Journal of Universal Computer Science, vol. 16, no. 15 (2010), 2111-2138
submitted: 31/1/10, accepted: 28/7/10, appeared: 1/8/10 © J.UCS

CAUCE: Model-driven Development of Context-aware Applications for Ubiquitous Computing Environments

Ricardo Tesoriero, José A. Gallud, María D. Lozano, Víctor M. R. Penichet
(University of Castilla-La Mancha, Albacete, Spain
[ricardo.tesoriero, jose.gallud, maria.lozano, victor.penichet]@uclm.es)

Abstract: In order to develop context-aware applications for ubiquitous computing environments we have defined an MDA approach that defines three layers of models. The first layer captures the conceptual characteristics of the application. This layer defines three complementary points of view of the system that are used to build the task, space and social views of the system. The second layer defines the software characteristics of the application.
It is composed by three new complementary points of view of the system that are used to build the referential space, the information flow and the entity context views of the system. Finally, the third layer defines the deployment environment of the system according to the views generated by the second layer.

Keywords: context awareness, ubiquitous computing, model-driven architectures
Categories: D.2.2, D.2.6, D.2.11, D.2.13

1 Introduction

The definition of context-aware applications (CAA) s has evolved through the time.
At the beginning, the difference between context-aware and location-aware applications was fuzzy. Most CAAs were location-aware applications [Want R. et al., 1992, Adams N. et al., 1993, Want R. et al., 1995].

Nowadays, location-aware applications may be considered a subset of CAAs because location is just a characteristic of the application context [Schmidt A. et al., 1999b, Schmidt A. et al., 1999a, Dey A. K. and Abowd G. D., 2000, Dey A. K., 2001].

A characterization of CAAs is exposed in [Schmidt A., 2002] where a feature-space working model was defined. This model defines two types of context characteristics: the human factors and the physical environment.

On the one hand, the characteristics that are related to human factors are divided into three categories: the information on the user (knowledge of habits, emotional state, bio-physiological conditions …), the user's social environment (co-location of others, social interaction, group dynamics…) and the user's tasks (spontaneous activity, engaged tasks, general goals…).

On the other hand, the characteristics that are related to the physical environment are also divided into three categories: location (absolute position, relative position, co-location…), infrastructure (surrounding resources for computation, communication, task performance…) and physical conditions (noise, light, pressure…).

Finally, the working model defines a feature that affects all these characteristics, the time.

Thus, this characterization represents the conceptual features of CAAs. However, in order to develop a system we have to take into account some features that are not directly related to CAAs features, these features are the software characteristics of the system.

These characteristics are mainly affected by the concept of ubiquitous computing environment (UCE) and Calm Technology [Weiser M. and Brown J. S., 1997]. Therefore, from the software point of view, a context-aware system (CAS) running in an UCE is not defined by a single application running stand alone, it is rather composed of a set of applications that exchange information as a distributed system.

Besides, although most of these systems are usually implemented by client-server architecture, they are not restricted to this architecture. Therefore, these systems do not follow a fixed architecture like other systems, such as Web applications. Even more, these systems are usually deployed in heterogeneous platforms, too. It means that the same application may be deployed in different platforms, even at the same time, according to the device that will run it.

The connectivity is also a key factor to board because not all devices are connected in the same way. The way devices are connected is tied to their capabilities. For instance, remote controls employ IrDA technology; PDAs and mobile phones may be connected via Bluetooth or Wi-Fi; portable computers or micro PCs use Wi-Fi networks; and smaller devices, such as RFID readers, may use wired connections, such as USB or RS232 links. Thus, connectivity affects the way information is transported from one device to another, resulting in different software implementations of the application according to the application deployment.

Thus, the problems we have to face in order to develop a CAS for UCEs can be summarized as:

1. The generation of multiple CAAs to be deployed on heterogeneous platforms of hardware and software.
2. The development of these applications embraces different aspects of the system (i.e. the social, spatial and behavioural) that are difficult to express from a single point of view.
3. The generation of a cooperative environment where entities of the system are able to exchange information in order to carry out system goals.

A MDA approach provides:

1. The possibility of defining software that is not tied to a platform, many applications can be deployed in different platforms due to the separation of the platform independent and specific models.
2. The possibility of defining software from different points of view, each view provides designers the ability to conceive software from different perspectives through the definition of different metamodels for each view of the system.
3. The use of model transformation provides designers with the ability to turn models that capture the conceptual views of the system into computational models that define how software artefacts interact with each other.

Therefore, due to the conceptual and software characteristics of CAAs for UCEs, the model-driven architecture (MDA) is the most suitable strategy to board the development of these applications.

The article begins exposing the motivation for the improvement of the development of CAAs for UCEs in Section 2. Then, it performs an analysis of current approaches in Section 3. Afterwards, Section 4 introduces the CAUCE methodology and explores the three layers that define it. Once the methodology was described, a case of study is presented to demonstrate the utility of the approach in Section 5. Finally, we expose conclusions and future work in Section 6.

2 Motivation

According to Weiser, CAA in UCE are an evolution of desktop computing and soon, these applications will be as usual as windows based interfaces are now for desktop applications.

Nowadays, mobile devices are very popular and affordable to almost any person in any developed society in the world (i.e. mobile phones are so popular that it is not easy to find someone who does not use it).

Besides, mobile phones have evolved; they also play the role of engagement book, address book, photo and video cameras, etc. Even more, tendencies point out that more and more functionality is being introduced into these devices as time goes by. Examples of this situation are Smart phones. They are the fusion of a PDA (Personal Digital Agenda) and a mobile phone allowing users to provide mobile communication to most popular desktop applications, turning these devices into real mobile offices.

This tendency was also encouraged by communication infrastructure improvements that have been really impressive for the last decades, leading us to think that in the near future these devices will be connected from almost everywhere in the whole world. One of the most relevant examples following this line is the Wi-Fi connectivity for entire cities.

Thus, communication and mobile device evolution provide software engineers with new scenarios that were unconceivable shot time ago when users were in front of a desktop computer. This technology provides users the ability to interact with the surrounding environment through these devices, instead of interacting with the computer. Thus, the environment becomes a key element to be explored when considering these types of applications.

The environment is an important part of the application context. Therefore, the development of CAA will face a huge market in the near future.

One of the most difficult obstacles to sort when developing these applications is the device capability to sense the environment, for instance, position, orientation, movements, etc. However, at the present these technological barriers are becoming invisible as time goes by. Actually, there are commercial products which allow users to get new information from the environment by being fused with some devices like GPSs, accelerometers, etc. These features provide developers with new resources to exploit.

Communication is also evolving concerning mobile devices. At first, mobile phones employed analogue lines to set up the communication. Later on, digital

technologies were applied. All these technologies were able to transport voice. However, they are actually able to communicate using several digital technologies. It is not strange to come across popular devices that support Bluetooth, Wi-Fi or IrDA communication facilities able to provide information exchange between desktop or portable computers and other mobile devices.

On the one hand, this technology, which is part of daily life, is not being fully exploited by actual software applications. On the other hand, context aware applications can take advantage of this technology to improve the calmness of applications in ubiquitous computing environments. Therefore, this work proposes a methodology to develop context-aware applications for ubiquitous computing environments using the MDA approach.

3 Motivation

The goal of this section is presenting a review most relevant related work to our proposal.

These works can be categorized into three groups according to the approach they use to board the problem:

1. Ontology Driven Development [Georgalas N. et al., 2007, Chen H. et al., 2003a, Chen H., 2003]
2. MDA using MOF [de Farias C. R. G. et al., 2007, Vale S. and Hammoudi S., 2008, Almeida J. P. A. et al., 2006]
3. MDAs on UML Profiles [Prezerakos G. N. et al., 2007, Carton A. et al., 2007]

3.1 Ontology-driven development

This section shows the most representative approaches to cope with the development of CAAs for UCEs based on ontologies.

3.1.1 Model-driven approach for ontology-based CAA development

A case study is developed using an ontology-based model-driven approach in [Georgalas G. N. et al., 2007] based on previous works like those exposed in [Brown P. J. et al., 1997, Kumar S. et al., 2000, Dey A. K. et al., 2001, Wang X. H. et al. 2004, Ou S. et al., 2006].

It exposes a model transformation mechanism for the generation of CAAs jointly with a case study on a context-aware pervasive service scenario that explains in deep detail how the proposed approach works in practice.

This work presents the development of CAAs following a defined sequence of steps introducing high-level concepts from the software perspective of the system.

However, from the conceptual point of view, some characteristics, such as task dependency and roles have not been taken into account.

3.1.2 Model-driven approach for ontology-based CAA development

In [Chen H. et al., 2003b, Chen H. et al., 2003a] a foundation ontology expressed in the Web Ontology Language (OWL) is defined to build a context-aware pervasive

computing framework known as CoBrA. CoBrA is an agent-based architecture for supporting context-aware computing in intelligent spaces [Chen H., 2003]. From the conceptual point of view, the CoBrA ontology models the basic concepts of people, agents, places, and presentation events in an intelligent meeting room environment introducing the notion of location as part of the model.

