A Comprehensive Context Modeling Framework for Pervasive Computing Systems

Roland Reichle[1], Michael Wagner[1], Mohammad Ullah Khan[1], Kurt Geihs[1], Jorge Lorenzo[2], Massimo Valla[3], Cristina Fra[3], Nearchos Paspallis[4], and George A. Papadopoulos[4]

[1] University of Kassel, Distributed Systems Group
{wagner,reichle,khan,geihs}@vs.uni-kassel.de
[2] Telefónica Investigación y Desarrollo
jorgelg@tid.es
[3] Telecom Italia Lab
{massimo.valla,christina.fra}@telecomitalia.it
[4] Department of Computer Science, University of Cyprus
{nearchos,george}@cs.ucy.ac.cy

Abstract. Context management in pervasive computing environments must reflect the specific characteristics of these environments, e.g. distribution, mobility, resource-constrained devices, or heterogeneity of context sources. Although a number of context models have been presented in the literature, none of them supports all of these requirements to a sufficient extent at the same time. In this paper, we present a comprehensive and integrated approach for context modeling in pervasive computing environments. It combines the advantages of existing approaches and addresses the need for supporting effective software development. The proposed context model follows an ontology-based approach and has three layers of abstraction, i.e. conceptual layer, exchange layer, and functional layer. This layered approach facilitates a model-driven development of context-aware applications. Throughout the paper we compare our solution with the related work in order to clearly demonstrate why we needed to develop a new context management framework and where we have adopted existing ideas.

Keywords: Context Awareness, Context Modeling, Ontology, Model-Driven Development, Pervasive Computing.

1 Introduction

In recent years, context awareness has attracted a lot of attention, especially in the realms of mobile and pervasive computing. Context-aware applications are capable of monitoring and exploiting information about external operating conditions. Typically, such systems are also self-adaptive, in the sense that they can dynamically adapt as a response to changes in the execution context. Automating the development of such systems is an important challenge.

R. Meier and S. Terzis (Eds.): DAIS 2008, LNCS 5053, pp. 281–295, 2008.

A context model provides an unambiguous definition of the context artifacts, their representations, semantics and usage. It takes into account the general characteristics of context information, such as its temporal nature, ambiguity, impreciseness, incompleteness and privacy. Furthermore, a context model must also address special requirements of pervasive computing environments like distribution, mobility, heterogeneity of context sources and resource-constrained devices. Often, pervasive applications require high-level context information that is derived from low-level context values. Therefore, support for automatic context reasoning has to be provided as well.

As it will be shown throughout this paper, existing context modeling approaches address a sub-set of these challenges only, or cover some of them only to a limited extent. Moreover, most of them view context modeling either from a pure conceptual or a pure functional perspective. However, when engineering context-aware systems, a software developer needs to deal with many aspects at the same time, e.g. define the semantics and relations between context elements at a conceptual view, realize the information exchange between heterogeneous nodes, and provide the concrete implementation of the context management functionality at a specific node.

The main contribution of this paper is a new comprehensive and integrated context modeling approach that is based on a new context ontology and three layers of abstraction: conceptual, exchange and functional. These three layers cover the identified requirements of context management in pervasive computing environments and, at the same time, facilitate the analysis and design of context-aware applications as part of a comprehensive, model-driven software engineering process. The presented context model is a result of a research EC IST project called *Self-Adapting Applications for Mobile Users in Ubiquitous Computing Environments (MUSIC)* [3]. The goal of MUSIC is to develop a comprehensive open-source computing infrastructure and an associated software development methodology that facilitate the development of self-adapting, context-aware applications in ubiquitous and pervasive computing environments.

The rest of this paper is organized as follows: Section 2 studies requirements for context modeling in pervasive computing environments, while Section 3 discusses existing approaches. The MUSIC context model is described in Section 4, and discussed in Section 5. Finally, Section 6 presents our conclusions and points to future work.

