Towards Context Modeling in Space and Time

Christian Piechnick, Georg Püschel, Sebastian Götz,
Thomas Kühn, Ronny Kaiser, and Uwe Aßmann

Software Technology Group, Technische Universität Dresden,
Nöthnitzer Str. 46, 01187 Dresden, Germany
{christian.piechnick,georg.pueschel,sebastian.goetz1,thomas.kuehn3,ronny.kaiser,uwe.assmann}@tu-dresden.de
http://st.inf.tu-dresden.de

Abstract. One of the main problems in software development for service robots is to create systems that reliably behave as intended, even though the real field of application and the concrete user requirements are unknown during design time. Consequently, the software controlling service robots has to be aware of its environment and has to adapt its behavior accordingly. A model representing environmental data is called a context model. Appropriate context models currently lack means for modeling temporal and spatial information simultaneously. While it is important to reason about historical context data for most of the Self-Adaptive Systems, there is an increasing need for treating the temporal dimension of context models as first-class-citizen. In this paper, we propose a graph- and role-based context model (GRoCoMo), which includes expressive means for describing time and location. A query language enables for reasoning on current and historical data, as well as future trends. A manipulation language enables the specification of rewrite rules for updating context models based on situations detected within the context.

Keywords: Context Modeling; Context Management; Context-Awareness; Temporal Context; Context History;

1 Introduction

A service robot is a reprogrammable, sensor-based, mechatronic device which performs useful services to support human activities [8]. In contrast to production robots, where the operating environment as well as all other influencing factors are known before deployment, the application sites of service robots are unknown. This information can only be gathered during runtime. Therefore, the software system controlling the service robot has to adapt its behavior dynamically. Such a system is called a Self-Adaptive System (SAS). One example of adaptive behavior within robotic software is the path planning of mobile robot platforms [4]. When a robot has to move to a target position, the robot's motion model (e.g., differential steering), as well as the environment (e.g., crowded areas), influence the planning strategy. Figure 1 shows an example path planning problem with two different strategies. The first strategy, *Shortest Path* (solid

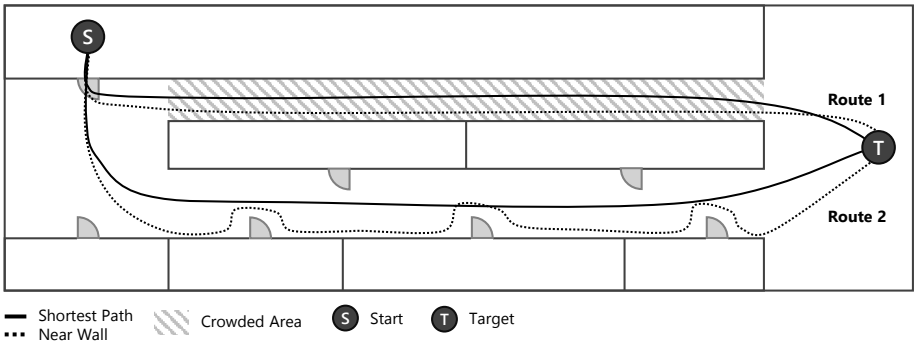

Fig. 1. Example for alternatives in global path planning

lines), calculates the shortest route from the starting location (S) to the target location (T) and provides two different alternatives. The other strategy, *Near Wall* (dashed lines), calculates routes that lead along walls. The grey dashed area is usually used by many persons and therefore, marked as "crowded". When the navigation algorithm decides to use the upper floor, the *Near Wall* strategy is the better option, since the probability to get in the way of humans is decreased. On the other hand, it might be disadvantageous on the lower floor, where the robot potentially has to drive around many open doors. Hence, the decision should be made at runtime. An inherent property of all SAS is that they are implementing a variant of the MAPE-K loop, first introduced by IBM [7]. The MAPE-K loop consists of four phases: in the *(M) Monitor* phase environmental data is gathered and processed and, then, interpreted in the *(A) Analyze* phase. The *(P) Plan* phase investigates the need for reconfiguration and creates reconfiguration plans accordingly, which are applied in the *(E) Execute* phase. All phases share a *(K) Knowledge Base*, which manages relevant information guiding the adaptation process. An essential part of this knowledge base is information about the execution environment (e.g., crowded areas in the upper example), i.e., the **Context Model**. Because different domains have varying requirements w.r.t. context modeling and management, various different modeling approaches for context information were developed during the last decades. Nevertheless, recent context modeling approaches are not sufficient to handle crucial aspects for the domain of service robots. Namely, reasoning on dynamic collaborations like in robotic applications, requires more expressive means to cover time and location in a processable manner. To address this problem, we present an extended graph-based context model with time and location as first class citizens and, thus, extended means for modeling and managing temporal and spatial context data for robotic applications.