However, it does not express the relationship between tasks (i.e. task dependency) neither role dynamics. Finally, it provides a centralized solution that leads to a fixed architecture.

3.2 MDA using MOF

This section shows the most representative approaches to cope with the development of CAAs for UCEs based on MDA using MOF.

3.2.1 A MOF metamodel for the development of context-aware mobile applications

The MOF metamodel for the development of context-aware mobile applications proposed on [de Farias C. R. G. et al., 2007] was structured according to the Core and Service views of the system. This approach provides a contextual model that is independent from the application domain.

However, it does not provide high level abstraction elements to express conceptual characteristics. For instance, it does not provide concepts such as task, task dependency, space, role, and so on. From software characteristics perspective, it does not take into account architectural or deployment issues, because it is only based on the SOA architecture.

3.2.2 Context independence in distributed CAA

A proposal that encourages the context independence in distributed CAAs using a model-driven approach is exposed in [Vale S. and Hammoudi S., 2008]. It presents five sets of viewpoint concepts for distributed software development: the Enterprise viewpoint (focused on the business domain and processes), the Computation viewpoint (describing the implementation details), the Information viewpoint (defining information semantic, representation and constraints), the Engineering viewpoint (focused on distributed characteristics of the system) and the Technology viewpoint (defining the target platform and hardware elements).

However, this approach does not provide high-level abstraction elements to express conceptual characteristics. Besides, although this proposal is quite complete from the software characteristics point of view, it is based on fixed service oriented architecture.

3.2.3 MDD of context-aware services

A model-driven design trajectory for context-aware services consisting of three levels of models with different degrees of abstraction and platform independence is defined in [Almeida J. P. A. et al., 2006]. This work divides the development in two phases (preparation and service creation phase). Besides, the modelling process is also

divided into service specification, platform - independent service design and platform specific service-design.

However, this approach does not provide high-level abstraction elements to express conceptual characteristics; it only defines low-level abstraction concepts, such as, events, queries and actions. Thus, although tasks may be modelled as services, task dependencies, roles or location issues are not taken into account.

3.3 MDAs on UML Profiles

This section shows the most representative approaches to cope with the development of CAAs for UCEs based on MDA using UML Profiles.

3.3.1 Model-driven composition of context-aware web services using ContextUML and Aspects

A model-driven composition of context-aware web services using aspects is exposed in [Prezerakos G. N. et al., 2007]. It argues that the interaction between the end-user and the service can be adapted to contextual parameters without affecting the overall goals related to the service logic. This proposal decouples core service logic from context handling by adopting the model-driven architecture (MDA) approach in the design phase and the Aspect Oriented Programming (AOP) [Elrad T. et al., 2001] during coding.

However, this approach does not provide high-level abstraction elements to express conceptual characteristics. For instance, it provides services, instead of entities or tasks. From the software characteristics point of view, the solution is focused on Web applications, so it does not provide a suitable solution for UCEs due to the lack of deployment features.

3.3.2 Aspect-oriented MDD for mobile context-aware computing

A combination of aspect-oriented development techniques and model-driven development (MDD) is presented in [Carton A. et al., 2007] as an approach to develop CAAs for UCEs. This paradigm provides a set of techniques to modularize crosscutting behaviour.

Main disadvantage of this approach is the lack of support for high-level abstraction elements to express conceptual characteristics.

Thus, to cope with all deficiencies detailed in previous approaches we propose the definition of a MDA based on MOF based on three levels of abstraction to:

1. Describe CAA conceptual characteristics from the task, space and social point of view.
2. Describe CAA software characteristics in terms of entities that exchange information.
3. Describe CAA deployment environment.

4 The CAUCE methodology

Due to the development weaknesses of the existing approaches described in Section 3 we have described the CAUSE proposal depicted on Figure 1. This figure divides the development process into three layers.

The analysis layer is related to the Computation Independent Model (CIM) of the MDA. It provides developers with the ability to represent the conceptual characteristics of CAAs exposed in [Schmidt A. et al., 1999b]. These characteristics are represented by three different points of view. Each point of view is represented by the task, the space and the social metamodels defined in the Essential Meta Object Facility (EMOF) [OMG, 2004] and the Object Constraint Language (OCL) [OMG, 2006].The combined models represent the conceptual view of the system.

The information layer is related to the Platform Independent Model (PIM) of the MDA. It provides developers with the ability to represent the software characteristics of CAAs (deployment, architecture and communication). These characteristics are represented by three different points of view. Each point of view is represented by the information flow, referential space and entity context metamodels defined in EMOF and OCL. The combined models represent the software view of the system.

In order to convert the conceptual representation of the CAA into a software representation of the system, a multi-model transformation is used. This transformation is defined using the Atlas Transformation Language (ATL) [Bzivin J. Jouault F. and Valduriez P., 2008]. It “interprets”' the model that is the result of merging the analysis layer models and turns them into a set of models conforming information layer metamodels.

In order to turn the PIM into source code we have defined the Mapping metamodel that defines the Platform Specific Model (PSM) of the system. Thus, the set of models conforming to the information layer models jointly with the Mapping model are turned into source code through a multi-model to text transformation defined using the MOFScript language [Eclipse Foundation, 2009].

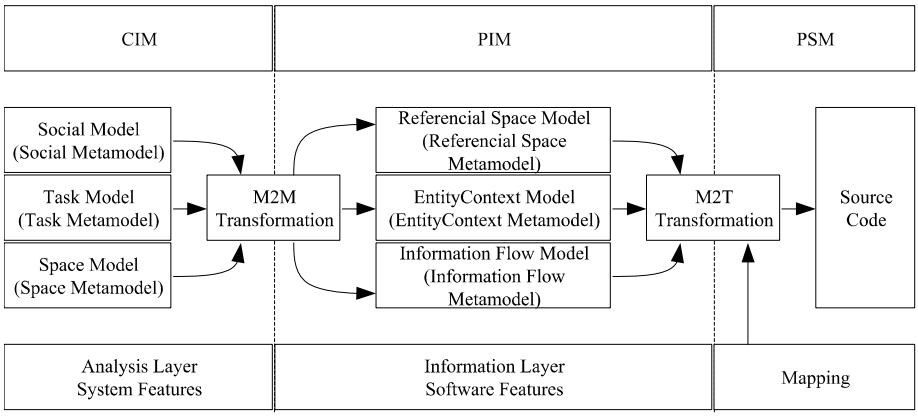

Figure 1: The CAUCE overview

4.1 The computation independent model

This layer is a bridge between requirements and the analysis phase of traditional development processes. The goal of this layer is the provision of modelling tools to capture conceptual features of CAAs defined in [Schmidt A. et al., 1999b]. Therefore, three metamodels have been defined to accomplish this goal.

- The task metamodel defines the set of tasks and the relationships among them that any entity in the system is able to perform.
- The social metamodel defines the social environment of the entities in the system. Therefore this metamodel is directly related to the social environment, entity task and entity information characteristics of the CAAs.
- The space metamodel defines the physical environment of the entities in the system. Therefore this metamodel is directly related to the physical conditions, infrastructure and location characteristics of the CAAs.

The core of the CIM is defined by the task metamodel. It combines the information of the social and space models into the task environment defining a temporal relationship. This relationship is achieved through regular expressions that relate both, the social and space models to the task model. These regular expressions provide developers with the ability to define conditions that are used to restrict the task execution to entities according to contextual facts. The combination process is depicted in Figure 2.

4.1.1 The task model

According to [Dey A. K., 2001], a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. This definition highlights the importance of two main concepts: task and entity. An entity represents any element that influences the system behaviour in any way. On the other hand, a task represents a work to be performed by an entity in order to fulfil a set of goals.

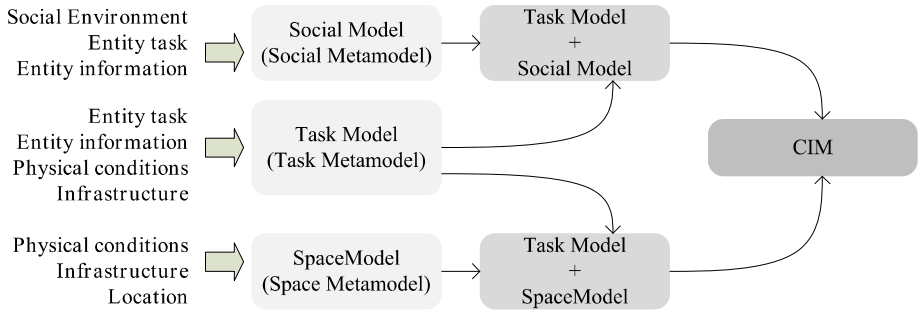

Figure 2: The CAUCE Computation Independent Model

Next, we describe the main concepts that the task metamodel is able to represent:

- The task dependency is related to the order tasks must be performed. For instance, some tasks are required to be performed before performing others; and some tasks may be performed currently.
- The task synchronization is related to task coordination to accomplish some goal. In highly distributed and dynamic environments, as ubiquitous computing are, it is a key issue.
- The work session is a period of time during which a group of entities collaborate to fulfil system goals; for instance, a chat session.
- The entity is an autonomous element that performs tasks to fulfil system goals. For instance, a browser session may be considered as an entity session.

