A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development

Shumao Ou[1,2], Nektarios Georgalas[1], Manooch Azmoodeh[1], Kun Yang[2], and Xiantang Sun[3]

[1] British Telecom Group, Ipswich, UK
{shumao.ou, nektarios.georgalas, manooch.azmoodeh}@bt.com
[2] University of Essex, Colchester, UK
{smou, kunyang}@essex.ac.uk
[3] University of Aberdeen, Aberdeen, UK
xsun@csd.abdn.ac.uk

Abstract. Context-awareness is a very important feature for pervasive services to enhance their flexibility and adaptability to changing conditions and dynamic environments. Using ontologies to model context information and to reason about context at a semantic level has attracted a lot of interest in the research community. However, most of the proposed solutions are ad hoc or proprietary. Therefore, employing standard approaches to formulate the development process becomes of importance. In this paper we examine how OMG's Model Driven Architecture (MDA) can be applied to tackle the issues of context modelling and Context-Aware Application (CAA) modelling and development. A Context Ontology Model (COM) is presented to model context information at two levels: upper-level and extended specific level. A Model Driven Integration Architecture (MDIA) is then proposed to integrate rigorous model specifications and generate CAA implementations either semi-automatically or automatically.

1 Introduction

In order to flexibly adapt to changing conditions and dynamic environments, pervasive services need to become more context-aware. A pervasive service can be a simple service such as helping a user on a mobile device (such as a PDA or a smartphone) to find their favourite restaurant in the immediate vicinity around their current location. The challenges of context semantic representation, inference and interoperation in pervasive computing environments are well recognised. Earlier research work focused on context information gathering and integration aiming to achieve reusability for higher level pervasive applications [1, 2]. Other work studied the modelling of information from types of context in a platform independent way in order to support context management and interoperation [3, 9]. More recently, the notion of *ontology*, which is often used by Artificial Intelligence practitioners for knowledge representation, has emerged as a new approach to context modelling. Ontologies can model context at a semantic level establishing a common understanding of terms and meaning and enabling context *sharing*, *reasoning* and *reuse* in pervasive environments [9, 10, 11, 15].

Languages such as W3C's OWL [13] or RDF Schema can be used to specify ontologies in a machine-interpretable way. Without loss of generality, we consider both of them in this paper. An ontology includes definitions of commonly understood vocabularies and of logic statements that specify what each term in the vocabularies mean and how they relate to each other. An ontology removes ambiguity and is semantically independent to context. Ontology is, therefore, useful in bridging terminology differences thus enhancing interoperability. The concepts and logic expressed by ontologies are commonly accepted and can be communicated between human users and computer programs from different vendors. These features make ontologies the right mechanism for modelling context information in support of Context-Aware Application (CAA) development for pervasive computing environments, as they tackle heterogeneity introduced by diverse device technologies, the multiplicity of vendors developing CAAs and various operating systems that CAAs run on.

The use of ontologies to model context augments the development process of pervasive services with additional complexity introduced by the work required for ontology specification and management. Therefore, to make the use of ontologies viable, development approaches need to be applied those are capable of tackling this complexity. Such an approach cannot be ad-hoc and proprietary but rather it must allow for rigorous/precise modelling of context ontology and for automatic development of ontology-based context-aware applications. To this end, we have been investigating the use of Model Driven Architecture (MDA) [12], the emerging standard by the Object Management Group (OMG) for software systems design and development in order to evaluate benefits of this approach in ontology development.

MDA aims at providing clear separation between technology-neutral and technology-specific concerns involved in the different stages of a system's development process. MDA [12] consists of a set of standards, namely, MOF, OCL, XMI and QVT [18] that enable the definition of Domain Specific Languages (DSLs) used to specify a system's structure and behaviour. DSLs are represented as meta-models based on the Meta-Object Facility (MOF) and can be precisely defined using the Object Constraint Language (OCL). OCL allows the definition of constraints over meta-models as well as actual models for a specific system.

