Journal of Universal Computer Science, vol. 14, no. 9 (2008), 1545-1559
submitted: 30/4/08, accepted: 19/2/08, appeared: 1/5/08 © J.UCS

Development of Ambient Intelligence Systems Based on Collaborative Task Models

Roberto F. Arroyo
(Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada, Spain
robfram@ugr.es)

Miguel Gea
(Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada, Spain
mgea@ugr.es)

José Luis Garrido
(Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada, Spain
jgarrido@ugr.es)

Pablo A. Haya
(Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain
Pablo.Haya@uam.es)

Abstract: So far, the Ambient Intelligence (AmI) paradigm has been applied to the development of a great variety of real systems. They use advanced technologies such as ubiquitous computing, natural interaction and active spaces, which become part of social environments. In the design of AmI systems, the inherent collaboration among users (with the purpose of achieving common goals) is usually represented and treated in an ad-hoc manner. However, the development of this kind of systems can take advantage of rich design models which embrace concepts in the domain of collaborative systems in order to provide the adequate support for explicit or implicit collaboration. Thereby, relevant requirements to be satisfied, such as an effective coordination of human activities by means of task scheduling, demand to dynamically manage and provide group- and context-awareness information. This paper addresses the integration of both proactive and collaborative aspects into a unique design model for the development of AmI systems; in particular, the proposal has been applied to a learning system. Furthermore, the implementation of this system is based on a blackboard-based architecture, which provides a well-defined high-level interface to the physical layer.

Keywords: collaborative model, ambient intelligence, ubiquitous computing, task modeling, context awareness
Categories: D.2.1, D.2.11, H.1.2, H.5.3

1 Introduction

Ambient intelligence (AmI) represents the next step in the user-centred approach of computer applications. They incorporate technology into an omnipresent and transparent infrastructure for the implementation of smart environments. AmI paradigm highlights on user-friendliness, more efficient services and support for human and group interaction [Campbell 03]. This paradigm is based on emerging technologies, such as ubiquitous computing, collaborative systems and intelligent user interfaces for natural interaction [Riva 05]. Up to date, although interest in this technology and its benefits are high, it is difficult to develop this kind of systems fulfilling all elicited requirements.

Task modeling [Paterno 99] is a useful technique to describe interactive and collaborative systems translating user activities and their required data into structured knowledge fragments, the so-called tasks. In general, a task is portrayed on the basis of a set of actions to be performed, namely of information required for these actions, and of the actors who perform them. However, no context information is normally taken into account in task modeling, or at least, it is not represented in an appropriate way.

Collaboration means a tacit implication to the achievement of a common goal [Aldunate, 02]. In this way, according to the Activity Theory (AT) [Nardi 95], a collaborative task is accomplished by different actors (playing roles) and coordinates them to reach a precise objective. Although several approaches focus on collaborative task modeling [Paterno 03; Veer 00], its integration in AmI systems taking advantage of more complete domain models results in new challenges in research.

When collaborative tasks are modeled in AmI systems, two kinds of actuation manners in participants can be observed: a) an explicit way if these participants are conscious of the intervention of the environment in the activity [Card 00]; b) an implicit way when they delegate portions of work to the system, which acts in transparently solving activities from context information. The environment is a participant itself, being able, if needed, to behave actively. The last way has very interesting connotations from the AmI point of view, since most of the system decisions can be taken using the context information. In [Dey 00], context is defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

This paper aims to address the modeling and design of the collaborative and proactive aspects of intelligent environments, where contextual information is required to manage situations successfully. To this objective, the use of rich models can provide benefits in order to pay special attention to relevant parts of the system.