The representation of these concepts is based on Petri Nets [Desel J. and Juhàs G. 2001, Peterson J. L., 1981, Petri C. A., 1962] and Workflows [WfMC, 1996] where transitions represent tasks and places define the state of the system according to the state of the system entities.

The basic pattern to build task models is depicted in Figure 3. The most relevant entities of the task metamodel are depicted in Figure 4. Finally, an example of the domain specific language for this metamodel is depicted on Figure 14 in Section 5.

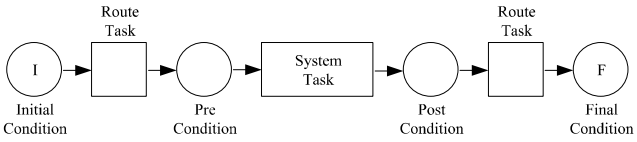

Figure 3: A task model sample

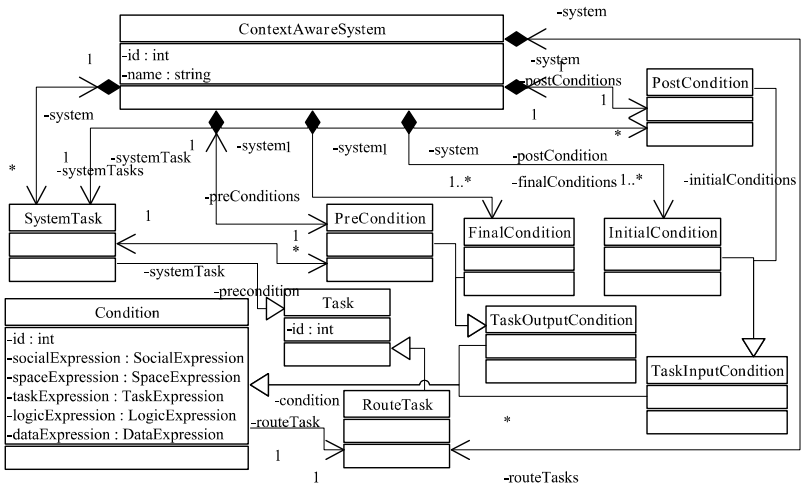

Figure 4: The task metamodel basic structure

4.1.2 The social model

According to [Schmidt A., 2002] the user information, the social environment and the user tasks are key characteristics of CAAs.

From our perspective, users interact with the system through some kind of device that, at least, turns physical stimulus into data that is interpreted by the system or physical actions that modifies the environment. Therefore, we base the social metamodel on a simplified version of the Organizational Structure Diagram (OSD) defined in [Penichet V. M. R., 2007].

Thus, this metamodel defines the following basic concepts:

- The role concept is related to the identification of a set of entity characteristics avoiding the reference to a particular entity. It is the analogous of "role concept" described in [Sunagawa E. et al., 2006, Kozaki K. et al., 2005] or the "Role" described in [Penichet V. M. R., 2007].
- The instance concept is related to the capability to represent an individual entity in a working session. This concept is analogous to the "Individual" actor described in [Penichet V. M. R., 2007].
- The instantiation represents the relationship between a role and an instance defining instance characteristics. It is close related to the "is-a" relationship defined by [Sunagawa E. et al., 2006, Kozaki K. et al., 2005] or the "plays" relationship defined by [Penichet V. M. R., 2007].
- The specialization represents relationships between roles that have "similar" sets of characteristics. If two roles, A and B are related through a Specialization from A to B, then B has the same characteristics of A, and therefore it is able to perform tasks performed by A. This concept is related to the "part-of" relationship defined by [Sunagawa E. et al., 2006, Kozaki K. et al., 2005] or the "hierarchy" relationship defined by [Penichet V. M. R., 2007].

The representation of these concepts is defined by a graph-based DSL where nodes represent roles and instances, and edges represent instantiations and specializations. The social metamodel is depicted in Figure 5.

Finally, an example of the domain specific language for this metamodel is depicted on Figure 16 in Section 5.

4.1.3 The space model

The space concept is a key issue to take into account in CASs. Thus, the space concept is used to characterize entity locations. Besides, system infrastructure is usually located somewhere in order to be used. Physical conditions are usually bound to a physical space, too. Finally, the time is a crucial issue to take into account in order to track entity activities.

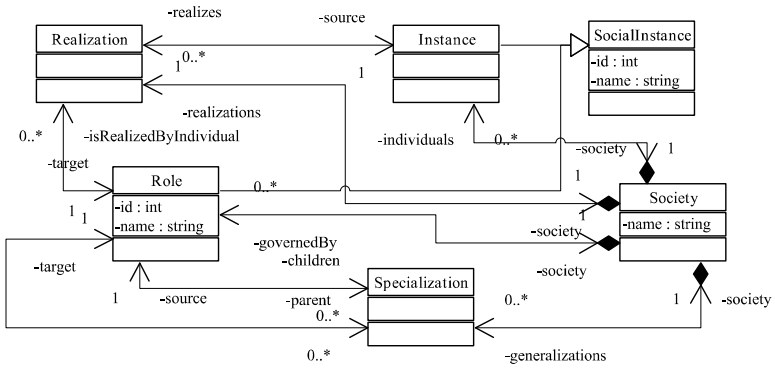

Figure 5: The social metamodel

Therefore, the main concepts to be modelled by this metamodel are the following:

- The positioning reference provides a generic way to position entities into the space. Thus, the expression of entity position may be boarded by different approaches: absolute, relative and even hybrid.
- The time is a key issue to position autonomous entities. From our point of view, the position is related to both location and time.
- The space granularity provides a natural an implicit location reference defined physical contention of spaces. For instance, if we are able to locate an entity into a room, we are able to locate the building it is location, just by the physical containment of spaces.
- The space characteristics define common features in a set of spaces that are not physically related. For instance, we may relate spaces by functionality (i.e. bathrooms), or by arbitrary attributes (i.e. a VIP zone).

The representation of these concepts is defined by a graph-based DSL where nodes represent spaces, and edges represent relationships among them.

Two types of spaces were defined: physical and virtual spaces. While physical spaces represent real world spaces, virtual spaces are usually used to characterize those spaces that do not have a physical relationship.

Besides, three types of relationships have been defined: physical containment, space grouping and space generalizations. Physical containments represent the physical space structure; for instance, a floor of a building is composed by rooms. Space grouping is used to represent the containment of that are not physically related; for instance, “hot spots” on user interfaces. Finally, space generalizations are used to define spaces with common features. For instance, a virtual space can be used to define the V.I.P. rooms of a building.

The space metamodel is depicted in Figure 6.

Finally, an example of the domain specific language for this metamodel is depicted on Figure 15 in Section 5.

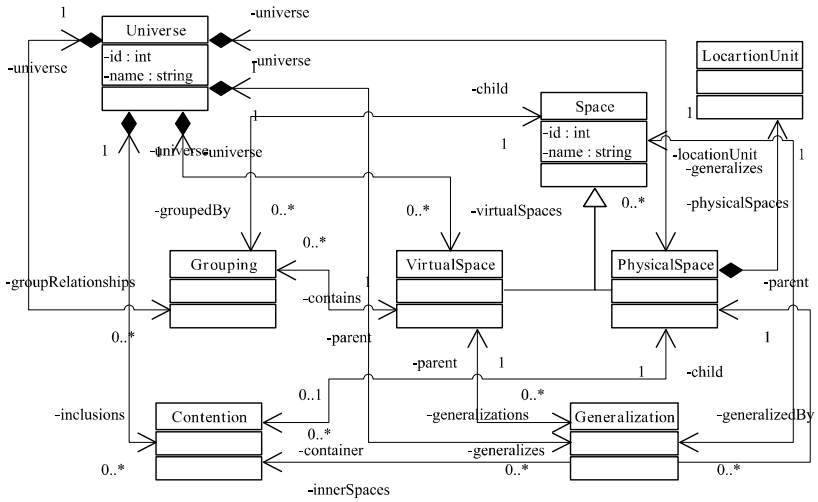

Figure 6: The space metamodel

4.2 The platform independent model

This layer represents the CAS characteristics captured in the CIM in terms of computation artefacts defines by a set of metamodels. Therefore, the goal of this layer is the provision of modelling tools to represent the context of an entity from the software point of view.

According to [Schilit B. et al., 1994, Schilit B. and Theimer M., 1994, Dey A. K., 2001] the CAAs for UCEs may be modelled as entities that exchange information, and affect and are affected by the environment. Thus, entities are a key concept addressed by this approach.