We have applied MDA in a number of case-studies that demonstrated the advantages the approach offers in the development process of systems and services [3, 4, 5]. In [3] and [4] we discussed the use of MDA for context-aware pervasive service modelling, provisioning and service composition. In [5] we presented how MDA is used for the design, development and integration of telecommunications Operations Support Systems (OSS) and the benefits gained in terms of improved quality, rapid delivery and lower development costs.

The above experiences lead to the conclusion that MDA can be a beneficial paradigm for capturing context ontologies with a number of advantages. Modelling ontologies as Platform Independent Models (PIMs) can be a one-off activity as these PIMs (models of roles, devices, and tasks) can be re-used in the development of other CAAs. Heterogeneity is also catered for since ontology and CAA PIMs can be transformed into implementations suitable for the platforms and devices at hand. MDA can facilitate the semi-automatic or automatic generation of ontology-based CAAs with significant reductions in time and costs during the development and maintenance phases.

This paper presents an MDA-based approach for context ontology modelling towards the development of context-aware applications for pervasive systems. To the best of our knowledge, no previous work has made use of MDA in this field. The primary contributions of our work are: 1) a context ontology model (COM) for pervasive services based on the RDFS and OWL meta-models; 2) a model driven integration architecture (MDIA) for ontology-based CAA development.

The rest of the paper is organized as follows. The next section presents related work in ontology based context modelling for pervasive services. Section 3 describes the context ontology models developed using MDA. Section 4 illustrates our Model Driven Integration Architecture (MDIA) for ontology-based CAA development. Section 5 provides some concluding remarks and plans for future work.

2 Related Work

Related research has dealt with the issue of ontology-based context modelling and reasoning in a number of perspectives. Wang et al [9] proposed an OWL-encoded ontology (CONON) for modelling and reasoning about context in pervasive computing environments. Chen et al [10] proposed an architecture called Context Broker Architecture (CoBra) that uses OWL to define ontology in intelligent environments. Furthermore, they proposed a Standard Ontology for Ubiquitous and Pervasive Computing Applications (SOUPA) [15]. Henricksen et al [11] proposed a hybrid approach for context modelling, reasoning and interoperation between object-oriented context models and ontology-based context models. All the above referenced research illustrated the advantages of handling context at a semantic level by using different solutions. However, no evidence was found of any solutions trying to model ontology in the context of MDA.

Other research work focuses on ontology-based CAA development. Biegel and Cahill proposed a framework to develop CAAs based on their sentient object model. They focus on fusing data from disparate sensors to ease context-aware application development by simple coding [16]. McFadden et al [17] proposed a model driven approach to develop CAA based on their object-oriented Context Modelling Language (CML). These practices are aiming to reduce the development effort or to automate the CAA development process through specific and proprietary mechanisms.

In our work, a pure MDA-based approach has been applied for context ontology modelling that is based on well-recognized OMG standards, such as MOF, OCL, and XMI and on OMG's recent efforts regarding ontology modelling, the Ontology Definition Meta-Model (ODM) [8], which deals with modelling and engineering of context information in the pervasive services domain.

3 Context Ontology Modelling

This section presents how ontologies are captured using the four layers of abstraction that MDA adopts. In the MDA paradigm, ontology languages need to be abstracted and expressed using MOF in the form of meta-models. Based on these meta-models, we then construct our Context Ontology Model (COM). COM consists of the

Upper-Level Context Ontology Model (ULCOM) and the Extended Specific Context Ontology Model (ESCOM) and is used to model context information. We employed an MDA tool, XMF, from Xactium[1] in our modelling work.

3.1 Ontology Meta-modelling

MDA is based on four layers of abstraction, M0 through M3. M0 contains application run-time data; M1 contains application models designed for a specific problem domain; M2 contains meta-models that capture domain specific languages (DSLs) used in the application designs of M1; M3 hosts the Meta-Object Facility (MOF), which serves as a language to specify DSLs.