2 Requirements

This section identifies requirements for a context model that aims to ease the development of context-aware applications in mobile and pervasive computing environments. A comprehensive list of requirements has been derived through a process where a set of case studies featuring both real (commercial) and fictional scenarios were studied and evaluated in the scope of mobile and pervasive computing [3]. The requirements identified from the case studies are:

- *Ease of development:* While at a conceptual level modeling the semantics and the relations between context information is very important, the runtime representation of the context data must aim for efficiency. Appropriate development support must be provided to the software developers to ease their tasks considering the whole development process and incorporating all views and aspects. In this respect, Model-Driven Development (MDD) is favored.
- *Considering the characteristics of mobile and pervasive computing environments:* Mobile and pervasive computing environments imply further complexity as they are characterized by distribution, heterogeneity, unpredictability, unreliable communication links, etc. Furthermore, the limited capabilities of mobile devices, e.g. in respect to processing power, memory and energy consumption, have to be taken into account.
- *Need for machine-interpretable representation of context information:* A typical approach to tackle heterogeneity and to provide a machine-interpretable representation of context information is the use of semantic annotations. They are attached to the actual context data to enable automatic exploitation and transformation of information in distributed context sharing scenarios. Furthermore, they can be utilized to enable automatic context reasoning.
- *Dealing with special context properties:* Unlike data in conventional database systems, context data is characterized by properties such as incompleteness, ambiguity, uncertainty, and temporal nature.
- *Dealing with context information partitioning:* In adaptive systems, sharing of context information is a natural requirement. However, because of the nature of mobile and ubiquitous computing, it is possible that the nodes carrying the context information are partitioned. The context models should cope with such circumstances and enable the merging of the data when needed.
- *Evolution and extensibility:* Context models should not be monolithic, but rather be flexible and extensible. New applications and possibly new context nodes shall be allowed to enter the system. As the applications and their context needs evolve, so should the context model.

This list describes the requirements that originate from our chosen scenarios. As they are not intended to focus on a small number of rather specific applications, but more on pervasive applications in general, they are naturally quite high-level. Although we believe that the list is rather comprehensive for pervasive applications, we do not claim completeness. Other applications may have different or additional requirements. In addition, more general requirements apply to context modeling, just like they apply to software systems in general. These include *platform independence*, *privacy and security issues*, *support for automatic test execution*, *logging*, *simulated operation* and *visualization* of the system state.

3 Discussion of Existing Context Models

In search for an existing context model that would satisfy all of the above requirements, we carefully examined the related work in this research field. Our

investigations revealed that research in the area of context modeling is well established and many ideas have been developed for addressing the above requirements individually.

3.1 Existing Approaches on Context Modeling

In order to provide application dependent context information through a context framework, a uniform way of representing and sharing context is required. Strang and Linnhoff-Popien [16] evaluate the most relevant context approaches based on the data structures used for representing and exchanging context information: key-value pair, markup scheme, graphical, object oriented, logic based and ontology based models. According to their evaluation, the most promising assets for context modeling for ubiquitous computing environments are found in ontology based models. In these models, the semantic context information is represented using one of the ontology markup languages, for example OWL (Web Ontology Language) [10]. We share their opinion and consider ontologies as an appropriate way to deal with the heterogeneity implied by ubiquitous computing environments. An ontology defines a common vocabulary to share context information among devices, services and users. This makes it possible to reason about various context types, thanks to machine-interpretable definitions on basic concepts in the domain and relations among them.

There are several projects that also apply ontologies as a central concept for modeling context information. For instance, Chen *et al.* [1] defined a context ontology based on OWL to support ubiquitous agents in their Context Broker Architecture (CoBrA). Their approach targets home area intelligent environments and applies sensor information detection and context awareness as a way of dealing with users' activities, intentions and movements between different home areas.

Ranganathan *et al.* [13] developed a middleware for context awareness and semantic interoperability in which they represented the context ontology in DAML+OIL [8]. One of the main shortcomings of this approach is that it does not deal with the specialized context characteristics, such as incompleteness, and that its extensibility is limited.

Context-Driven Adaptation of Mobile Services (CoDAMoS) [12] defines a generic ontology to model context in Ambient Intelligence infrastructures that suits very well the requirements of mobile computing. This ontology is based on four general entities. (1) The user is the central entity, including the user's profile, preferences, mood and current activity. The rest of the entities should adapt to the user, not vice versa. (2) The environment in which the user interacts, including information such as temperature and lighting. (3) The platform that describes the hardware and software of a device, including device resources such as memory and bandwidth. (4) The service that provides specific functionality to the user.