The starting point is AMENITIES [Garrido 05], which provides a conceptual and methodological framework on the basis of task models and group behavior and has been specially devised for the study and development of cooperative systems. The concepts that are present in collaborative systems are also present in AmI systems, namely, tasks, groups, roles, artefacts, shared resources, etc. Furthermore, the methodology proposes the creation of a system model, which is independent of its implementation. A software platform based on the blackboard architecture has been built in order to support the development of these systems [Haya 04]; the context layer is implemented by using a blackboard model-based middleware that maintains a global data structure for relevant information about the world model. Blackboard architectures facilitate the dynamic configuration of the system components. This is achieved by means of an anonymous communication mechanism that enables loosely-coupled interactions among different elements of the architecture. Accordingly, every time a new component appears, the rest remain unchanged, rather than resetting up to discover the location of the new component. Thereby, the blackboard paradigm is very suitable for AmI systems, in which the composition of the environment changes very frequently.

The paper is organized as follows. Section 2 briefly introduces the case study used to describe this research work in detail. In Section 3 we present the conceptual framework in which the proposal is based on. Section 4 proposes a design model specially devised for the development of AmI systems. We describe how to integrate the conceptual design to the main AmI mainstream in Section 5. The implementation of this design by means of the blackboard architecture will be illustrated in Section 6. Finally, conclusions and future works are exposed in Section 7.

2 Learning AmI Systems: Case Study

The work shown in this paper is part of the framework of U-CAT (Ubiquitous Collaborative Adaptive training) project, a Spanish research project for the definition and development of smart learning environments. The aim of this project is to adapt the proposed teaching activities, the tools to be used and the contents presented to the users, by considering the context of each element involved in the activities, together with the information about the physical devices that can be used to perform the activity. This provides an interesting social community model with different actors playing roles (students and teachers) and collaborating, acquiring knowledge and skills when they participate in classes, do exercises in group and share resources (slides, documents, etc).

AmI systems can improve the traditional model of teaching. For example, students can participate from a different physical location where the lesson is lectured, using devices like PDAs or computers, and sharing the same context. Similarly, the system is aware of several facts, such as the arrival of teachers in the classroom, the presence and number of students in it, etc. [William, 04 and Nylander, 03] show some properties obtained with ubiquitous e-learning systems, which share properties with the AmI environments.

In this case study, we focus on the analysis of giving a class, taking into account participants, events to be carried out and other relevant concepts, as well as AmI system features (proactiveness, ubiquity, etc). In order to teach a lesson, first this task must be scheduled first, and then carried out. Hence, we have two related tasks linked up by an order relationship: the teacher must schedule an appointment with his/her students (date and place of the lesson). The teacher who gives the lesson and the students attending the lesson must appear in person at that place and date agreed.

Usually, the scheduling is done in an asynchronous way, and the arrival of the teacher to the classroom represents the beginning of the lesson. Both tasks need the participation of the students, the teacher and the system. Some students can attend the lesson in a virtual manner, thus not being physically in the classroom but using a device which let them send and receive comments and contributions. We can also think about the possibility that the student does not want to participate voluntarily in the lesson because he/she is busy (i.e. with another lesson and he/she does not want to be interrupted).

Scheduling is a collaborative task among teachers, students and environment.

This task uses context information such as the student location, previously stated academic timetable of teachers and students, their availability, their preferences, etc.

The decision may be conditional to certain policies which determine the type of lesson (mandatory, recovery, tutorship, etc.) and the specific setting according to the students who are participating (the whole class, some groups, an individual), and different coordination strategies (an hour established by the official timetable, by consensus, by voting, by available timetable holes, etc.) As we can see in the case study above considered, a first challenge arises from the need to provide a feasible orchestration procedure for collaborative task scheduling.

Moreover, this procedure should overcome the mapping between available physical resources and tasks to be done. To address these problems, we propose as starting point the use of the conceptual framework defined in AMENITIES, which is introduced in the following section.

