations with different tools (e.g. MA3DS or SIAFU) will be used to evaluate the systems usability, adaptability, reliability, accuracy and latency. Model checking methods will be applied during the entire evaluation process to check the systems models and meta models with respect to consistency. Within a usability-test a prototyping process will be evaluated. Finally domain experts will test a prototype system under real conditions.

VI. Conclusion

In this paper we introduced an approach for a dynamic and flexible assistance system for offshore operations. It is based on a multi-layered context model which is realized as ontology (ontology context model) and deployed as query plan (runtime context model) within a DSMS for the actual recognition process. The advantages of our approach are:

- sensor technology independent definition of situations due to definition on context variables
- simple reconfiguration of sensor technology for only context variable values must be delivered
- simple integration of new sensor technology by implementation of new context recognizers
- possible use of multiple sensor types to derive values of a certain context variable
- easy analysis of the mapping between sensors and situations to find appropriate sensor configurations
- using both the advantages of ontology modeling based design and the DSMS technology

The implementation of the runtime context model as an executable query plan (chapter IV) allows taking full benefit of the advantages of DSMS like processing of highly dynamic data sources, and flexible redesign and exchange of queries during the system operates. With some initial test were already performed, there are already good reasons to expect the approach to be successful for a large number of offshore operations.
In the SOOP project (see chapter I), tools and methods for risk analysis and operation planning are developed. These tools will be used for planning an operation and relate the critical situations to their corresponding operations. For now, the data stream queries introduced in chapter IV are implemented manually based on results of the risk analysis and operation planning. As all these tools are based on ontologies that can be transformed into another and joined, they are also the backbone of our ontology context model. A semi-automatic generation and deployment of data stream query plans, and therefore the runtime context model would be feasible and require a meta model to be developed. The automatic adaptation of the runtime context model during the system operates, depending on the recognized situation and operation, would be of advantage. The requirement of semi-automatically code generation and extensions to the system might introduce the need of using another modeling language to describe the ontology context model. All results will have to be evaluated with state-of-the-art methods.

Acknowledgment

The authors would like to thank the entire SOOP team for the good collaboration. This work was founded by the European Regional Development Fund (ERDF) and the federal state Lower Saxony.

References

[1] *Project SOOP: Safe Offshore Operations*, Deutsche Gesellschaft für Ortung und Navigation (DGON). Deutsche Gesellschaft für Ortung und Navigation e.V., 10 2012.

[2] R. Droste, C. Läsche, C. Sobiech, E. Böde, and A. Hahn, "Model-based risk assessment supporting development of HSE plans for safe offshore operations," in *Formal Methods*

for Industrial Critical Systems, ser. Lecture Notes in Computer Science, M. Stoelinga and R. Pinger, Eds. Springer Berlin Heidelberg, jan 2012, no. 7437, pp. 146–161.

[3] T. Wehs, M. Janssen, C. Koch, and G. von Cölln, "System Architecture for Data Communication and Localization under Harsh Environmental Conditions in Maritime Automation," in *IEEE 10th International Conference on Industrial Informatics*, 2012.

[4] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[5] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive and Mobile Computing*, vol. 8, no. 1, p. 3666, 2012.

[6] K. Henricksen, "A framework for Context-Aware pervasive computing applications," Ph.D. dissertation, School of Information Technology and Electrical Engineering, The University of Queensland, Sep. 2003.

[7] P. Hu, J. Indulska, and R. Robinson, "An autonomic context management system for pervasive computing," in *Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), 17-21 March 2008, Hong Kong*. IEEE Computer Society, 2008, pp. 213–223.

[8] M. Grossmann, M. Bauer, N. Honle, U. P. Kappeler, D. Nicklas, and T. Schwarz, "Efficiently managing context information for Large-Scale scenarios," in *Proceedings of Pervasive Computing and Communications*. IEEE Computer Society, 2005, pp. 331–340.

[9] C. Bettini, D. Maggiorini, and D. Riboni, "Distributed context monitoring for the adaptation of continuous services," *World Wide Web Journal*, vol. 10, no. 4, pp. 503–528, 2007.

[10] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *The VLDB Journal*, vol. 12, no. 2, pp. 120–139, 2003.

[11] J. Krämer, "Continuous Queries over Data Streams-Semantics and Implementation," Ph.D. dissertation, Philipps-Universität Marburg, 2007.

[12] B. Gedik, H. Andrade, K. Wu, P. Yu, and M. Doo, "SPADE: The System S declarative stream processing engine," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. Vancouver, Canada: ACM, 2008, pp. 1123–1134.

[13] Realtime Monitoring GmbH, "RTM Analyzer," 2010. [Online]. Available: http://www.realtime-monitoring.de/pdfs/WhitePaper_RTM_Analyzer_DE.pdf

[14] X. Huang and C. S. Jensen, "Towards a streams-based framework for defining location-based queries," in *Proceedings of STDBM*, 2004, pp. 78–85.

[15] M. F. Mokbel and W. G. Aref, "SOLE: scalable on-line execution of continuous queries on spatio-temporal data streams," *The VLDB Journal*, vol. 17, no. 5, pp. 971–995, Apr. 2007.

[16] N. Cipriani, M. Eissele, A. Brodt, M. Grossmann, and B. Mitschang, "NexusDS: a flexible and extensible middleware for distributed stream processing," in *Proceedings of the 2009 International Database Engineering & Applications Symposium*. Cetraro - Calabria, Italy: ACM, 2009, pp. 152–161.

[17] M. Wojciechowski and M. Wiedeler, "Model-based development of context-aware applications using the MILEO-context server," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, 2012, pp. 613–618.

[18] B. Guo, D. Zhang, and M. Imai, "Toward a cooperative programming framework for context-aware applications," *Personal Ubiquitous Comput.*, vol. 15, no. 3, pp. 221–233, Mar. 2011.

[19] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: aiding the development of context-enabled applications," in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, 1999, pp. 434–441.

[20] H.-J. Appelrath, D. Geesen, M. Grawunder, T. Michelsen, and D. Nicklas, "Odysseus: a highly customizable framework for creating efficient event stream management systems," in *DEBS*, F. Bry, A. Paschke, P. T. Eugster, C. Fetzer, and A. Behrend, Eds. ACM, 2012, pp. 367–368.