In order to model entities, two set of views have been defined: the inter-entity and intra-entity views.

On the one hand, the inter-entity views are defined by the information flow and referential space views of the system. The information flow view provides a description of the inter-entity communication. It is focused on how entities exchange information among each other. In addition, the referential space view describes the entity instance dependence. It is focused on relationships among entities, such as the referential knowledge and the existence.

On the other hand, the inter-entity views are defined by the entity context and entity core views of the system. The entity context view describes how the entity perceives the environmental. It is focused on the situations, and the conditions that define them, that affect the entity perception of the environment.

Finally, the entity core view defines the business logic of the entity. This metamodel is not boarded by our approach because it depends on the entity business domain. This metamodel is the most important the extension point of the

methodology because it provides designers the ability to adapt the context to any business domain.

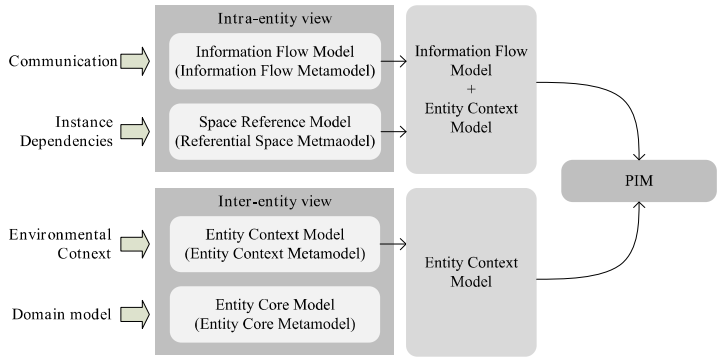

Figure 7: The CAUCE Platform Independent Model

The Figure 7 depicts the relationships among the views of the CAUCE PIM.

4.2.1 The information flow model

The information flow model expresses the communication among entities in the CAS. Therefore, the main goal of this metamodel is the description of the information flow among entities of the system.

The modelling approach was inspired by the producer--consumer view of CAAs proposed in [Zimmer T., 2004]. Thus, entities consume information produced by the environment and produce information that is consumed by the environment.

The main addressed by this metamodel are the following:

- The entity represents an active component of the system that is able to perform system tasks in order to carry out system goals. An entity is able to both, produce and consume information flows.
- The information flow is the representation of the communication between entities. Information flows define a source and a target entity. The source entity produces the information and the target entity consumes it.
- The data delivery is related to how information is delivered to the entities through the information flows. As the idea of entity from the information flow perspective is related to a type of entity, data delivery defines whether the information is delivered to a particular entity (an instance) or to the set of them belonging to the same type (a role).
- The data definition of an information flow describes the structure of the information that is being transported by the information flow.

The Domain Specific Language (DSL) is based on the Data Flow Diagrams DFD [Yourdon E., 2006, Stevens W. et al., 1974] and the UML Communication Diagram [Fowler M., 2003, OMG, 2005].

The information flow metamodel is depicted in Figure 8.

Finally, an example of the domain specific language for this metamodel is depicted on Figure 18 in Section 5.

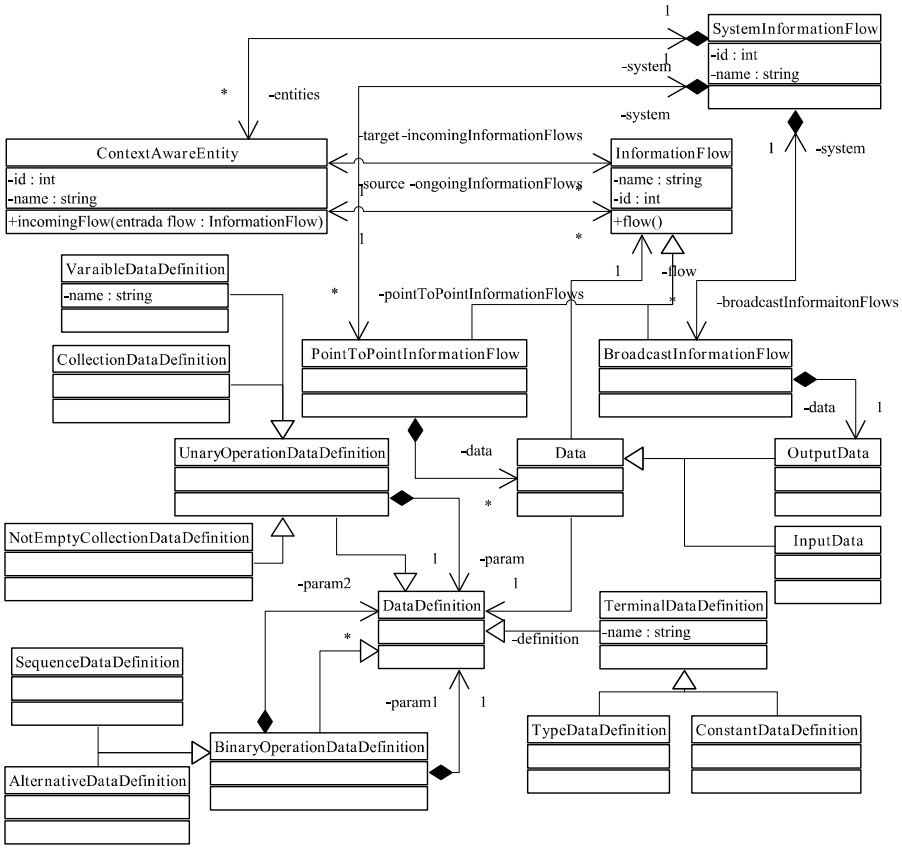

Figure 8: The information flow metamodel

4.2.2 The referential space model

The referential space model expresses referential dependencies between entities. Therefore, the goal of this view is the definition of the runtime environment of the entity context in CASs for UCEs.

Thus, the model describes the dependency between entities, i.e. a button depends on the panel it is contained.

The basic concepts behind this model are:

- The Entity Reference is a reference to a type of entity (defined by the information flow view). Thus, an entity type may be referenced several times in the referential space view.
- The Referential Space is used to define a runtime environment where entities share a referential space. It is also used to define the referential

dependencies of a set of entities. A work session is a typical example of referential space because a set of entities are allocated when the session starts and de-allocated when it finishes.

- The referential dependency defines a relationship between an entity reference and a referential space. When a referential space is de-allocated, all entity references related to it are de-allocated too.
- The reference cardinality is an attribute defined by a referential dependence. It defines the amount of entity references in the referential space. Thus, relationships may be defined as ONE-TO-ONE, ONE-TO-MANY and MANY-TO-MANY.

The referential space metamodel is depicted in Figure 9.

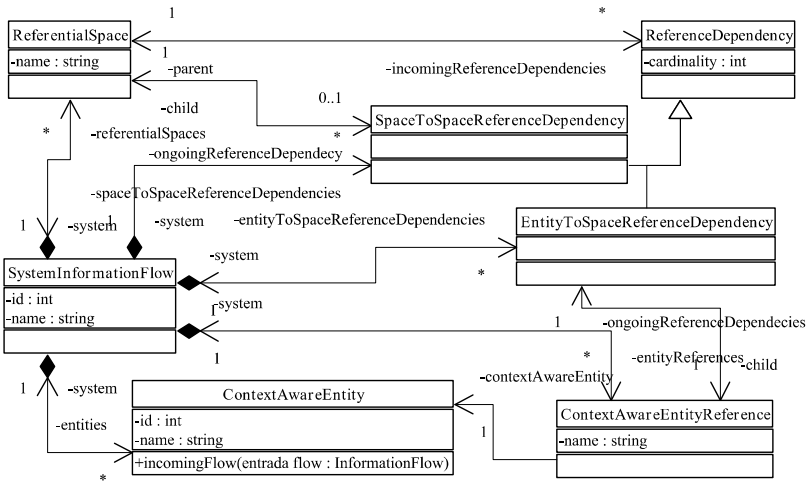

Figure 9: The referential space model

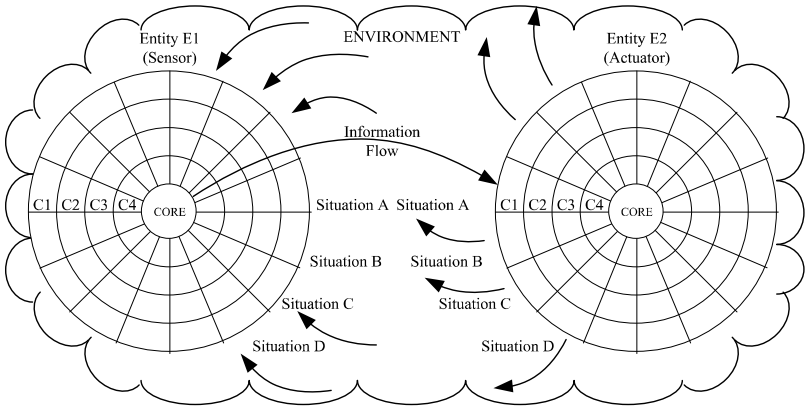

Figure 10: The information processing mechanism of the entity context model

Finally, the notation used to represent these concepts is closely related to UML Class diagrams [OMG, 2005].