3 Conceptual Model

AMENITIES, a methodological approach developed in the Granada University, is based on tasks and behavior models, with tasks being the main concept of any system modeled using this methodology. [Fig. 1a] shows, using an UML class diagram, the concepts and relationships included in the AMENITIES conceptual framework. According to this framework, an action is an atomic unit of work. Its event-driven execution may require/modify/generate explicit information. A subactivity is a set of related subactivities and/or actions. A task is a set of subactivities intended to achieve certain goals. A role is a designator for a set of related tasks to be carried out. An actor is a user, program, or entity with certain acquired capabilities (skills, category, etc.) that can play a role in the execution (using artefacts) of (or responsibility for) actions. A group performs certain subactivities depending on interaction protocols. A cooperative task is one that must be carried out by more than one actor, playing either the same or different roles. A group is a set of actors playing roles and organized around one or more cooperative tasks. A group may comprise (i.e. be formed of) related subgroups. A law is a limitation or constraint imposed by the system that allows it to adjust the set of possible behaviors dynamically. An event is based on its common software definition, which is an occurrence or happening of significance to a task or program. An organization consists of a set of related roles. Finally, a cooperative system is composed of organizations, groups, laws, events and artefacts.

In Fig. 1b the U-CAT physical laboratory of the Autonomous University of Madrid is shown. This laboratory has got several devices such as cameras, TV, radios, lights or switches. The AMENITIES proposal provides a conceptual framework for defining task-based models, and the laboratory provides some of the AmI features such as location or physical interaction. Fig. 1c shows our proposal of a design to model an AmI system with the benefits of tasks, without forgetting the context information such as location, and their connections to real entities.

Fig. 2 shows the specification of the roles in AMENITIES involved in the task of scheduling a lesson. This collaborative task ScheduleClass must be carried out by an actor playing the role of Teacher, none or more actors (specified as 0..n) playing the role of Student, and another one actor playing the role of System. Moreover, this task is defined according to the ordering of the following subactivities:

- Identifying teacher preferences (GetDatesAndResources); the teacher decides which resources he/she will need for the lesson, and the dates for it.
- Obtaining the list of possible configuration of the resources and dates (ProvideDateAndResources); the scheduler gets the dates in which the required resources are available and the state of the required resources for the specified dates.
- Selecting the candidate dates (ChooseCandidateDates); the teacher chooses a suitable set of dates as possible solutions to reach the goal: "find a date to give the lesson".
- Choosing the kind of the class (KindClass); the teacher decides the best manner to adjust the date and time of the lesson. He/she may choose one, delegate it to the scheduler, or even ask the students their preferences (by voting).
- Depending on the previous decision, the following options are possible alternatives:

1. To determine the best date (DecideDate), on which the scheduler chooses the dates (according to the classroom availability) best fitting the teacher preferences..
2. To vote the best date (VoteDates), where every student votes the date according to their personal preferences.

- To add the lesson and notify it (NotifyClass); the scheduler annotates the new appointment, and notifies it to the related people. Additionally, to maintain the consistency, it updates the availability of the used resources for the selected date.