This paper is structured as follows. In Section 2, we give an overview on context modeling and outline important properties of context models. In Section 3, we discuss relevant context modeling approaches and their suitability w.r.t. the

identified properties. We present our approach in Section 4 and discuss our prototypical implementation in Section 5. Finally, Section 6 presents our conclusion and future work.

2 Context Modeling and Management

To enable an application to adapt itself to changing environmental conditions, information about the environment must be gathered and analyzed. For this purpose, a variety of approaches have been developed, to tackle the specific requirements in different application domains of SAS [2]. Zimmermann et al. identified six different modeling elements of context models [17]:

Z1 - Entities: An entity can be any real or virtual thing that is of interest for the adaptation process (e.g., user, device, application, location, etc.).
Z2 - Individuality: The individuality encompasses any information that can be observed about an entity (e.g., dynamic and static properties, etc.).
Z3 - Relationships: A relationship expresses a semantic dependency between two entities. Zimmermann et al. distinguish between social-, functional-, and compositional relationships.
Z4 - Activities: The activities dimension encompasses any information about an entity's past, present and future needs, goals, tasks and plans.
Z5 - Time: Statements in a context model often have a temporal dimension. Time can be expressed using time zones (e.g., Central European Time) or virtual times (e.g., milliseconds after system start). Furthermore, overlay models can be used for abstraction (e.g., working hours, weekends, etc.).
Z6 - Location: Since most of a context model's elements represent objects from the physical world, which are arranged spatially, location is a major aspect of context information. The location dimension can include real or virtual locations (e.g., IP address in a network). Those locations can use absolute, relative, or symbolic location models.

Strang et al. [13] identified six different types of context modeling approaches: (1) Key-Value Models, (2) Markup Scheme Models, (3) Graphical Models, (4) Object Oriented Models, (5) Logic Based Models, and (6) Ontology Based Models. Depending on the specific requirements of the application domain, different advantages and disadvantages can be observed. They evaluated those types of context models regarding their ability to (a) be composed in a distributed computation environment, (b) the richness and quality of information, (c) the ability to handle incompleteness and ambiguity, (d) the level of formality, and (e) their applicability to existing environments. Considering those properties, Strang et al. conclude that ontologies are the best-rated modeling type, while Key-Value Models are the worst-rated. On the other hand, the construction and management of Key-Value Models is much simpler and the performance of analysis scales much better for simple requests. For the domain of service robots the properties (b), (c) and (e) are crucial because of the robots complex and unknown execution environment. The properties (a) and (d) become important

when the sensors (e.g., temperature sensor) and actuators (e.g., door opening) are distributed across the environment, and, thus, multiple computational units have to share knowledge based on a shared interpretation. Hence, according to the provided evaluation, ontologies should be used for the modeling of contextual information in the domain of service robots. Furthermore, a context model for service robots should include the modeling elements *Z1 - Z6*, according to Zimmermann et al. [17].

3 Related Work

The research area of SAS is still very popular, resulting in thousands of publications each year. This is also true for research on context modeling and management. For our related work research, we searched for papers published between 2000 and 2013 and containing the words "*context model*" in their title, using Google Scholar[1]. The result of the indicated query was a set of 3469 papers. We filtered the result-set manually to exclude publications that were not intended for the application in SAS. The result was a reduced set of 1228 manually filtered papers. Among them, we investigated five context model survey papers [1–3,6,13]. We have chosen six representative context model publications, which consider the dimensions *time (Z3)* and/or *location (Z4)*. Four of those papers [5,12,14,15] were chosen based on the description in the context model surveys. Because the latest survey was published in 2010 by Bettini et al. [2], we have selected two additional publications [9,16], published between 2010 and 2014. We have investigated their modeling capabilities w.r.t. the properties stated in Section 2. The results are summarized in Table 3.