Fig. 1 shows how the ontology models and meta-models are positioned around the above four layers. M2 hosts the MOF-based Ontology Definition Meta-model (ODM) and the UML profile for Ontology. Domain Ontology Models are situated on M1 and are instances of ODM representing models of domain-specific ontologies. An example of a domain ontology model is the Context Ontology Model (COM) introduced in the next section. M0 contains models that are instances of M1 domain-specific ontologies.

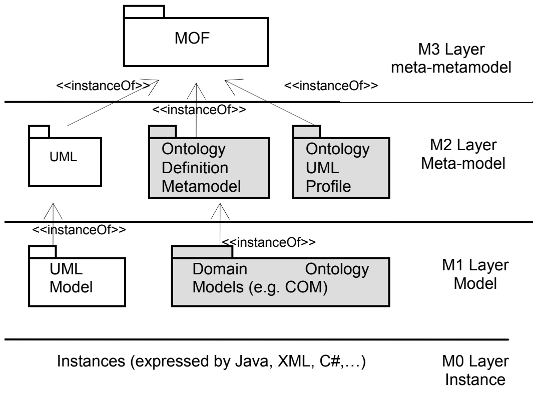


Fig. 1. Ontology Modelling in MDA Four-layer Architecture

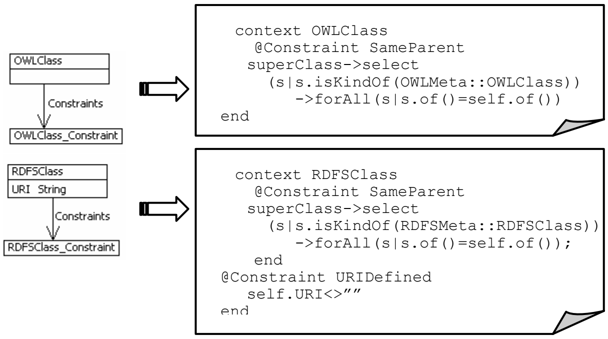
To enable ontologies become machine-interpretable, they need to be represented as software artifacts. To achieve this in MDA, the primary elements of ontology need to be abstracted out and be represented as a meta-model using MOF.

Several efforts have already been made towards ontology meta-modelling in the MDA paradigm. Fuchs et al proposed a meta-model specification for OWL DL [6]; Duric et al proposed a meta-model for Semantic Web ontology [7]. OMG launched a request for proposals (RFP) regarding an Ontology Definition Meta-model (ODM). The latest adopted submission of ODM proposal is available on [8]. All this work aims to use MDA standards for ontology engineering. Our ontology meta-modelling work presented in this section is compliant to [8].

[1] Xactium: www.xactium.com

Based on RDF, RDFS, OWL and ODM, we constructed the RDFS Meta-Model and OWL Meta-Model; both are MOF-based meta-models that allow users to define ontology models using the same terminology and concepts as those are defined in RDFS and OWL, respectively.

One challenge that characterizes the definition of MOF-based ontology meta-models is how to make these meta-models precise enough so that ontology model definitions on M1 are unambiguous. We tackle this by means of the Object Constraint Language (OCL) that is used to specify constraints on ontology meta-model elements against which ontology models' consistency can be checked.



```
  context OWLClass
    @Constraint SameParent
   superClass->select
     (s|s.isKindOf(OWLMeta::OWLClass))
        ->forAll(s|s.of()=self.of())
end
```

```
  context RDFSClass
    @Constraint SameParent
   superClass->select
     (s|s.isKindOf(RDFSMeta::RDFSClass))
        ->forAll(s|s.of()=self.of());
    end
@Constraint URIDefined
   self.URI<>""
end
```

Fig. 2. Example of Constraints in Meta-models

Fig. 2 depicts two examples of constraints in ontology meta-models. `OWLClass`, an entity of the OWL meta-model, is augmented with constraint `SameParent`. This constraint coerces any `OWLClass` instance A to only subclass a class B if and only if B is also an instance of `OWLClass` and does not instantiate any other meta-model entity. In the OCL scripts, `s.of()` is used to get the superclass of an entity. Fig. 2 also shows constraint `URIDefined` imposed on class `RDFSClass`, which specifies every `RDFSClass` must have a non-empty URI (Uniform Resource Identifier) defined.