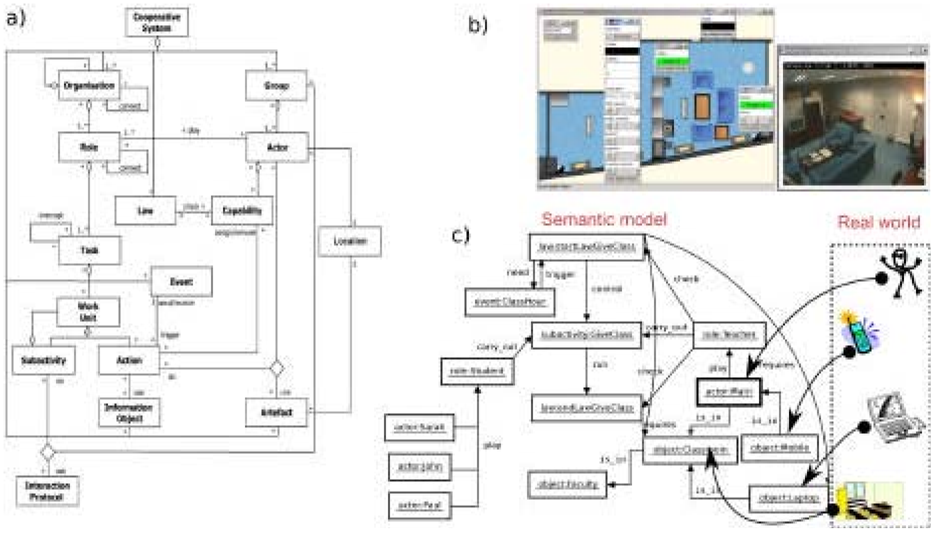
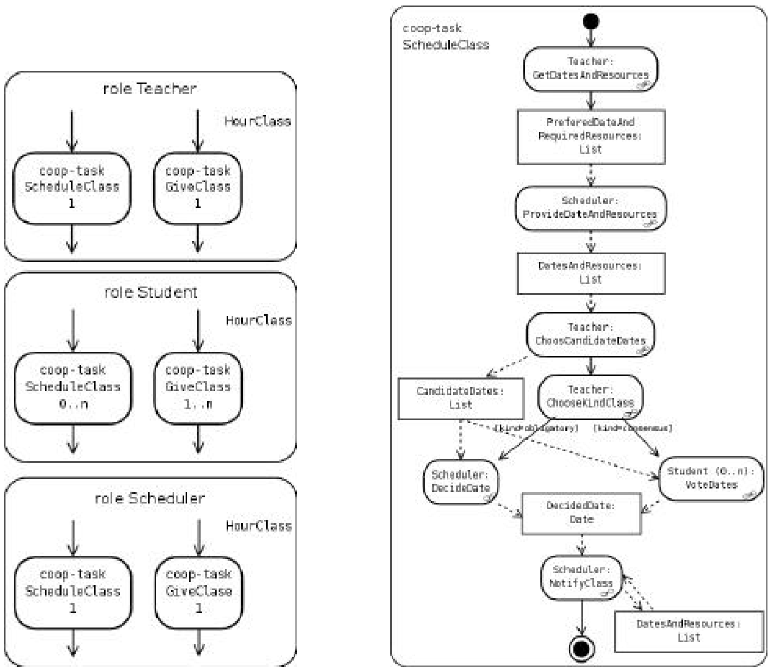


Figure 1: a) AMENITIES, main concepts for task modeling; b) The U-CAT laboratory: Web interface and on-line camera screen; c) Semantic model entities corresponding to physical entities

This model describes the complete system from a technology-independent point of view, it doesn't include any information about how the computer-based system will be implemented. However, it describes main functional requirements to be satisfied, which provide key information about, for instance, what tasks/subactivities can be assigned to the system (e.g., those assigned to the role Scheduler), or even the need of providing some type of awareness mechanisms in order to accomplish certain tasks/subactivities adequately (e.g., the support to notify the class agreed to be defined in the subactivity NotifyClass). The following section just describes the design proposal for the development of the computer-based system in charge of supporting these requirements (i.e. the AmI System).



(a) (/b)

Figure 2: (a) Teacher and Student roles and (b) Cooperative task ScheduleClass modeled in AMENITIES conceptual framework

4 Design Model

Our approach is based on a model-driven architecture [Mellor 04], where these abstract concepts are refined and mapped onto other computational models.

Therefore, abstract concepts and relationships will be mapped on components with certain internal structure, encapsulating an expected behavior (interface) hiding, in some cases, the physical device. Relevant components are entities (roles, actors, objects), related set of actions (tasks), relevant facts (events) and rules governing the system (preconditions, laws). These components are helpful to understand and represent the world, and further, to detect the underlying context of such activities to predict events and anticipate the intelligent assistance. Associations between components (links) could be tagged to add additional information about the nature of the relation. Afterwards, we describe the mechanism to translate conceptual statements to this compact notation, considering relations and cardinality. The advantages of such an approach are the management of concepts of different nature in a straightforward and homogeneous way. Thus, these concepts are enough to describe collaborative task on intelligent environments.