The Service-Oriented Context-Aware Middleware (SOCAM) [4] [17], is an architecture for building context-aware services based on a two-level context model. This middleware acquires context information from different sources and

interprets it. The context ontology is divided into a two-level hierarchy, distinguishing between common and specific context information. The upper level describes global concepts of the ontology and captures general knowledge about location, type of entity, person or activity. On the other hand, the lower level is divided into several pervasive computing sub-domains, each one of which defines specific details and properties for each scenario. Depending on the situation and the available devices, an appropriate sub-domain is selected from the lower level. When environment changes are detected, the lower level ontology can be dynamically plugged into and unplugged from the upper ontology, thus dynamically changing this association. This mechanism appears to be very reasonable also with respect to resource limited devices. An ontology resulting from the extension of the top-level ontology with a domain-specific ontology can be kept quite small in comparison with a single huge ontology capturing all potentially involved concepts.

Strang *et al.* [15] describe a context modeling approach using ontologies as a formal foundation. They introduce their Aspect-Scale-Context (ASC) model and show how it is related to other models. A Context Ontology Language (CoOL) is derived from the model, which is used to enable context-awareness and contextual interoperability during service discovery and execution in a distributed architecture. One highlight of the ASC model is that it explicitly addresses heterogeneity with regard to different representations (called scales) of context information.

Apart from the ontology-based approaches, there are several other projects on context modeling that fulfill several of the requirements that were stated in the previous section. One example is CML from Henricksen and Indulska [6] [7]. They also incorporate ontologies to address particular aspects like privacy. The formal foundation of their context modeling approach is an enhancement of the ORM language. With the situation abstraction they provide elaborate support for reasoning on context information. Their context model can also deal with special characteristics of context information, such as temporal nature, incompleteness, ambiguity, etc. Additionally, the context modeling approach is complemented by a model-driven development approach (which provides an API for the application developer) and a methodology for the development of context-aware application [7]. In this approach, context information is addressed from three levels, i.e. conceptual, management and implementation level.

The Comprehensive Structured Context Profiles (CSCP) [5] was developed based on RDF to represent context by means of session profiles. However, this approach does not deal with all our required context characteristics, like the temporal nature of context.

In [11], Hoenle *et al.* highlight the benefits of integrating meta-data into the context model. They argue, that meta-data facilitate important aspects like the assessment of the quality of context information, sensor fusion and data cleansing and provide more flexibility when dealing with context information. In their approach meta-data are associated to context information at object level as well as at attribute level.

3.2 Why Another Context Model?

If we look at our requirements and at the approaches described above, the first impression is that it should not be a difficult task to find an existing context model that is suitable for our purposes. However, none of the examined approaches supports all of our requirements to a sufficient extent. Ease of development using MDA, as one of our key requirements, is only addressed sufficiently by Henricksen and Indulska, but their model is not based on ontologies as the primary modeling concept. Similar to many related works, we consider the concepts of ontologies necessary to establish a common vocabulary in a heterogeneous pervasive computing environment. Such an environment also implies heterogeneous representations of context information. This is also not explicitly addressed by CML but by the ASC model from Strang *et al.* However, their approach does not provide such an elaborate development support based on MDA as the CML project.

As we could not find an approach, that fulfilled all of our requirements, our next step was to figure out, if one approach can easily be extended to cover all the aspects. Having in mind that CML already utilizes ontologies for issues like privacy we investigated the feasibility of incorporating ontologies as primary modeling concept in CML. But we quickly came to the conclusion, that this would require too much effort, as it would mean to completely replace the ORM and its extensions or to establish a mapping from an ontology based approach to the ORM. Furthermore, even if we had established such a mapping, the problem of heterogeneous representations would still remain unsolved.

The idea to complement the ASC model with MDD support appeared to be quite promising, in particular when considering, that CoOl was also designed to facilitate the mapping to other context models. Problems with the ASC model were found in small details. In our view context information should characterize an entity of the world (e.g. laptop, device, user, *etc*) with a certain type or scope of information (e. g. location, current situation, battery status, *etc*) in a certain representation (e.g. GPS coordinates in the case of location). In our terminology a context scope is a kind of context information type. Therefore, the three concepts entity, type and representation should be clearly separated. In the ASC model, the type of information (called aspect) is only referred indirectly through the scales, which correspond to a certain representation in our terminology. We faced problems with this indirection when building taxonomies of context information types and corresponding taxonomies of representations. Clearly separated concepts not only facilitate building taxonomies, they also ease the automatic model-based generation of context interpreters that are responsible for one context information type and can deal with several representations.