An example of the domain specific language for this metamodel is depicted on Figure 17 in Section 5.

4.2.3 The entity context model

The entity context model defines the context representation from the entity core point of view. Thus, the goal of the entity context metamodel is the description of the context that an entity is able to perceive from the environment.

The main concepts addressed by this view of the system are:

- The independent context feature allows an independent definition of the entity context from the entity core domain. Thus, system and context functionalities are defined separately.
- The situation defines the set of relevant states of the entity that are relevant to the entity core and the entity context. Thus, the entity context is defined in terms of situations that are expressed as conditions bounded to the entity context state. There are five types of conditions according to the aspect of the context to be defined:
    1. The task conditions are defined in terms of regular expressions that describe task dependency in the system
    2. The social conditions are also defined in terms of regular expressions. They describe aspects of the entity, such as the role of the entity in the system, cooperation issues, etc.
    3. The space conditions are defined in terms of regular expressions too. They describe spatial conditions, such as the location of the entity, co-location (jointly with the social expression), etc.
    4. The data expressions define the information passing among entities.
    5. The logic expressions are usually coupled to data expressions because they allow the definition of first order logic conditions bounded to information defined by the data expressions.
- Information flows are processed by entities. Thus, entities are able to "sense" the information flows that are contextualized according to the entity context state.
- The data definition allows designers to define the structure of the information that is exchanged by information flows are defined in terms of regular expressions.

The Figure 10 depicts the conceptual mechanism of how the entity context works.

The system is composed by entities that may play the role of sensors, actuators or both of them. While entities that are able to "perceive" environmental changes are known as sensors (E1), entities that are able to "modify" the environment are known as actuators (E2).

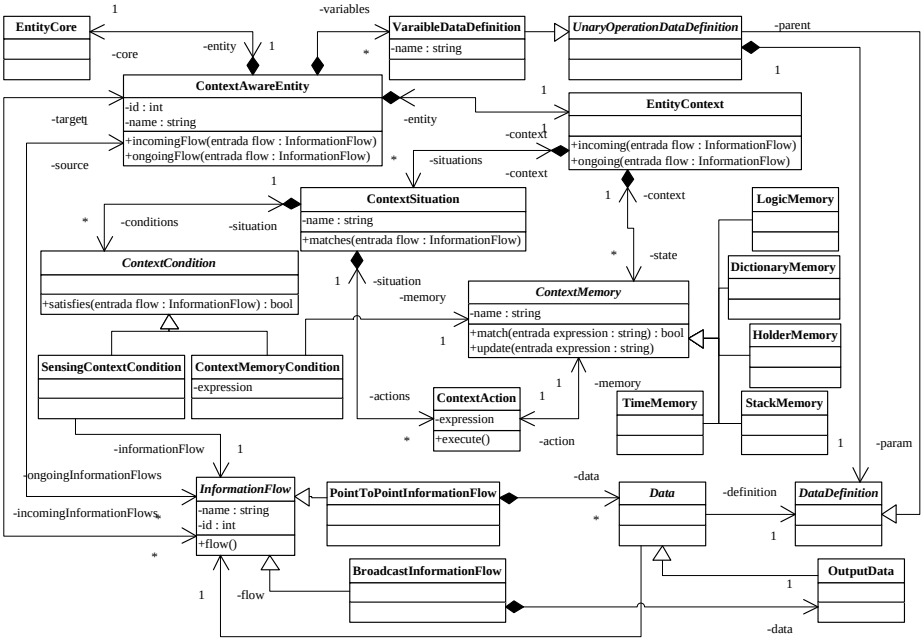

Figure 11: The entity context metamodel

Thus, sensors capture situations from the environment through a set of conditions (C1, C2, C3 and C4 defined by E1). When a situation arises, it modifies the entity context information and notifies the core. Eventually, the core may produce an information flow (Information Flow) that is consumed by other entities (i.e. E2).

In order to contextualize the Information Flow according to E2 context perception, a situation is defined through a set of conditions (C1, C2, C3 and C4 defined by E2). This situation is notified to the code of E2 acting on the environment consequently.

The entity context metamodel is depicted in Figure 11.

Finally, an example of the domain specific language for this metamodel is depicted on Figure 17 in Section 5.

4.3 The platform specific model

The last layer in the model is defined by the platform specific model. As the computational representation was defined in the platform independent model, this model specifies the deployment environment of the system based on the platform specification.

In order to define the information related to the platform and the deployment layout the mapping metamodel model was defined.

The main goal of the mapping metamodel is the definition of the deployment environment and the platform specification of the system. It defines two types of elements:

1. Mapping elements (devices and connections)
2. Definition elements (protocols, media, operation systems or programming languages).

Mapping elements assign physical representations to computational elements defined in the PIM. Entities are assigned to devices that provide them the runtime environment to perform their tasks. In addition, information flows are assigned to connections that provide them communication among devices.

In order to define platform issues, protocols and media are assigned to connections; and operating systems and programming languages are assigned to devices. Thus, both connections and devices can be turned into source code through a model-to-text transformation.

The deployment metamodel is depicted in Figure 12

Finally, an example of the domain specific language for this metamodel is depicted on Figure 18 in Section 5.

5 The case of study

To show the utility of the methodology, a set of CASE tools that support the creation and edition of models conforming to the metamodels defined in Section 4 was developed.

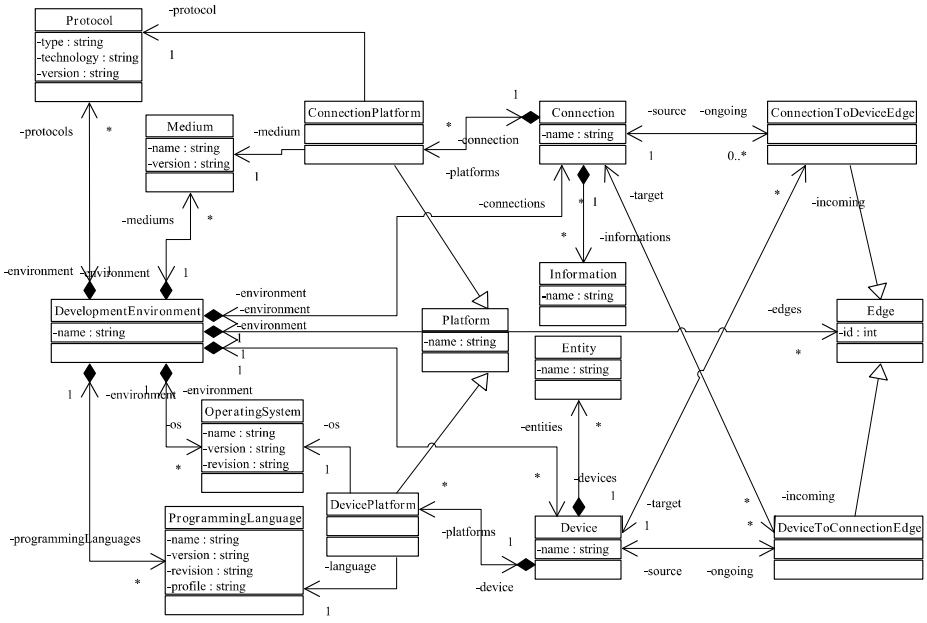

Figure 12: The deployment metamodel

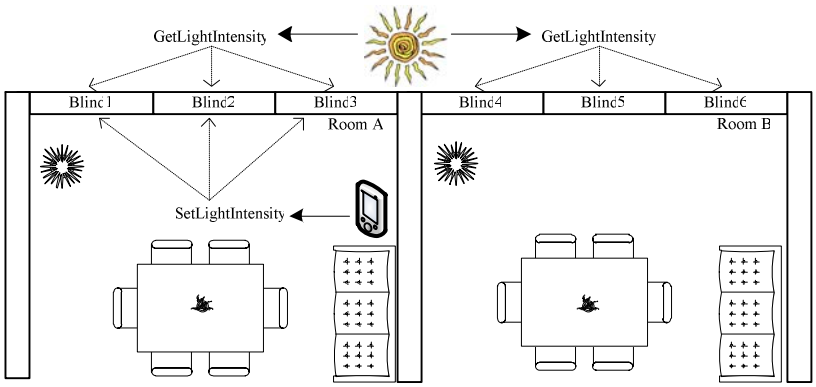

Figure 13: The light controller scenario

These tools were developed as Eclipse Plugins using the Eclipse Modelling Framework (EMF) and Graphical Modelling Framework (GMF).

The demonstration is based on the definition of all models regarding the "Light controller" system. This system is a domotic application that allows users to set the light intensity within a room using the axial movement of electronic blinds avoiding the use of artificial light (for energy saving reasons).