In 2003 Strang et al. proposed the ontology-based context model **CoOL** (Context Ontology Language) [14]. CoOL provides the concept of an "Entity", while type information (e.g., Person, Place) must be expressed using the individuality dimension. *Individuality (Z2)* can be modeled using "Aspects" with different "Scales". *Relationships (Z3)* can be expressed using facts. Even though they show that the *time, place and activity dimensions (Z4 - Z6)* can be treated as an aspect as well, they do not provide a special interpretation semantic for time-bound, historical, location-, or activity-specific data.

Gu et al. presented an ontology-based context model for their service-oriented context-aware middleware **SOCAM** in 2004 [5]. They support several types of entities (e.g., Device, Network) as well as predefined and user-defined properties and relationships (*Z1 - Z3*). Activities are also treated as first-class-citizens (Z4). Time (*Z5*) is partially considered, but only as start and end. The model provides special nodes for locations (i.e., in- and outdoor locations) but does not show how locations can be related.

Wang et al. proposed the CONtext ONtology (**CONON**) [15] in 2004, by extending the SOCAM context ontology. In contrast to the previous model they provide means for describing location (*Z6*) in a more fine-grained manner and

[1] Google Scholar: http://scholar.google.de/, visited 20.05.2014

	CoOL	SOCAM	CONON	MUSIC	ERMHAN	CACOnt
(Z1) Entities	(+)	+	+	(+)	+	+
(Z2) Individuality	+	+	+	+	+	+
(Z3) Relationships	+	+	+	+	+	+
(Z4) Activities	(–)	+	+	(–)	+	+
(Z5) Time	(–)	(–)	(–)	(–)	–	–
(Z6) Location	(–)	(+)	+	(–)	+	+

Table 1. Evaluation of the related models w.r.t. to their modeling capabilities.
(- not considered, (-) partially considered, (+) implicitly provided, + fully provided)

explain how those locations can be related to each other, to create hierarchical location models.

Reichle et al. described an ontology-based context model for the **MUSIC** project in 2008 [12]. Like the CoOL ontology, they provide an abstract type `Entity` which can be categorized using special type attributes (*Z1*). The model provides means for describing attributes and relationships (*Z2 and Z3*), but does not treat activities as special entities (*Z4*). Hence, activities can only be modeled by creating used-defined activity type attributes. The model contains basic types, such as `DateTime` or `GPS-Coordinate`, but does not provide a first-class-citizen interpretation semantics for time and location (*Z5 and Z6*).

In 2011, Paganelli and Giuli presented a context model for the **ERMHAN** service platform for Ambient Assisted Living (AAL) scenarios [9]. The model provides means for describing several entity types, attributes, relationships and activities (*Z1 - Z4*). Furthermore, they consider several types of interrelated locations (*Z6*), but do not consider time (*Z5*) within the model. They only consider time externally by tracking the change of context values. Based on the type of the changed value, they interpret a time-bound sequence of values.

In 2013, Xu et al. presented the Context-Aware Computing Ontology **CACOnt**. It predefines several types of entities, properties, relationships and activities (*Z1 - Z4*). The authors extensively investigate the location dimension (*Z6*), by providing different levels of abstraction for the specification of an entities location (e.g., GPS, location hierarchies). They do not consider the time dimension (*Z5*). Thus, a CACOnt model only provides information on the current context state. However, like in every model with extensible attributes, it is possible to express time information using attributes with a custom interpretation logic.