Based on this analysis, we saw the need to design a new context modeling approach that utilizes the advantages and most promising features of the existing works in order to develop a comprehensive integrated approach. As can also be seen from the considerations above, combining the different concepts is not at all a trivial task.

4 The MUSIC Context Model

This section describes the context model of the MUSIC project. In the first subsection we describe the general structure of the new context model. Then we introduce the different layers and show in the last subsection how to use our approach in the Model Driven Development.

4.1 Three Layers of Abstraction

We identify three basic layers of abstraction that correspond to the three main phases of context management: the conceptual layer, the exchange layer and the functional layer. The conceptual layer aims to be leveraged by the developers and to be exploited in the model-driven development approach. This layer enables the definition of context artifacts such as elements, scopes, entities and representations based on standard specification languages like UML and OWL [10]. The exchange layer aims to be utilized for interoperability between devices. At this layer, the context information can be expressed in any adequate representation, such as XML, JSON (JavaScript Object Notation) [9] or simply CSV (Comma Separated Values). Finally, the functional layer refers to the actual implementation of the context model representation and the internal mechanisms used in the different nodes. This model can be object-based, but it does not necessarily need to be interoperable as it is platform-specific and as different devices might use different implementations of it, using for example Java and .NET. The main objective of this layer is efficiency, both in terms of processing speed and resource consumption. This paper focuses on the conceptual and the exchange layers of the proposed hierarchy. Figure 1 illustrates how these concepts fit into these three layers.

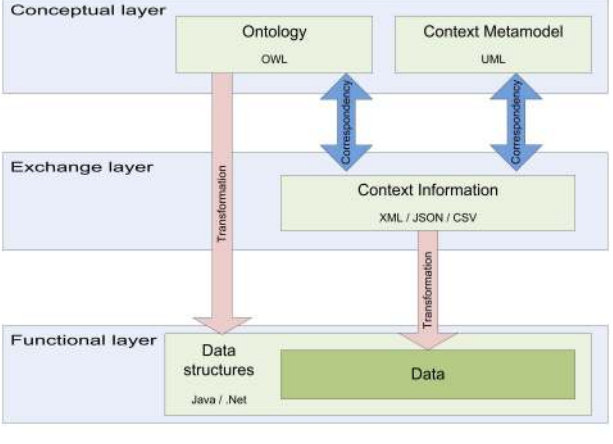

Fig. 1. The three layers of the MUSIC context model

For our context model, we decided to incorporate the concept of ontologies in the conceptual layer of the context model for several reasons. (1) Ontologies facilitate the establishment of a common understanding of the semantics of context elements and their associated metadata and therefore boost interoperability. (2) Similar to the ASC model proposed by Strang *et al.* [15], ontologies can also be used to define the internal structure of context data, thus allowing several representations, their interpretation and automatic conversions between them. (3) By incorporating ontologies, it is possible to model a wide range of relationships between context elements, which is essential for a flexible context reasoning approach. Also, a context meta-model is defined to facilitate automatic transformation between the different layers of the context modeling approach and to define basic guidelines for modeling the ontology.

The ontology is described in OWL and the context meta-model is specified in UML. Together they form the conceptual layer of the MUSIC context modeling approach. The context meta-model defines the general structure of context information and shows how concepts and/or individuals/entities specified in the ontology are referenced. In turn, as the context meta-model defines a general representation of context information it can also be considered as a kind of schema for defining the concrete representations of context elements in the ontology.

At the exchange layer, an instance of the conceptual model is represented in XML (or alternatively in JSON or CSV). The representation in XML is quite straightforward, as it is the common way to represent individuals of the ontology (which can be seen as context information).

The functional layer also defines a set of data structures for storing the context information. As the internal structure of context elements is specified in the ontology, it is possible to automatically generate the corresponding data structures for specific platforms along with appropriate serialization and de-serialization methods. Thus, the data structures can easily be filled with the information represented at the exchange layer without much overhead spent for interpretation. It is also worth noting here, that the information concerning the ontology is only transferred once or on demand. All these features take into account the quite limited resources of mobile devices in pervasive computing environments.