3.2 Context Ontology Model (COM)

An ontology of context represents knowledge about the context domain and comprises definitions of a set of *context entities*, the entity *attributes*, the *functions* the entities provide, the *relationships* between context entities, the *instances* of context entities and the *axioms* used for context reasoning.

We have defined COM that describes context for pervasive services. COM consists of two parts, namely, the Upper-Level Context Ontology Model (ULCOM) and the Extended Specific Context Ontology Model (ESCOM). ULCOM captures an ontology of concepts that are essential for generically characterizing context in the pervasive services domain. The ULCOM specification uses the RDFS/OWL meta-models. ESCOM defines specific concepts for context as extensions of ULCOM entities. Fig. 3 depicts a part of COM.

ULCOM includes three core concepts, namely, *Entity*, *EntityProperty*, and *EntitySpecification*:

- **Entity,** stereotyped as `OWLClass`, represents five types of context concepts that are usually involved in a typical pervasive service – *person, device*, communication-channel (*ComChannel*), *function*, and *event*.
- **EntityProperty:** Apart from the proprietary attributes an entity may have, EntityProperty is also used to characterize general attributes, such as, *time*, *identity*, *activity*, and *location*. These attributes are necessary to determine the when, who, what, and where type of knowledge relating to an entity. EntityProperty is a type of `OWLProperty`.
- **EntitySpecification** models the configuration of each entity and entity property in terms of constraints. It is an instance of `OWLRestrictions` and contains OCL scripts for constraints definition and model checking.

For simplicity, there are only a few of relationships depicted in Fig. 3. For instance, a person *owns* devices and a person is *nearby* another person.

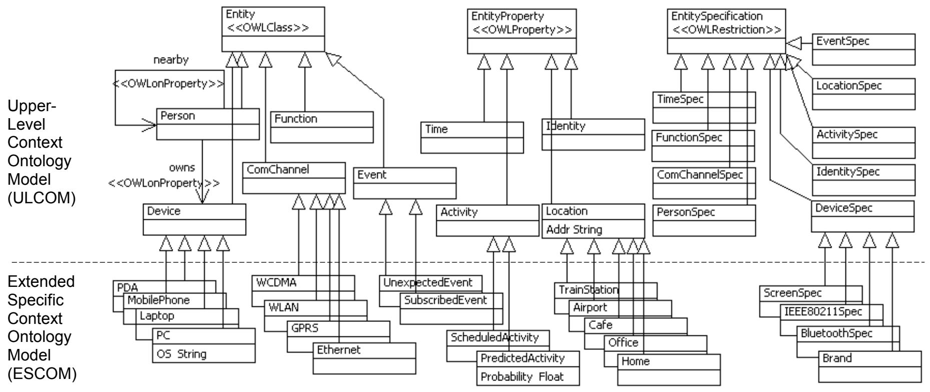


Fig. 3. A Part of the Context Ontology Model (COM)

ESCOM and ULCOM are M1 layer models. ESCOM is used to define more specific context entities and their corresponding properties and specifications. Some examples of ESCOM entities are *PDA, laptop, PC, mobile-phone* and *TV* which are devices normally used in a pervasive computing environment. These devices have specifications that define certain constraints on device features, e.g. *ScreenSpec*, or configurations of the device to support different types of network access e.g. *BluetoothSpec* and *IEEE80211Spec*. Further concepts in the ESCOM include different types of activities, such as *ScheduledActivity* or *PredictedActivity*, different types of locations, such as *home, office* or *café,* types of events that may emerge, e.g. *SubscribedEvent* and *UnexpectedEvent*, and types of communication channels supported by the devices or the user locations, e.g. *WLAN* and *GPRS*.