4.1 Law

Law is defined in AMENITIES as a *constraint that determines the system behavior*.

A law has the following properties: self-information to identify the law itself; *preconditions*, comprising a set of conditions that must be satisfied in order for the law to be fulfilled; *actions* that are performed once the law has been fulfilled; and finally, a logical expression connecting previously defined preconditions by means of logical operators and possible events (with or without parameters) producing changes in the system activity. If this *logical expression* has not been specified, then the AND operator among preconditions is assumed by default. A scheme of a law definition is shown in Fig. 3. In order to create the logical expressions, we need a set of operations. For example, a law can specify that a lesson cannot start if the teacher is absent.

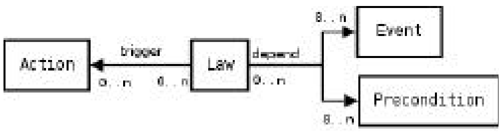


Figure 3: Law definition

4.2 Preconditions

A precondition is another logical component containing a set of restrictions to be applied to laws. Therefore, a set of logically interrelated restrictions is formed.

This comprises *self-information*, a set of *restrictions* specifying a list of particular attributes that a task should guarantee in order to carry out the system activities. These restrictions consist of elements with the attribute to be evaluated, a condition to be satisfied for this attribute, and a field indicating the obligatory nature of the restriction; a *logical expression* on the defined restrictions or an implicit *logic* AND (if the logical expression has been omitted). The compulsory nature can be used as a preference criterion for the candidate object choice. For example, when we establish the preference of one classroom instead of another, due to capabilities or equipment, we are speaking about preconditions.

The restriction set represents the attribute list that any goal object must achieve to fulfil the precondition. It is composed of the attributes to evaluate, the condition that each attribute must fulfil, and a field containing the obligatory nature of that restriction. Additionally to the pair <*attribute, value*>, a field regarding the type of restriction is defined. This field can contain *requisite* (0) or *preference* (1).

4.3 Tasks/Subactivities

A task is a subactivity in a higher level of abstraction with an intended goal.

Consequently, we will refer to elaborate groups of actions (or simpler ones) with the term subactivity. *Subactivity* is defined as the combination of actions performed by active or passive entities.

Analysing the needs of a subactivity, we characterize its components as a set of *roles*, determining which ones take part in this subactivity; a set of *outcoming events*, determining what events are generated; a set of *incoming events*, determining the events required; a set of *actions*, specifying what actions are done by the subactivity; and a set of *subactivities*, determining what subactivities are called by this one. A student carries out work when he/she gets involved in subactivies.

We must note that our purpose is to define a subactivity in a way that the system will be able to determine, using stored information, which tasks can depend on the subactivity, or which one is necessary to carry out the subactivity. Fig. 4 shows an example of it.

To facilitate the dependency identification process, the event representation will be in an unique event entity, related to two different ways (generate or receive) to determine which entities generate the necessary event enabling a specific subactivity to continue.

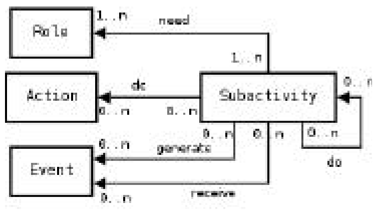


Figure 4: Relationship scheme about the objects which can participate in a subactivity

4.4 Events

An *event* is defined as an *occurrence or incident of significance for a task or program*. The information about its concrete use is stored in the relationships (link).

This specification consists of the *type* of relation with the event, i.e. whether the event is being sent or received; of a *roles* expression including at least one role or a composition of some of them which uses an exclusive OR operator, or the reserved word *any* followed by a list of roles to be excluded; and of a list of *parameters* that might be necessary for certain events. For instance, when a teacher leaves the classroom, an event is generated.