As shown in Table 3, the presented context models provide means for modeling the dimensions of entities, individuality, and relationships (Z1-Z3). The activity dimension (Z4) is either provided, directly or can be modeled separately, using an extensible entity-model. The most recent works consider the location (Z6) as an essential part of a context model, and, thus, provide means

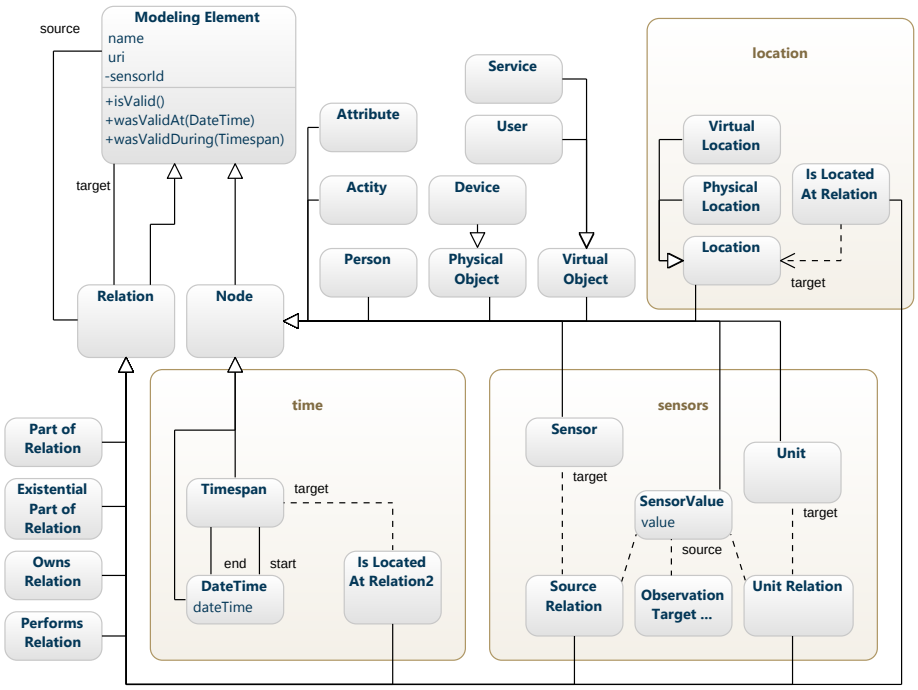

Fig. 2. The GroCoMo-Core metamodel.

for handling spatial information as a first-class-element of context models. The time dimension (Z5), however, is considered important in state-of-the-art literature, but current context models do not provide explicit modeling elements to handle time appropriately.

4 Context Modeling in Space and Time

In this section, we present our Graph- and Role-based Context-Model (**GRoCoMo**), a context model supporting all modeling dimensions stated in Section 2.

4.1 Structure

As depicted in Figure 2, the context model consists of `Nodes` and `Relations`. Both, `Node` and `Relation` inherit from the abstract type `Modeling Element`. Each element has a `name`, a `sensorId` to identify values created by the same sensor and a unique resource identifier (URI), to identify the individual element. A `Relation` connects exactly one `Source Element` to exactly one `Target Element`. Both, source and target, are of the type `Modeling Element`. Hence, it

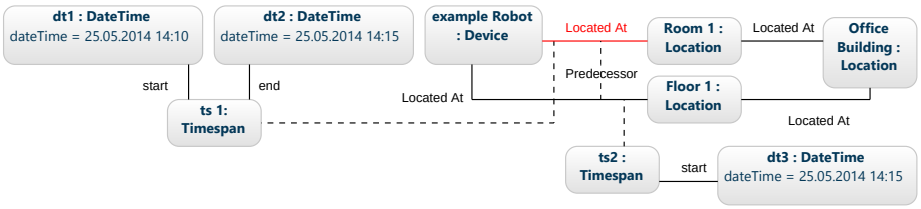

Fig. 3. Example model for time and location information representation.

is possible to define relations on relations. A node represents an entity of the context model. The model provides predefined node types (e.g., Person). However, the metamodel can be extended with domain-specific node types by subclassing. A `Relation` represents a typed, complex relationship between two entities. Furthermore, each relationship can contain several directly assigned attributes.