4 MDA-Based Context-Aware Application Development

This section presents our MDA-based approach for Context-Aware Application (CAA) development. Context ontology alone is useful but not sufficient to entirely support CAA development as it only captures knowledge about the CAA context. For CAA it is necessary to further specify models describing the application logic, the graphical user interfaces (GUI), the application data and the way the CAA integrates with other systems and services. Therefore, alongside COM, more meta-models have been developed to facilitate the automatic generation of CAAs.

Fig. 4 gives an overview of our Model Driven Integration Architecture (MDIA) for CAA development. At the meta-model layer there are three categories of artifacts: *CAA integration related meta-models, implementation languages meta-models*, and *mappings* between the meta-models.

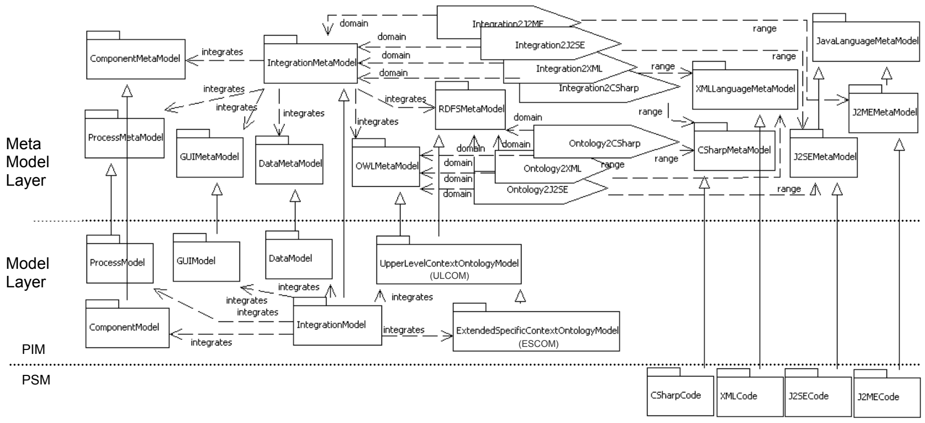


Fig. 4. Model Driven Integration Architecture (MDIA) for Context-Aware Application Development

The CAA integration related meta-models category includes the following six packages:

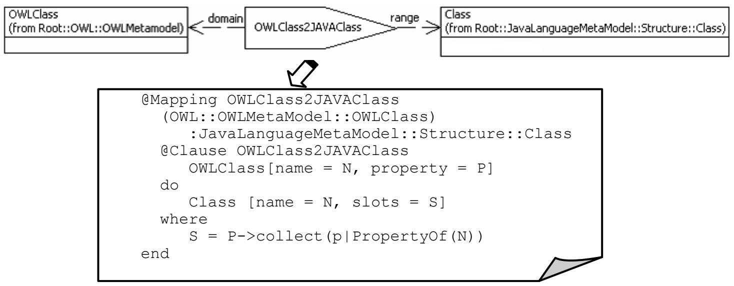
- **ComponentMetaModel** defines a language to model functional interfaces of existing functional components (such as ontology reasoning components in our application domain, or inventory components in OSS systems [5]). Using this language we can model at the M1 level ontology handling functionality of Commercial-Off-The-Shelf (COTS) components (or libraries) which we can then integrate into the models of CAAs.
- **ProcessMetaModel** represents a language that can be used on M1 to specify system logic in the form of a process. The meta-model defines elements of a UML activity diagram.
- **RDFSMetaModel and OWLMetaModel** are used to define context-aware ontology data in our architecture.
- **GUIMetaModel** defines basic elements of a language to describe a graphical user interface, such as window, label and textbox and an event-based model describing the dynamic way GUI elements can trigger logic associated with them.
- **DataMetaModel** describes a language for the specification of application-related data on M1. This meta-model is based on the UML class diagram.

- **IntegrationMetaModel** is fundamental as it defines the way all previous meta-models associate and integrate. It serves as the glue that brings all necessary elements together in order to compose a CAA. More specifically, this meta-model defines how (1) a flow of process activities integrates different components by invoking certain operations on each component to deliver an activity; (2) GUIs integrate with processes by events GUI elements generate and trigger process activities or entire processes representing the logic behind these elements; (3) data integrates with both components and processes that consume and produce information of different types.