4.2 The Conceptual Layer of the MUSIC Context Model

As depicted in Figure 1 the MUSIC context model is composed of an ontology and a metamodel at the conceptual layer, which is described in the following.

The MUSIC Context Meta-Model. Figure 2 illustrates the proposed Context Meta-model. Context information is abstracted by *context elements* which provide information about *context entities* and *context scopes* and that can be composed of other context elements and can contain a number of *context values*. For example, a context element's *network connections* in a *device's* context can contain the elements *Wi-Fi* and *Bluetooth*, and both elements can have the values *Cost* and *Bandwidth*. *Context elements* are associated to *context scopes* that group context values belonging to the same context domain. For example, the

scope *Position* groups context values like: *Longitude*, *Latitude* and *Accuracy*. The *context entities* refer to concrete entities in the world, for example *User*, *Device*, *etc.*

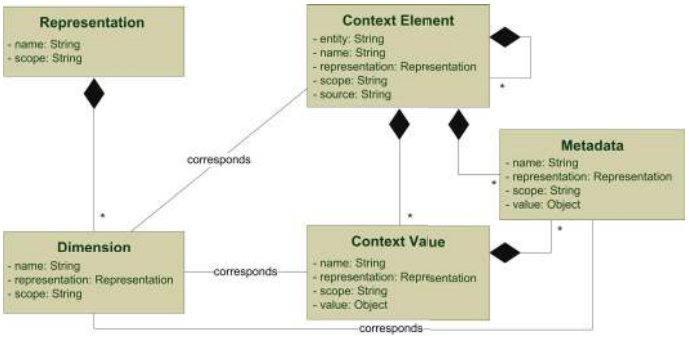

Fig. 2. The MUSIC Context Meta-Model

Metadata can be associated with context elements and context values. Here we distinguish between *predefined* (or suggested) *metadata* and *user-specific metadata*. The proposed model includes the *predefined metadata*: *name*, *entity*, *scope*, *representation*, *source* and *sourceType*. The *name* serves as an identifier. *Scope*, *entity* and *representation* refer to the MUSIC context ontology; *scope* refers to the semantic concept that groups context values belonging to the same context domain and characterizes the context information, e.g. *deviceStatus*; *entity* refers to the concrete individual to which the context information is associated, e.g. "My Windows XP Laptop". The *representation* refers to the internal representation of the context information which is also specified in the ontology. With these types of metadata, it is specified that a context element characterizes the semantic concept *scope* for the individual *entity* and its internal structure corresponds to *representation*. The *source* is a unique identifier of the component that provides the context information (e.g. a context sensor or reasoner). For *context values* the suggested types of meta-data are *name*, *scope* and *representation* that have the same meaning as the corresponding metadata types of the context element. In addition, it is allowed to associate *user-specific metadata* to context elements and context values. In a way, these metadata can be seen as additional *context values* and they are also represented in the same way. However, in contrast to *context values*, *metadata* can be associated to *context elements* and *context values*. Each *context element*, *context value* and *metadata* has a *representation*. According to aspects in the ASC model described in Strang *et al.* [15], each *representation* (in the ASC model called aspect) aggregates one or more *dimensions* (scales in ASC). Each *dimension* corresponds to a certain *context element*, *context value* or *metadata element*. A *dimension* itself has a *representation*, which again can consist of several *dimensions*. With these concepts, the internal structure of the context information is defined through the context element.

The MUSIC Context Ontology. This section introduces the MUSIC context ontology through an example. This example does not claim completeness but rather aims at showing the general modeling concepts and illustrating how the conceptual layer, which contains the context meta-model, is complemented by an ontology. In order to provide an extensible ontology that is well-structured and easy to understand, we introduce a two-level hierarchy for the ontology, similar to SOCAM. Here, we introduce the structure of the top-level ontology.

The context meta-model refers to the ontology with regard to three aspects: the *context scope* that is characterized by the context element, the *type* of the particular individual/entity of the characterized *scope* and the *representation* of the context information. These different aspects have to be covered while modeling the ontology.

Figure 3 presents the classes corresponding to the semantic concepts we would like to characterize through context information/context elements in our context management system. This figure only includes a small number of classes, such as for example the concept *DateTime* which is a subclass of *BasicConcepts*. As depicted in the figure, the most important relation is that each *Concept* has a *Representation*. The class *Concept* is not only used to classify *EntityTypes*, *ContextScopes* and *BasicConcepts*. Additionally, some further relations between these classes and its subclasses can be defined (e.g. *isLocatedIn*). These relations can be used for ontology reasoning.