4.2 Handling Time

To cover temporal aspects, `Timespans` can be assigned to each modeling element (i.e., nodes and relations), to state when the validity of a modeling element started and stopped. Each modeling element with a validity timespan that has no associated end-time, is considered valid at the current time. By default the validity of a modeling element starts when it is created and can be invalidated by assigning an invalidation time. Because each element can have multiple validity times, a previously invalidated node or relation can be re-validated again. Each modeling element may have a `Predecessor Relation` to another modeling element of the same type that was replaced by the respective element w.r.t. its validity. In the scenario described in Section 1, the robot moves from a starting location `S` to a target location `T`. While the robot moves, a node, representing the robot, contains an outgoing `LocatedAt` relation. As shown in Figure 3, when the robot moves from `room1` to `floor1` the first `LocatedAt` relation is invalidated and replaced by a new relation. Because the invalidated relation is not deleted from the model, it is still accessible and can later be analyzed (e.g., by creating motion profiles). Furthermore, different representations of "time" can be modeled. As shown in Figure 3, date and time combinations can be represented as absolute timestamps in a given calendar. Hence, it is important to have an associated location to every timestamp representation, which is inferred in the provided example, because the timespan is assigned to a `LocatedAt` relation. Furthermore, other representations of time (e.g., weekdays or holidays) can be modeled and assigned to nodes and relations.

4.3 Handling Location

Locations are represented by special `Location` nodes (see Figure 2). The model separates physical (e.g., the main station) or logical locations (e.g., a folder

in a file system). Physical locations can further be divided into sub-symbolic (e.g., GPS coordinates) or symbolic (e.g., Dresden Main Station) locations. To relate an entity to a location node, the GRoCoMo metamodel provides a generic `LocatedAt` relation. The target of such a relation must always be a location node. When the source node is a location node as well, the relation represents a part-of relation for a specified point in time (e.g., Dresden Main Station *Is Located At* Dresden). The `LocatedAt` relation is transitive. Lets consider for example a person is located in a car and this car is located in the city of Dresden. In this case, the person is located in Dresden as well. While the car changes its location when it is moving, the driver will not change its position relative to the car, but its location relative to the geographical location. The part-of relation on locations forms a graph that has no cycles.

4.4 Context Model Query

In order to query the context model, we have created a first prototype for a query language (GRoCoMo-QL) based on pattern matching in graphs. Listing 1.1 shows an example. Each query starts with a definition of roles. Each role has an `id` (e.g., node1) and represents a node with an optional type constraint (e.g., Location). Then, relations can be defined. Each relation has an `id`, an optional type constraint, a source and a target role, as well as a temporal constraint. The last part of the query is a restriction clause, where any restrictions on the structure of the previously defined roles and relations can be specified. The query from Listing 1.1 will return all tuples `(node1, node2, rel1)`, where `node1` is a location node, `node2` is a device node and the name attribute of `node2` has the value ``Example Robot''. Furthermore, both nodes must be connected by a `LocatedAt` relation from `node1` to `node2`. As a temporal constraint, within all results it is guaranteed, that all nodes and relations were/are valid at the same time and only nodes and relations are considered, that were valid within the last 5 minutes.

```
1 nodes {
2  node1 : Location [valid within last 5min];
3  node2 : Device [valid within last 5min];
4 }
5 relations{
6  rel1 : LocatedAt(node2,node1) [valid within last 5min];
7 }
8 where node2.name = "Example_Robot";
```

Listing 1.1. A GRoCoMo-QL example.

4.5 Context Model Manipulation

The context model can be changed using a sequence of the following basic graph rewrite operations:

Add Node/Relation: This operation adds a node/relation to the graph. The concrete type is specified on the client side.

Remove Node/Relation: This operation takes the `id` of a node/relation as an input and will remove the corresponding node. In contrast to the invalidation operation, a deletion will irreversibly remove the node.

Invalidate Element: The invalidation operation of a modeling element (i.e., nodes and relations) sets the end time of the corresponding element to the current time. Hence, it will be considered invalid.

Validate Element: Analogously to the invalidation, the validation operation will create a new valid timespan and sets the start time to either the current or the provided time.

Set Property: Some of the GroCoMo meta-classes (cf. Figure 2) define built-in properties (e.g., name). Those properties can be changed using this operation. The changes of built-in properties are not tracked (w.r.t. historical data).