All above CAA integration related meta-models are tools/languages that facilitate the technology-neutral specification of CAAs. In order to enable the generation of technology-specific CAA implementations, it is important to introduce another category of meta-models, namely, implementation languages meta-models. In this category, we defined *JavaLanguageMetaModel*, *XMLLanguageMetamodel*, and *CSharp-MetaModel,* which constitute specifications of the respective languages' syntax, including grammars, expressions, statements and programming structures (classes, operations, variables etc). It is worth to note that *J2SEMetaModel* and *J2MEMeta-Model* are defined which are extensions of *JavaLanguageMetaModel.* They are specifying to two sub-sets of Java language meta-data for generating the Java implementations for Standard and Micro Edition platforms, respectively.

What is still missing before MDIA is completely enabled to automatically generate CAA implementations is specifying precise transformations of technology-neutral into technology specific meta-models. More specifically, we define two types of mappings in the architecture:

- Mappings between integration and implementation language meta-models, namely, *Integration2Java, Integration2XML,* and *Integration2CSharp*. These are used to generate CAA implementations.
- Mappings between ontology language (*RDFSMetaModel* and *OWLMetaModel*) and implementation language meta-models, namely, *Ontology2Java, Ontology-2XML,* and *Ontology2CSharp.* These are used to generate technology-specific representations of ontological artifacts in the specified implementation languages.


OWLClass
(from Root::OWL::OWLMetamodel)
domain
OWLClass2JAVAClass
range
Class
(from Root::JavaLanguageMetaModel::Structure::Class)

```
@Mapping OWLClass2JAVAClass
  (OWL::OWLMetaModel::OWLClass)
     :JavaLanguageMetaModel::Structure::Class
  @Clause OWLClass2JAVAClass
     OWLClass[name = N, property = P]
  do
     Class [name = N, slots = S]
  where
     S = P->collect(p|PropertyOf(N))
end
```

Fig. 5. A Mapping Example

Fig. 5 shows an example of a mapping specification that transforms `OWLClass`, of the `OWLMetaModel`, into `Class`, of the `JavaLanguageMetaModel`. The mapping script is written in XMap, the proprietary language of the XMF tool to define transformations. XMap uses pattern matching and the particular script of the example maps an `OWLClass` with a name and properties onto a Java class that has the same name and variables (slots) as the `OWLClass`.

Utilising the meta-models presented above, a designer can now specify the model of a CAA at the M1 layer. Fig. 4 illustrates the model layer being populated by generic forms of integration related model packages, corresponding to the CAA integration related category of meta-models that specify all aspects of a platform independent model for the CAA. These aspects are application logic (*ProcessModel* that can reuse and integrate COTS capabilities described in *OntologyComponentModel*), data (*DataModel*), context (*ULCOM* and *ESCOM*), GUIs (*GUIModel*) and the ways all aspects integrate (*IntegrationModel*). Rigorous specification of the CAA PIM allows for the automatic generation of complete PSMs represented in various implementation languages. Fig. 4 illustrates packages *JavaSourceCode*, *CSharpCode*, and *XMLCode* in the PSM of the model layer, that respectively include the code in Java, C# and XML representation of the CAA PIM as they are automatically generated by the correspondent transformations defined in the meta-model layer. For instance, *JavaSourceCode* results from the execution of the *Integration2Java* mapping that transform instances of the integration related meta-models into Java code. Similarly, the *Ontology2Java* mapping generates a Java code representation of an ontology.

5 Conclusion and Future Work

Our primary goal in this paper is to explore the feasibility of amalgamating UML, MDA and ontology languages (such as RDFS and OWL) towards context ontology modelling and an MDA-based integration architecture for automatic development of context-aware applications aiming at improving the accuracy and reducing time and costs.