Figure 13 exposes the application scenario where we have two rooms and three blinds on each room.

The system is composed by three main actors: the PDA that provides users the ability to change the light intensity of the room he/she is in, the blinds that should be able to turn according to the light intensity the user has set and the actual light intensity of the room, and the light sensors (one for each room) that is in charge of perceiving the light within the room.

5.1 The computational independent model

The computation independent model is depicted in Figure 14, Figure 15 and Figure 16.

It is composed by three models:

- The social model is depicted in Figure 16. It describes the entities that are part of the system (PDAs, Blinds and LightSensors)
- The space model is depicted in Figure 15. It defines the space scenario with two rooms (RoomA and RoomB) which contains all the blinds (Blind1 to Blind6) and sensors (LightSensorA and LightSensorB). Although the SetButton is the only control that is explicitly modelled, the model includes the representation of the remaining PDA controls through the PDAControl virtual space.
- The task model is depicted in Figure 14. It defines two cooperative tasks:

1. The SetLightIntensity between the PDA and the group of blinds that is co-located with the PDA

```
SP: aPDA(release,  SetButton), aPDA(Action,
PDAControl)*, aPDA(enter, Room),
Blind(enter,Room)
```

2. The GetLightIntensity between the blinds and the light sensor that is co-located in the room.

```
SP: (aLightSensor(enter, Room), Blind(enter,
Room)) + (Blind(enter, Room),
aLightSensor(enter, Room))
```

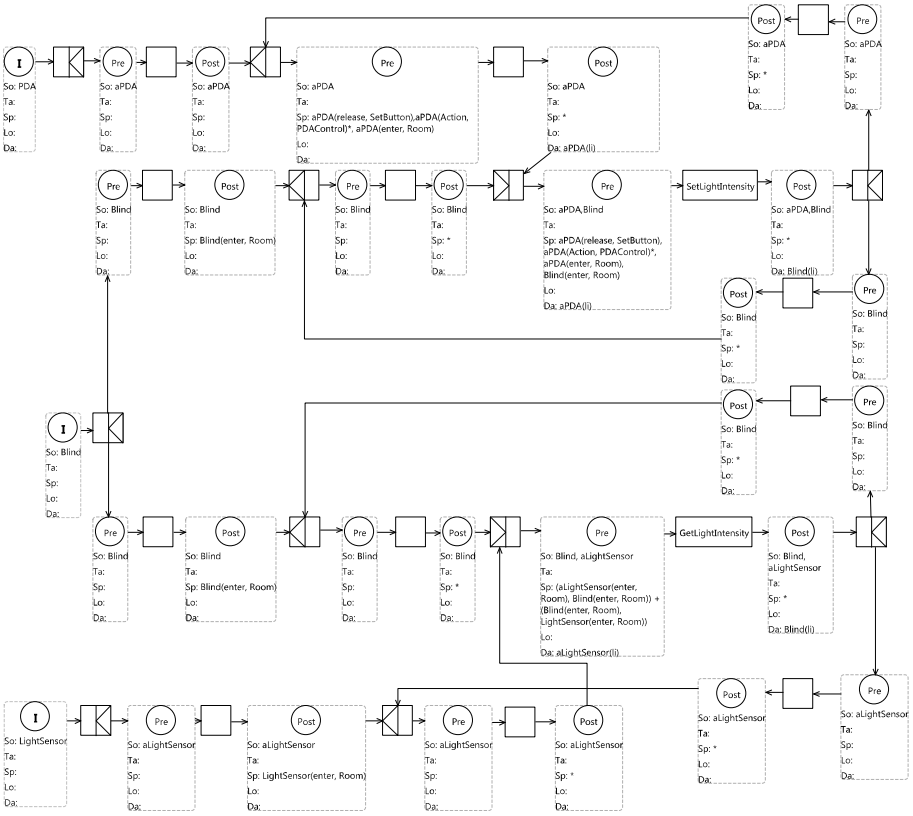

Figure 14: The light controller task model

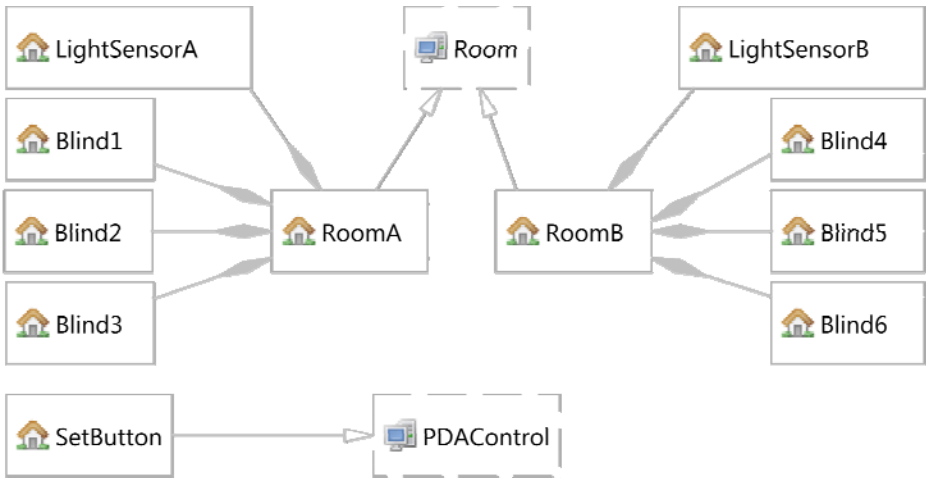

Figure 15: The light controller space model

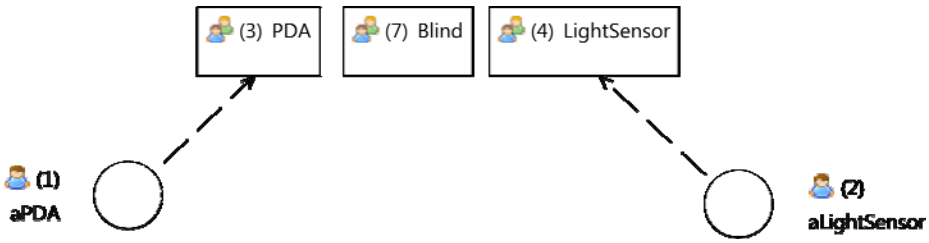

Figure 16: The light controller social model

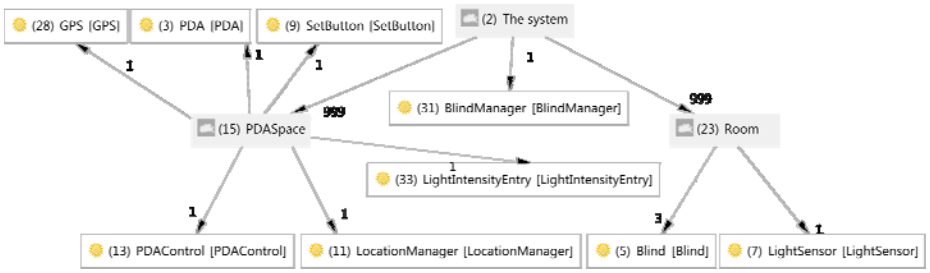

Figure 17: The light controller referential space model

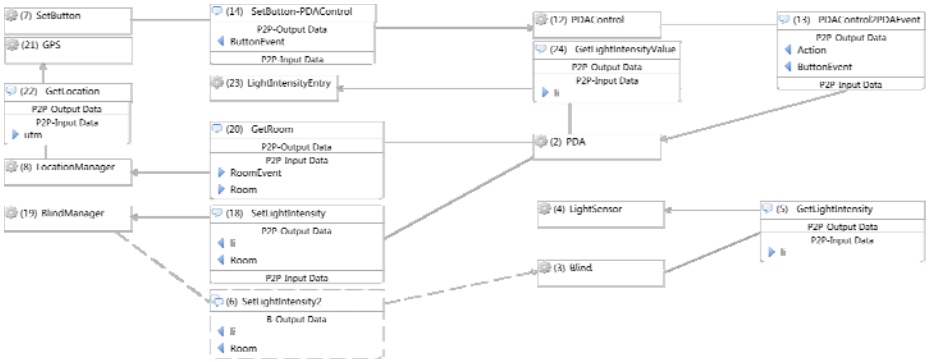

Figure 18: The light controller information flow model

5.2 The platform independent model

The platform independent model is depicted in Figure 17, Figure 18 and Figure 19. It is composed by three models:

- The referential space model is depicted in Figure 17. It describes the inter-entity relationships in terms of referential spaces. Thus, the system is composed by three spaces or applications: the PDA (PDASpace), the BlindManager and the Room. These applications are composed by smaller interrelated components. The relationship among these components is described by the information flow model.
- The information flow model is depicted in Figure 18. It describes the inter-entity relationships in terms of information flows. The information flow can be seen from:
    1. The PDA perspective; it has to notify the blinds the illumination level for a room. To perform this task, it receives the information from three entities:
        - The SetButton is used by the user to set the illumination level in the room.
        - The LightIntensityEntity gets the illumination level value from the user interface of the PDA.
        - The LocationManager returns the PDA location from the GPS entity.