From those basic operations, complex operations can be composed (e.g., replace node, set attribute, set location). The context model can be manipulated by (a) *sensors*, (b) *inference-*, and (c) *cleanup units*. Sensors observe the environment (physical or virtual) and update the context model accordingly. Inference units enrich the context model with new nodes and/or relations based on analysis of the available data in the context model. Cleanup units remove nodes and relations based on application- and hardware-specific rules in order to avoid memory overloads. To express the manipulation of those different manipulation units, we have created a prototypical manipulation language (*GRoCoMo-ML*), based on the Query Language sketched in Section 4.4. Listing 1.2 shows an example. A manipulation script consists of a set of labeled situations, where each situation contains exactly one query. Then, conditions on the results of the corresponding queries can be stated. The match/mismatch of situations can be combined using logical operators (e.g., `and`, `or`, etc.), as well as aggregation operations stated on the number of matches. In the provided example, the corresponding sequence of manipulation operations is executed, when the pattern, described in the situation `"PersonInRoom1"`, is matched more than 5 times.

The presented context model GRoCoMo provides a predefined set of node-types (e.g., Person, Activity, Location, Time), representing contextual entities (supporting modeling dimension *Z1*). The core model introduces specific nodes (i.e., `Attribute Node`) and specific relations (i.e., `HasAttribute` relation) to model the individuality of entities (dimension *Z2*). Through this approach dynamic complex types can be modeled by creating nested attributes. Relations represent relationships either between entities or between other relations (dimension *Z3*). Activities can be modeled using special `Activity` nodes (dimension *Z4*). In order to express temporal and historic data (dimension *Z5*), valid times by means of `Timespan`s can be assigned to each node and relation, to express when the validity of a modeling element started and ended. Beside the represented timestamps, it is also possible to assign symbolic representations of

```
Situation "PersonInRoom1"{
 nodes {
  room1 : Location;
  person : Person;
 }
 relations{
  rel1 : LocatedAt(person,room1);
 }
 where room1.name == "kitchen";
}
ON Count(PersonInRoom1) > 5 {
 an = new AttributeNode(name = "is␣crowded", value=true);
 rel = new AttributeRelation(source = room1, target = an);
}
```

Listing 1.2. A GRoCoMo-ML Example

time (e.g., Monday, Holiday etc.). Finally, spatial information is captured by `Location` nodes (dimension *Z6*).

5 Implementation

To investigate the feasibility of the presented approach, we have created a prototypical implementation using the role-based self-adaptive system *Smart Application Grids* (SMAGs) [10]. SMAGs is a component-based modeling and execution approach for runtime reconfiguration. SMAGs defines a predefined implementation for a MAPE-K loop, which can be adapted at runtime [11] as well. We implemented the GRoCoMo as a special `Context Model` component and integrated the Query Language and the Manipulation Language in the Sensor, Inference and Adaptation Component. For the context model representation, we used the JUNG[2] graph framework. For the pattern matching, we used the GUERY[3] framework. GUERY defines a textual syntax for *Motifs*, representing patterns, which are either provided by simple text files or can be created using an object-oriented API. On top of GUERY, we defined two Domain-Specific Languages (DSLs) for the GRoCoMo-QL and -ML using the Eclipse-based DSL-framework Xtext[4]. Instances of GRoCoMo-QL, as well as the query parts from GRoCoMo-ML, are transformed to valid GUERY-queries. Based on the result propositions in the manipulation language, the results of the queries are investigated and based on the evaluation of the situation guards, the provided reconfiguration scripts are executed accordingly.

[2] JUNG: http://jung.sourceforge.net/ (visited 20.05.2014)
[3] GUERY: https://code.google.com/p/gueryframework/ (visited 20.05.2014)
[4] Xtext: http://www.eclipse.org/Xtext/ (visited 20.05.2014)