We presented our Context Ontology Model (COM) which can be validated against precise meta-models. The Model Driven Integration Architecture (MDIA) is designed to integrate different types of DSLs and technologies. For instance, under the umbrella of the MDIA, GUI models, process models and ontology models can be integrated to build a platform independent CAA model representing user interfaces, business logic and ontology-based context data involved in the CAA.

This paper only presents the first step of our work towards a model driven ontology-based pervasive service engineering platform. As part of our future work, a comprehensive case study on ontology-based pervasive service provisioning is to be carried out to evaluate the new challenges introduced by ontology and MDA amalgamation. Our next big step will be the application of our MDA-based ontology approach to the design and integration of enterprise information systems in the telecom OSS domain.

References

1. A.K. Dey, D. Salber and G.D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), pp. 97-166, (2001)

2. P.J. Brown, J.D. Bovey and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace", IEEE Personal Communications, 4(5), 58-64 (1997)
3. K. Yang, S. Ou, A. Liotta and I. Henning, "Composition of Context-aware Services Using Policies and Models", Proc. of IEEE GlobeCom 2005, IEEE Press, Dec. 2005, St. Louis, USA (2005)
4. K. Yang, S. Ou, M. Azmoodeh and N. Georgalas, "Policy-based Model-driven Engineering of Pervasive Services and the Associated OSS", BT Technical Journal (BTTJ), Vol 23, No 3, pp. 162-174 (2005)
5. N. Georgalas, M. Azmoodeh and S. Ou, "Model Driven Integration of Standard Based OSS Components", Proc. of the Eurescom Summit 2005 on Ubiquitous Services and Applications, Heidelberg, Germany (2005)
6. F. Fuchs,I. Hochstatter,M. Krause and M. Berger, "A Meta-model Approach to Context Information", Proc. of the 3rd Int'l Conf. on IEEE Pervasive Computing and Communications Workshops, pp. 8-14 (2005)
7. D. Duric, D. Gasevic and V. Devedzic, "A MDA-based Approach to the Ontology Definition Meta-model", Proc. of a 4th Workshop On Computational Intelligence And Information Technologies,Serbia (2003)
8. IBM and Sandpiper Software, Inc., "Ontology Definition Meta-model", http:// www.omg.org/docs/ad/05-08-01.pdf (2005)
9. X.H. Wang, T. Gu, D.Q. Zhang and H.K. Pung, "Ontology-Based Context Modelling and Reasoning using OWL", Context Modelling and Reasoning Workshop at PerCom (2004)
10. H. Chen, T. Finin and A. Joshi, "Using OWL in a Pervasive Computing Broker", In Proc. of Workshop on Ontologies in Open Agent Systems (AAMAS 2003) (2003)
11. K. Henricksen, S. Livingstone, and J. Indulska, "Towards a hybrid approach to context modelling, reasoning, and interoperation", Proc. of the 1st Int'l Workshop on Advanced Context Modelling, Reasoning And Management, UbiComp'2004 (2004)
12. Object Management Group (OMG), Model Driven Architecture, http://www.omg.org/mda
13. D.L. McGuinness and F. Harmelen, "OWL Web Ontology Language Overview", W3C Recommendation, http://www.w3.org/TR/owl-features/ (2004)
14. D. Brickley and R.V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema", W3C Recommendation, http://www.w3.org/TR/rdf-schema/ (2004)
15. H. Chen, F. Perich, T. Finin and A. Joshi, "SOUPA: Standard Ontology for Ubiquitous Pervasive Applications", Proc. Of the 1st Int'l Conf. on Mobile and Ubiquitous System, IEEE (2004)
16. G. Biegel, V. Cahill, "A Framework for Developing Mobile, Context-aware Applications", Proc. of 2nd IEEE Conf. on Pervasive computing and Communications (2004)
17. T. McFadden, K. Henricksen, J. Indulska, "Automating context-aware application development", First Int'l workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004, England (2004)
18. QVT Partners. Initial Submission for MOF 2.0 Query/View/Transformations RFP, QVT-Partners, http://qvtp.org/downloads/1.1/qvtpartners1.1.pdf (2003)