As a second part of the ontology, the *representations* for the *concepts* must also be specified. As depicted in Figure 3 a *concept* can have one ore more *representations*. By allowing representing certain context information in several ways, we do not only face the challenge of heterogeneous context sensors for a certain semantic concept, but we also ease the merging of ontologies, at least to a certain extent. If an ontology matches a second one with regards to the classes for the concepts and their relation, and only differ in the representation of context information, the second ontology can be integrated in the first one in a straight forward manner.

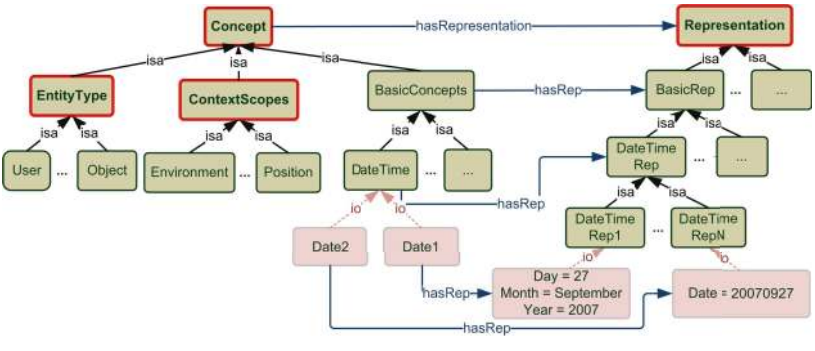

Fig. 3. The main structure of the MUSIC Context Ontology

As we envisage explicit support for heterogeneous representations of context scopes, we also allow the definition of Inter-Representation-Operations (IRO) as in the ASC model [15]. This concept is a further step in supporting context providers and consumers in a heterogeneous environment. It allows to ask for context data by a context consumer by describing a certain *scope*, characterizing a certain *entity* of this *concept* and having a certain *representation*. If this does not match the representation provided by the context sensor, an appropriate one can be computed with the corresponding IRO.

4.3 Model-Driven Development

As already discussed in the previous section, we do not want to provide only a new context model, but rather an integrated approach for context modeling, reasoning and querying together with support for application development. Thus, we use our context model also as a key ingredient in the model-driven application development. In general, context-aware software is developed using traditional programming methods and models, and the use of context information is implemented directly into the source code. Even if the logic used to access and process context information and to react to context changes is isolated within special components, the applications are still difficult to maintain, as source code must be modified to support additional classes of behavior and context. To facilitate the application development process, we use the context ontology also at design-time to support the MDD of context-aware applications. The MDD methodology exploits mainly the conceptual layer, where the context artifacts (elements, sensors, *etc*) are defined based on standard specification languages, like UML and OWL. From the high-level specifications provided at this layer, appropriate data representations and data structures for the other layers can be automatically generated. It is even possible to automatically provide serialization and de-serialization methods to be leveraged at the exchange layer and to incorporate IROs for converting between different representations. Additionally, we provide a software development methodology for adaptive context-aware applications in ubiquitous computing environments. Further information about this methodology can be found in [3]. As depicted in the example in Figure 3, the representation of a concept embodies also the main structure of the context information. This structure can be used to automatically generate the corresponding data-structure. For *data1* in the example in Figure 3, the data-structure in Java would be generated as following:

```
Class DateTimeRep1 implements Serializable{
   private int day = null;
   private string month = null;
   private int year = null;
...   }
Date1 = new DateTimeRep1(27, "September", 2007);
```

Furthermore, both the constructors and the getter/setter methods can be automatically generated.

```
DateTimeRep1(){...}
...
DateTimeRep(int d, String m, int y){
   this.day = d;
   this.month = m;
   this.year = y;}
```

As aforementioned we use the concept of IRO to transfer context information from one representation to another (in our example from DateTimeRep1 to DateTimeRep2). The skeletons of the IROs can also be generated automatically:

```
static DateTimeRep1 IRO_DTRep2_To_DTp1(DateTimeRep2 date2){
   DateTimeRep1 date1 = new DateTomeRep1();
   //TODO for Developer: Fill out the missing calculations and
   check the variables defined and assigned above
   return date1;}
```