6 Conclusion and Future Work

The domain of service-robots highly requires software systems that adapt their behavior based on past, present and potential future situations of the involved system. The MAPE-K loop represents the adaptation process from data acquisition, to system reconfiguration, based on data stored in a shared knowledge base. An important part of this knowledge base is the context model, capturing environmental data. It was observed, that structured knowledge representations (e.g., ontologies) are best suited for modeling open and unknown environments. Current approaches, however, fail to support context data analysis over time and location simultaneously. In this paper, we have proposed the context model **GRoCoMo** (Graph- and Role-Based Context Model), using a typed, attributed and directed graph as a foundation. The model supports different predefined entities and relations, which can be extended for specific domains. The model treats activities, time and location as first-class-citizens. For temporal information, validity-timespans are attached to each modeling element, representing the timespan when an element is/was valid (w.r.t. a specific location). To model locations, the model provides specialized location nodes and relations, as well as a transitive semantics for those relations. We have outlined a first version of a pattern-based *Query Language*, as well as a *Manipulation Language* using pattern-based situation detection and a set of predefined manipulation operations, to change the context model. For future work, the provided prototypical implementations of the API and the corresponding languages have to be finished, stabalized and published. In addition, the presented approach has to be evaluated in real-world examples. Bettini et al. [2] described additional properties of context models that mainly focus on data quality. Those properties were already considered, but were not described in this paper. Those properties have to be investigated, covered and evaluated as well. Finally, it has to be investigated how pattern recognition techniques can be used to automatically detect situations in terms of context graph patterns, enabling machine-learning adaptation strategies, as well as situation specification guidance.

Acknowledgment

This work is supported by the German Research Foundation (DFG) within the Cluster of Excellence “Center for Advancing Electronics Dresden” and the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. 2(4), 263–277 (Jun 2007)
2. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. 6(2), 161–180 (Apr 2010)

3. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. 36(4), 19–26 (Dec 2007)
4. Crowley, J.L.: Navigation for an intelligent mobile robot. Robotics and Automation, IEEE Journal of 1(1), 31–41 (1985)
5. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference. pp. 270–275 (2004)
6. yi Hong, J., ho Suh, E., Kim, S.J.: Context-aware systems: A literature review and classification. Expert Systems with Applications 36(4), 8509 – 8522 (2009)
7. IBM Corp.: An architectural blueprint for autonomic computing. IBM Corp., USA (Oct 2004)
8. Kawamura, K., Pack, R., Iskarous, M.: Design philosophy for service robots. In: Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on. vol. 4, pp. 3736–3741 vol.4 (Oct 1995)
9. Paganelli, F., Giuli, D.: An ontology-based system for context-aware and configurable services to support home-based continuous care. Trans. Info. Tech. Biomed. 15(2), 324–333 (Mar 2011)
10. Piechnick, C., Richly, S., Götz, S., Wilke, C., Aßmann, U.: Using role-based composition to support unanticipated, dynamic adaptation-smart application grids. In: ADAPTIVE 2012, The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications. pp. 93–102. Nice, France (2012)
11. Piechnick, C., Richly, S., Kühn, T., Götz, S., Püschel, G., Amann, U.: Contextpoint: An architecture for extrinsic meta-adaptation in smart environments. In: ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications. Venice, Italy (2014)
12. Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, J., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.: A comprehensive context modeling framework for pervasive computing systems. In: Proceedings of the 8th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems. pp. 281–295. DAIS'08, Springer-Verlag, Berlin, Heidelberg (2008)
13. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp - Sixth International Conference on Ubiquitous Computing, Nottingham/England (2004)
14. Strang, T., Linnhoff-Popien, C., Frank, K.: Cool: A context ontology language to enable contextual interoperability. In: Distributed Applications and Interoperable Systems, Lecture Notes in Computer Science, vol. 2893, pp. 236–247. Springer Berlin Heidelberg (2003)
15. Wang, X., Zhang, D.Q., Gu, T., Pung, H.: Ontology based context modeling and reasoning using owl. In: Pervasive Computing and Communications Workshops. pp. 18–22 (March 2004)
16. Xu, N., Zhang, W.S., Yang, H.D., Zhang, X.G., Xing, X.: Cacont: A ontology-based model for context modeling and reasoning. Applied Mechanics and Materials 347, 2304–2310 (2013)
17. Zimmermann, A., Lorenz, A., Oppermann, R.: An operational definition of context. In: Modeling and Using Context, Lecture Notes in Computer Science, vol. 4635, pp. 558–571. Springer Berlin Heidelberg (2007)