4.5 Roles

A role is a set of interruptible subactivities that can be assigned to an actor with both event-triggered and law-controlled executions. Therefore a role is an object composed by interruptible subactivites with both event-triggered and law-controlled execution, a role is formed by the interruptible specification defining under what circumstances a specific *subactivity* is interruptible; a *task list* composed by a *starting law*, a subactivity and an *ending law*. Fig. 5 shows a graphic representation of the relation between role object, laws and subactivities. Teachers and students are two kinds of roles in learning scenarios.

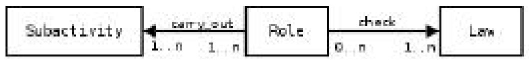


Figure 5: Relationships between different objects of a role

4.6 Actions

Anyway, we need to reflect in any way the actions made by tasks and other entities in the whole system. Therefore, we add them as an object in the semantic network. This is the purpose of the action object.

5 From the Task Model to the AmI System Model

This section describes the application of the proposed design model to the previous case study, showing the transformation of the abstract conceptual model of AMENITIES into the corresponding design model for these components, emphasising on the AmI characteristics.

Fig. 6a shows the model of a *Teacher* role using our proposed design model based on the objects, according to the previous specification of the roles for this case study (see [Fig. 2a]). Note that the relations among objects together with the object attributes allow maintaining contextual information about the modeled system without loosing the hierarchical structure of task modeling. [Fig. 2b] has showed the modeling of collaborative task *ScheduleClass*, and Fig. 6b shows the set of subactivities that the role *Scheduler* performs inside it. The information related to the data flow about the shown part has been omitted due to clarity, since the interesting part is centred on the actor System, which plays the role Scheduler as shown.

Accordingly, we can check how the participation of the system is made explicit in relation to perform tasks. It determines which task/subactivities are performed automatically. Conversely, in the conceptual model, it has only been specified the set of roles which takes part in the tasks. However, in this phase, the underlying AmI nature is taken into account on the modeling. The task to be carried out must be determined to provide proactivity.

Additionally, those in which the system will participate or aid users must be determined to. As it is an AmI system, it can be considered that the system is present in the performance of every task in an implicit manner. For example, inside the VoteDates task which is carried out by the student, the system is responsible for showing the different options to the student; it may be qualified for, in an automatic way, asking for the academic schedule of this student, remarking the applicant dates which coincide with leisure in its schedule and providing the necessary information about it. We must note that the conceptual model developed as a semantic network contains context information too, as reflected in Fig. 7.

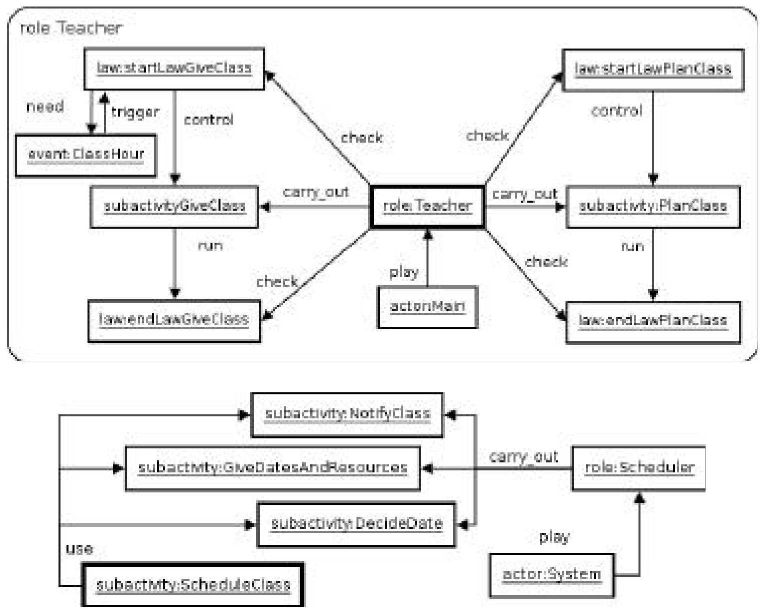


Figure 6: (a) Teacher role and (b) Scheduler role subactivities inside the ScheduleClass subactivity