       Moreover, it sends this information to the blind manager in order to update the desired illumination level for the room.
    2. The Blinds perspective; they have to check the desired illumination level for the room and adjust themselves automatically to match the user request. To perform this task, they receive the actual illumination level from the LightSensors and the desired illumination level from the BlindManager (via broadcasting). Thus, they have all the information to calculate their position.

- The entity context model is depicted in Figure 19. It describes the intra-entity structure. The model exposes the context logic of all entities defined in the system that supports the interaction between the environment and the core model of the application.

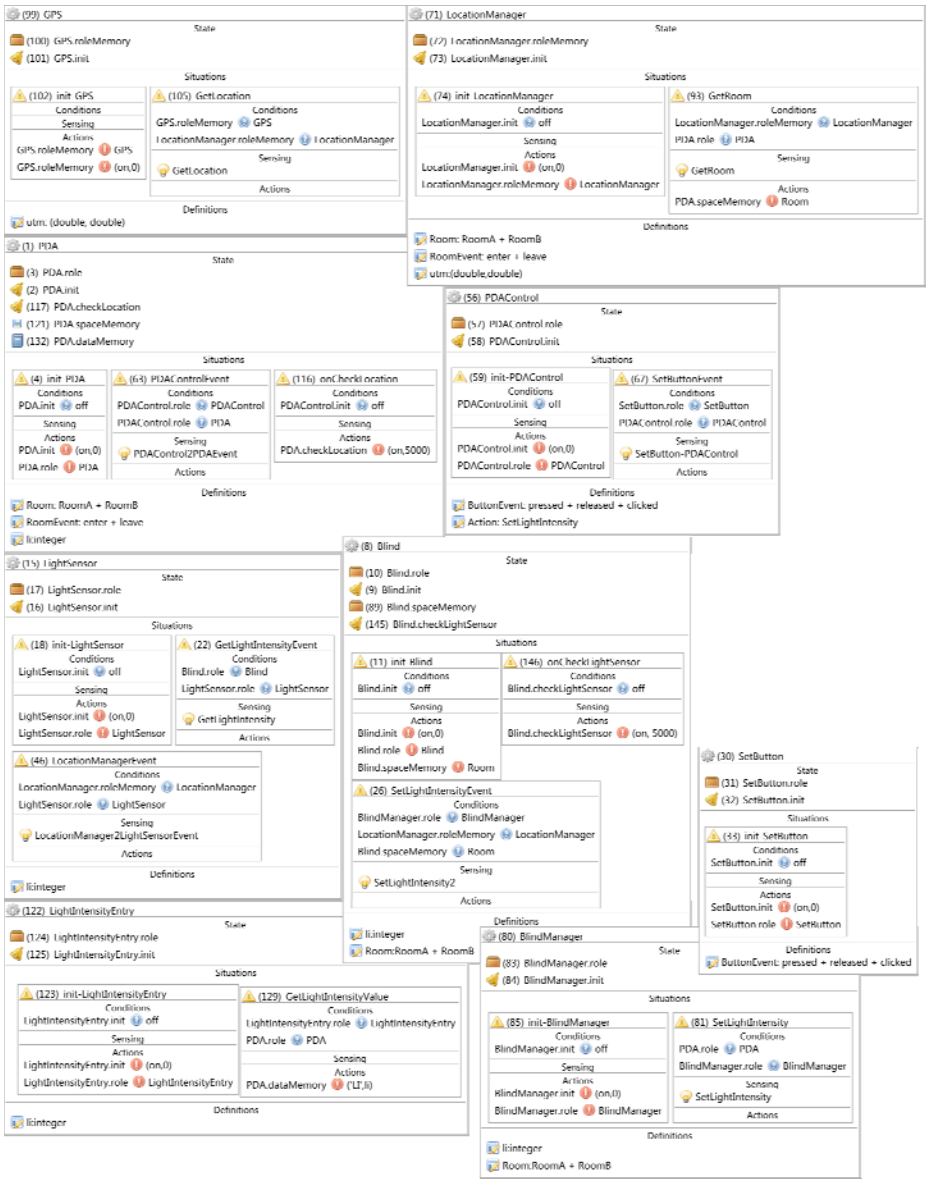

Figure 19: The light controller entity context model

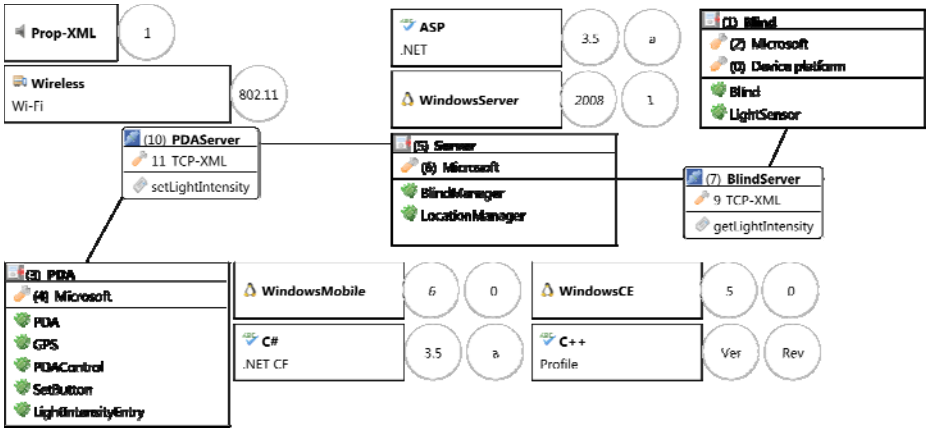

Figure 20: The light controller deployment model

5.3 The platform specific model

The platform specific model defined for this application is based on a Microsoft platform. The Figure 20 shows the characteristics of the system.

The deployment model embraces three devices:

- The PDA defines a runtime container for the entities that are related to the mobile device (PDA, GPS, PDAControl, SetControl, LightIntensityEntity).
- The Server defines a runtime container for the blind and location managers. They are defined as service providers.
- The Blind defines the runtime container for the blinds and the light sensors.

The connections (PDAServer and BlindServer) among devices use 802.11 as communication protocol and TCP/IP transport and address protocol.

6 Conclusions and future work

This work exposes a methodology to develop CAAs for UCEs based on a MDA.

The architecture is divided into three layers of models: CIM, PIM and PSM. The CIM model defines three types of models conforming the social, task and space metamodels. These metamodels provides designers with the ability to represent CAA features defined by [Schmidt A. et al., 1999b].

Once the CAS is characterized, a computational model is defined in terms of the information flow, the referential space and the entity context metamodels.

Models conforming to these metamodels are part of a PIM and most of the information they represent is derived from the CIM through a multi-model transformation defined in ATL.

Thus, the software representation of the system can be customized by the modification of these models. After defining the PIM, the mapping metamodel defines the deployment platform of the system by creating the PSM of the application.

A model-to-text transformation gathers the PIM jointly with the mapping model and turns them into source code. This code is part of an abstract architecture that is supported by the platform.

Due to space restrictions, the article describes just a small example of the expressive power of the methodology to define CAA characteristics. However, on [Tesoriero R., 2009] there are three full examples where all the characteristics defined by [Schmidt A. et al., 1999b] are exposed.

Therefore, this methodology addresses the development of CAAs from different points of view defining different layers of metamodels according to the abstraction level the developer of the application has to address.

As future work, we are currently developing the adaptation between the abstract architecture and a set of production frameworks.

We are also working on the generation of verification code from the post conditions of the simple tasks. Besides, although the consistency of models is checked using OCL restrictions, a coherence mechanism to make a set of models consistent through OCL restrictions is under construction.

Finally, we are defining a simulator program based on Petri Nets properties in order to check the liveness and other characteristics of the system.

Acknowledgements

This project has been partially supported by the Spanish Science and Innovation Ministry TIN 2008-06596-C02-01 CENIT MIO! Project (CENIT-2008 1019).

References

[Adams N. et al., 1993] Adams, N., Gold, R., Schilit, B., Tso, M. and Want, R.: "The ParcTab Mobile Computing System"; Proc. 4th Workshop on Workstation Operating Systems (WWOS-IV), IEEE, Napa U.S., 1993, 34-39.

[Almeida J. P. A. et al., 2006] Almeida, J. P. A., Iacob, M.-E., Jonkers, H. and Quartel, D.: "Model-Driven Development of Context-Aware Services"; Proc. Distributed Applications and Interoperable Systems, Lect. Notes Comp. Sci. 2025, Springer, 2006, 213-227.

[Brown P. J. et al., 1997] Brown, P. J., Bovey, J. D. and Chen, X.: "Context-aware Applications: from the Laboratory to the Marketplace"; IEEE Pers. Comm. 5 (4), October 1997, 58-64, http://www.cs.kent.ac.uk/pubs/1997/395.