Additionally, it is even possible to automatically provide serialization and de-serialization methods to be leveraged at the exchange layer. This means that we can automatically generate the necessary methods to send or receive the data via the exchange layer in the different formats (i.e. XML). Here we use also the IRO. A context sensor which provides the context information in a certain representation, uses its serialization method to submit this data via the exchange layer. Then the context consumer uses the de-serialization method to insert this data into his data structure. In this method, we check if the information corresponds to the requested representation, if not then automatically a corresponding IRO is called. Here we have to highlight, that the application developer does not need to worry about this process of serialization/de-serialization and conversion. The application developer just uses the generated getter-/setter-methods to access to data in the data structure. As part of our comprehensive approach for context modeling, reasoning and querying, we provide also an appropriate Context Query Language (CQL), which is described in Reichle *et al.* [14] in more details. This CQL will also be used for the MDD as we can automatically generate the code corresponding to a static query.

5 Discussion

In this paper, apart from other important requirements we emphasize the need of using ontologies to establish a common vocabulary of concepts and to explicitly address heterogeneous representations of context information in pervasive computing environments. At the same time, we highlight the need for software development support that allows developers to easily construct context-aware and self-adaptive systems. A representative set of related context modeling approaches is described in Section 3. We have argued that none of these approaches fulfills all of our requirements to a sufficient extent at the same time. Therefore, our proposal extends the state of the art, by combining the most promising features of existing approaches to a context model that is comprehensive and

fulfills important requirements arising in pervasive computing environments. As it is already shown in Section 3, the task of integrating the different ideas was challenging, as some problems were obvious, while others were visible only when focusing on specific details.

We have introduced a two-level hierarchy for the ontology. Similar to SOCAM, we distinguish between a top-level ontology capturing global knowledge and general concepts, and the domain-specific extensions. By allowing to integrate domain- or application-specific extensions our context modeling approach is not monolithic but evolvable for new applications entering the system. In order to cope with heterogeneous representations we define the internal structure of context information along with Inter-Representation-Operations in the ontology, similar to the ASC model [15]. In addition to establishing a common vocabulary for context information through an ontology, the concept of Inter-Representation-Operations further boosts interoperability. Therefore, our approach explicitly addresses the requirements arising from a heterogeneous computing environment. Metadata can be associated to context elements and also to context values similar to what is proposed by Hoenle *et al.* [11], which comes as an appropriate mean to deal with the special properties of context information and also facilitates merging of context information when nodes have been partitioned. Last but not least, we incorporate some ideas from Henricksen and Indulska [6] [7] in order to ease the development task by employing an MDD approach. In summary, our new context model provides:

- Support of all three context management layers (conceptual, exchange and functional layer). The exchange layer and the corresponding links to the conceptual and functional layers are introduced to face the challenges that arise from a distributed context sharing scenario in heterogeneous computing environments.
- Explicitly addressing MDD by using the ontology, not only to introduce a general vocabulary and relationships between context elements, but also to define different representations which comprise information about the used data structures.
- An ontology that is divided into two corresponding hierarchies: concepts and representations. The hierarchy of concepts contains the general vocabulary and the relations between the elements, whereas the hierarchy of representations is used to define the internal structure of context elements. With this division, it is possible to use only the light-weight concepts hierarchy for context reasoning while omitting large parts of the ontology that only contain the representations.

Our new context model is based on concepts that have already been proved viable. However, only a simplified version of the context model has been prototyped so far. It is currently used by the pilot service developers in the MUSIC project [3]. They will provide feedback from the implementation phase of the pilot applications. This feedback will then be leveraged to improve and fine-tune our approach. Although we have not yet implemented the complete context management system, we are very confident that it can be done. The first experiences

with the new approach are quite promising and it seems to be applicable and sufficient for all our case studies.

However, we are aware that some issues deserve further attention. One issue for example might be the resource limitations of mobile devices that are currently available. Although we keep the ontology as small as possible, utilizing the two-level approach, ontology reasoning at run-time remains a resource consuming task, but is unavoidable to some extent. Furthermore, the classes and data-structures that are generated at design-time can be loaded at run-time, and furthermore, they provide serialization and de-serialization methods. Additionally, they allow interpretation of context information and the conversion between different representations. Thus, we provide a convenient and efficient method for dealing with heterogeneous context information, although these advantages incur additional memory requirements, which could be a serious problem on devices with limited resources. Furthermore, some problems could also arise from the plugging mechanisms used for the ontology. In many cases, extending ontologies through other ontologies also implies ontology merging to some extent, which is a really challenging task.