The System actor is responsible of generating the event that will trigger the starting of a lesson (HourClass event in the begin transition of the GiveClass task) too, which implies that it must check the set of planned lessons every time, and it will trigger those which meets the specified requirements. Therefore, from this case of study we can extract a well known stereotyped behavior for schedule systems, composed by a set of meeting and a temporal trigger for them, which can be modeled as separated and be reused in any design. The implementation is done using the blackboard architecture of the U-CAT project, described in the following section.

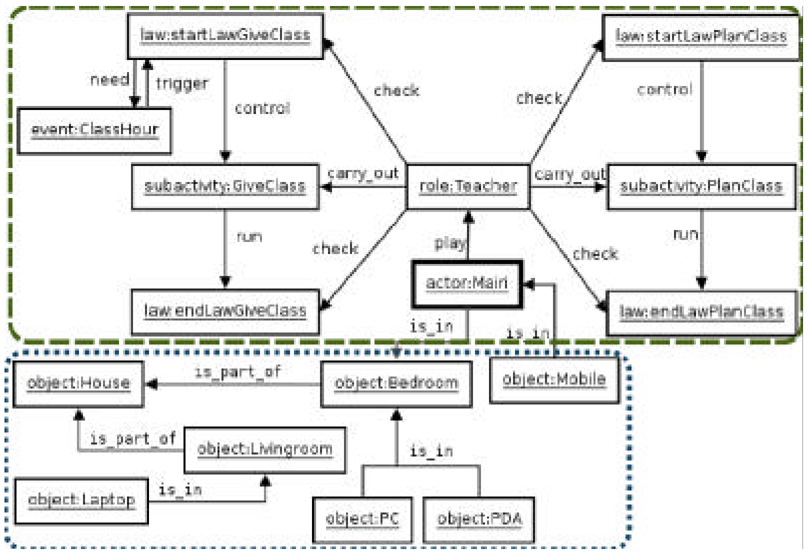


Figure 7: The network specifies both structure (tasks, roles) and context awareness (location)

6 Implementation based on a Blackboard Architecture

The AmI system is implemented on the basis of a blackboard model [Haya, 04]. The blackboard stores the relevant and information available about the environment. In it, the set of devices, people, relations, etc. are defined, establishing a world model. The specification shown in Fig. 6 is stored in the blackboard, which defines a model based on a semantic network about concepts and properties for maintaining the state and context of the system. The model stored in the blackboard is based on a global data structure. Classic blackboard implementations rely on a shared tuple data space.

On the contrary, the core of this blackboard implementation is a directed graph composed of entities and relations. One of the key benefits in using our approach is how the linked representation of the graph better fits the structure of the explained model rather than the tuple representation. Another important advantage is the improvement on the browsing mechanism. Every object (roles, actors, laws, events, etc.) that has to be introduced in the blackboard must be described using this graph for its storage.

The blackboard is based on the client-server programming paradigm. A reduced set of operation allows access to AmI model. Such operations include querying and updating model variables, discovering new entities and relationships and subscribing to blackboard changes.

The blackboard allows five kinds of clients that interact with it: sensors, actuators, interpreters, producers and consumers. These are distinguished according to whether they contribute to or obtain information from the blackboard, and according to whether they belong to the physical world or to the virtual world.

- Sensors. They are information sources belonging to the physical world. A sensor directly measures context from the real world, providing fine-grain information with very little abstraction.
- Actuators. These components convert blackboard changes into the real world. Alike sensors, actuators are physical devices.
- Interpreters. These components are subscribed to blackboard changes.
  Depending on how the changes are processed, we can split interpreters into two different types. The first ones convert raw sensor data into high-level information. These interpreters are new entities and relationships that are added to the blackboard model. The second ones divide complex tasks into several simpler actions.
- Consumers. They are the final receivers of the blackboard changes. This group includes user interface views or event-based autonomous applications.
- Producers. This group are composed of applications, modules and agents, which update the AmI model stored at the blackboard. These changes can be received by consumers or can be interpreted as commands affecting the physical world such as a temperature variation.