[Bzivin J., Jouault, F. and Valduriez, P., 2008] Bzivin, J., Jouault, F. and Valduriez, P.: "The Atlas Transformation Language ATL", 2008, http://ralyx.inria.fr/2003/Raweb/atlas/uid10.html.

[Carton A. et al., 2007] Carton, A., Clarke, S., Senart, A. and Cahill, V.: "Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing"; Proc. 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments IEEE Computer Society, Washington DC, USA, 2007, 5.

[Chen H., 2003] Chen, H.: "An Intelligent Broker Architecture for Context-Aware Systems"; PhD thesis, University of Maryland Baltimore County.

[Chen H. et al., 2003a] Chen, H., Finin, T. and Joshi, A.: "An ontology for context-aware pervasive computing environments"; Knowl. Eng. Rev.,Cambridge University Press, 18(3), 2003, 197-207.

[Chen H. et al., 2003b] Chen, H., Finin, T. W. and Joshi, A.: "Using OWL in a Pervasive Computing Broker"; Proc. the Workshop on Ontologies in Agent Systems at the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems, Cranefield, S., Finin, T. W., Tamma, V. A. M. and Willmott, S. (eds) , 73, Melbourne, Australia, July 2003, 9-16.

[de Farias C. R. G. et al., 2007] de Farias, C. R. G., Leite, M. M., Calvi, C. Z., Pessoa, R. M. and Filho, J. G. P.: "A MOF metamodel for the development of context-aware mobile applications"; Proc. ACM Symposium on Applied Computing, ACM, New York USA, 2007, 947-952.

[Desel J. and Juhàs G., 2001] Desel, J. and Juhàs, G. (eds.): "What is a Petri net? Informal answers for the informed reader"; Unifying Petri Nets. Lect. Notes Comp. Sci. 2128, Springer, 2001, 1-27.

[Dey A. K. et al., 2001] Dey, A., Abowd, D., G. and Salber, D.: "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications"; Human-Computer Interaction, 16 (2/4), 2001, 97-166.

[Dey A. K., 2001] Dey, A. : "Understanding and Using Context"; Pers. Ubiq. Comp., 5, 2001, 4-7.

[Dey A. K. and Abowd G. D., 2000] Dey, A. K. and Abowd, G. D.: "Towards a Better Understanding of Context and Context-Awareness"; Proc. Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems, Thea Turner and Gerd Szwillus and Mary Czerwinski and Fabio Peterno and Steven Pemberton (eds.) Georgia Institute of Technology, The Hague, The Netherlands, April 2000.

[Eclipse Foundation, 2009] The Eclipse Foundation: "MOFScript". http://www.eclipse.org/gmt/mofscript/

[Elrad T. et al., 2001] Elrad, T., Filman, R. E. and Bader, A.: "Aspect-Oriented Programming"; Comm. ACM, 44 (10), October 2001, 29-32.

[Fowler M., 2003] Fowler, M.: "UML Distilled: A Brief Guide to the Standard Modeling Object Language"; Object Technology Series, 3rd edition, Addison-Wesley, 2003.

[Georgalas N. et al., 2007] Georgalas, N., Ou, S., Azmoodeh, M. and Yang, K.: "Towards a Model-Driven Approach for Ontology-Based Context-Aware Application Development: A Case Study"; Proc. 4th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, IEEE Computer Society, Washington DC, USA, 2007, 21-32.

[Kozaki K. et al., 2005] Kozaki, K., Kitamura, Y. and Mizoguchi, R.: "Developing Ontology-based Applications using Hozo"; Proc. 4th International Conference on Computational Intelligence, Calgary Canada, 2005, 273-277.

[Kumar S. et al., 2000] Kumar, S., Cohen, P. R. and Levesque, H. J.: "The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams"; Proc. 4th International Conference on Multi-Agent Systems, IEEE Computer Society, 2000, 159–166.

[OMG, 2004] The Object Management Group: "Meta Object Facility (MOF) 2.0 Core Final Adopted Specification"; (Mar. 2004) http://www.omg.org/cgi-bin/doc?ptc/03-10-04.

[OMG, 2005] The Object Management Group: "UML Version 2.0", July 2005, http://www.omg.org/spec/UML/2.0/

[OMG, 2006] The Object Management Group: "Object Constraint Language (OCL) Specification, Version 2.0", 2006, http://www.omg.org/technology/documents/spec_catalog.htm.

[Ou S. et al., 2006] Ou, S., Georgalas, N., Azmoodeh, M., Yang, K. and Sun, X.: "A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development"; Proc. 2nd European Conference on Model Driven Architecture - Foundations and Applications, Rensink, A. and Warmer, J. (eds), Lect. Notes Comp. Sci. 4066, Springer, Bilbao Spain, July 2006, 188–197.

[Penichet V. M. R., 2007] Penichet, V. M. R.: "TOUCHE (Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments)"; PhD thesis, University of Castilla-La Mancha, Spain, 2007.

[Peterson J. L., 1981] Peterson, J. L.: "Petri Net Theory and the Modeling of Systems"; Prentice–Hall Inc., 1981.

[Petri C. A., 1962] Petri, C. A.:"Kommunikation mit Automaten"; PhD thesis, University of Bonn, Bonn, Germany.1962 (In German)

[Prezerakos G. N. et al., 2007] Prezerakos, G. N., Tselikas, N. D. and Cortese, G.: "Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects"; Proc. International Conference on Web Services, IEEE Computer Society, Utah, USA, July 2007, 320-329.

[Schilit B. et al., 1994] Schilit, B., Adams, N. and Want, R.: "Context-Aware Computing Applications"; Proc. Workshop on Mobile Computing Systems and Applications, IEEE, Santa Cruz, US, 1994.

[Schilit B. and Theimer M., 1994] Schilit, B. and Theimer, M.: "Disseminating Active Map Information to Mobile Hosts"; IEEE Network, 8(5), 1994, 22-32.

[Schmidt A., 2002] Schmidt, A.: "Ubiquitous Computing - Computing in Context"; PhD thesis, Lancaster University, UK, 2002.

[Schmidt A. et al., 1999a] Schmidt, A., Adoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. and Van De Velde, W.: "Advanced Interaction in Context"; Proc. 1st International Symposium on Handheld and Ubiquitous Computing, Springer Verlag, 1999, 89-101.

[Schmidt A. et al., 1999b] Schmidt, A., Beigl, M. and Gellersen, H.-W.: "There is more to context than location"; Computers & Graphics, 23(6), 1999, 893-901.

[Stevens W. et al., 1974] Stevens, W., Myers, G. and Constantine, L.: "Structured Design"; IBM Systems Journal, 13, 1974, 115-139.

[Sunagawa E. et al., 2006] Sunagawa, E., Kozaki, K., Kitamura, Y. and Mizoguchi, R.: "Role Organization Model in Hozo"; Proc. of 15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes Art. Int., 4248, Podebrady, Czech Republic, October 2006, 67–8.

[Tesoriero R., 2009] Tesoriero, R.: "CAUCE: Model-driven Development of context-aware applications for ubiquitous computing environments" PhD thesis, University of Castilla-La Mancha, Spain, December 2009.

[Vale S. and Hammoudi S., 2008] Vale, S. and Hammoudi, S.: "Towards context independence in distributed context-aware applications by the model driven approach"; Proc. 3rd InternationalWorkshop on Services integration in pervasive environments, ACM, New York USA, 2008, 31-36.

[Wang X. H. et al., 2004] Wang, X. H., Gu, T., Zhang, D. Q. and Pung, H. K.: "Ontology Based Context Modeling and Reasoning using OWL"; Proc. International Conference on Pervasive Computing and Communication, IEEE, 2004, 18-22.

[Want R. et al., 1992] Want, R., Hopper, A., Falcao, V. and Gibbons, J.: "The Active Badge Location System"; Tran. on Inf. Syst., ACM, 10 (1), January. 1992, 91-102.

[Want R. et al., 1995] Want, R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R. and Weiser, M.: "An Overview of the ParcTab Ubiquitous Computing Experiment"; Pers. Comm., IEEE, 2, (6), December 1995, 28-43.

[Weiser M. and Brown J. S., 1997] Weiser, M. and Brown, J. S.: "The Coming Age of Calm Technology"; Beyond Calculation: The Next Fifty Years of Computing, Denning, P. J. and Metcalfe, R. M. (eds.), Copernicus, 1997, 75-85.

[WfMC, 1996] The Workflow Management Coalition: "Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011)"; Brussels, 1996.

[Yourdon E., 2006] Yourdon, E.: "Just Enough Structured Analysis"; Yourdon Press, 2006.

[Zimmer T., 2004] Zimmer, T.: "Towards a Better Understanding of Context Attributes"; Proc. PerCom Workshops, IEEE Computer Society, 2004, 23-27.