6 Conclusions and Future Work

In this paper, we have introduced a comprehensive context modeling framework for pervasive computing. We have adopted a three-layer architecture, featuring a conceptual, an exchange and a functional layer. In the conceptual layer, an ontology-based model is used, mainly at design-time, to enable model-driven development of context aware applications. The same context model is also used at run-time for the representation and the exchange of context information in the functional and exchange layers. We have also shown how we extend the state of the art by overcoming some of the limitations of existing approaches and by working towards a comprehensive solution which meets a set of preset requirements. In our on-going and future work, we endeavor to strengthen these results, first by evaluating the potential drawbacks as discussed in Section 5. Furthermore, we will extend our prototype implementation to completely support our approach. The prototype implementation will be used by the pilot application developers in the MUSIC project. Their feedback will then be leveraged to further improve and fine-tune our approach.

Acknowledgments. The authors of this paper would like to thank their partners in the MUSIC-IST project and acknowledge the partial financial support given to this research by the European Union (6th Framework Programme, contract number 35166).

References

1. Chen, H., Finin, T.: An Ontology for a Context Aware Pervasive Computing Environment. In: IJCAI workshop on ontologies and distributed systems, Acapulco MX (August 2003)

2. European EC-FP6 project MADAM (Mobility and ADaptation enAbling Middleware), http://www.intermedia.uio.no/confluence/display/madam
3. European IST-FP6 project MUSIC (Self-adapting applications for Mobile User. In: ubiquitous Computing environments), http://ist-music.eu
4. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An Ontology-based Context Model in Intelligent Environments. In: Proceedings of communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, USA, pp. 270–275 (2004)
5. Held, A., Buchholz, S., Schill, A.: Modeling of Context Information for Pervasive Computing Applications. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando (July 2002)
6. Henricksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: Second IEEE International Conference on Pervasive Computing and Communications, pp. 77–86. IEEE Computer Society, Los Alamitos (2004)
7. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Journal of Pervasive and Mobile Computing 2(1), 37–64 (2006)
8. Horrocks, I.: DAML+OIL: a Reason-able Web Ontology Language. In: Chaudhri, A.B., Unland, R., Djeraba, C., Lindner, W. (eds.) EDBT 2002. LNCS, vol. 2490, Springer, Heidelberg (2002)
9. JSON (JavaScript Object Notation), http://www.json.org/
10. OWL Web Ontology Language, http://www.w3.org/TR/owl-features/
11. Hoenle, N., Kaeppeler, U., Nicklas, D., Schwarz, T.: Benefits Of Integrating Meta Data Into A Context Model. In: Proceedings of 2nd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea), Hawaii, March 12 (2005)
12. Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an extensible context ontology for ambient intelligence. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 148–159. Springer, Heidelberg (2004)
13. Ranganathan, A., Campbell, R.H.: A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, pp. 143–161. Springer, Heidelberg (2003)
14. Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.: A Context Query Language for Pervasive Computing Environments. In: Proceedings of 5th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2008) in conjunction with the 6th IEEE International Conference on Pervasive Computing and Communication (PerCom), pp. 434–440 (2008)
15. Strang, T., Linnhoff-Popien, C., Frank, K.: CoOL - A Context Ontology Language to enable Contextual Interoperability. In: Stefani, J.-B., Demeure, I., Hagimont, D. (eds.) DAIS 2003. LNCS, vol. 2893, pp. 236–247. Springer, Heidelberg (2003)
16. Strang, T., Linnhoff-Popien, C.: A Context Modeling Survey. In: 1st International Workshop on Advanced Context Modeling, Reasoning And Management during UbiComp 2004 (2004)
17. Wang, X.H., Gu, T., Zhang, D.Q., Pung, H.K.: Ontology Based Context Modeling and Reasoning using OWL. In: Proceedings of Workshop on Context Modeling and Reasoning (CoMoRea 2004), Orlando, Florida, USA (March 2004)