Fig. 8 summarizes the interaction between these five groups of components. As it can be shown in this figure, the blackboard acts as a rendez-vous point for the rest of the AmI system components. Dotted arrows indicate events produced by changes on the AmI model. On the contrary, solid arrows reflect two different types of communications: queries and updates. The first ones retrieve information about the entities and relationships of the AmI model. The last ones modify the value of some property or change the number of entities or relation of the model.

When producers/sensors/interpreters need to communicate new changes, they modify the information stored on the blackboard. There are two ways that consumers/actuators can use to find out what new information is available: they can either consult the blackboard to see if there are any new changes or if they can subscribe to blackboard modifications whereby they are notified of any modification.

Currently, the blackboard supports a collection of heterogeneous control buses such as EIB or X10. Various drivers provide a transparent gateway between the virtual and physical world. Hence, the status of the physical devices is reflected in the value of model variables, and vice versa.

One of the advantages of the proposed paradigm stems from the fact that it is not necessary for each client to be aware of the existence of the remaining components; each client only knows the location of the blackboard and the part of the model they are interested in. This approach loosely connects the different components on two levels: a temporal level and a spatial level. On one hand, clients do not need to be synchronized, which means that a producer can make changes to the model and finish its execution. Then, a consumer can make a request to the blackboard and retrieve the change since it has already been stored. On the other hand, when a client makes a modification on the blackboard, he/she will not be aware of the users affected by that change. Each client interacts with the blackboard as if they were the only one; hence, the development is easier.

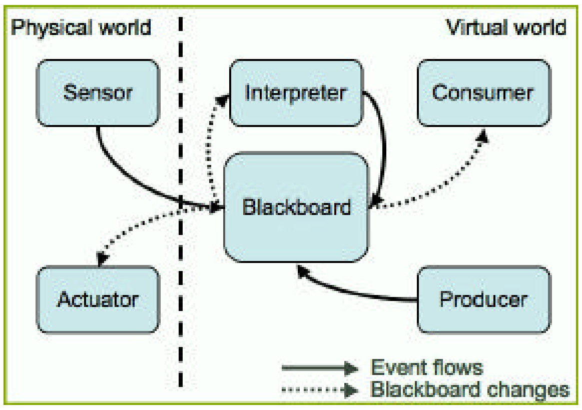


Figure 8: Blackboard architecture

As shown in Fig. 6b, we already separate in a conceptual way the *Scheduler* role, as this is played by the *System* actor. In the implementation stage over the blackboard these parts are clearly identified. The System actor is implemented as a producer-consumer client of the blackboard, responsible of carrying out the assigned tasks. For example, when it will execute the *GetDatesAndResources* task, this client will query the blackboard to obtain the state of everyone of the requested resources by the teacher in the selected dates. It must consult the dates by which every resource is available, and generate the expected result from them.

In summary, the system acts inside the blackboard model as a loosely connected agent in charge of managing a set of resources from its relation with roles, preferences, etc. Additionally, from the point of view of the blackboard, it can be a producer/consumer of transverse context information to many tasks, as a prerequisite or as a resource pager (simply changing the set of roles and policies which it depends on.)

7 Conclusions and Future Works

This paper presents the application of a task-based methodology, called AMENITIES, and also the use of a blackboard-based implementation for a learning AmI system, in the scope of the U-CAT Spanish project. The importance of AmI systems is growing in the last years because of its presence in different types of scenario (collaborative learning, mobile computing, etc.) The features of AmI systems imply an inherent complexity to address their development. Therefore, methodologies and techniques aimed at ensuring the success of the system should be applied. In this way, a proposal of a task-driven design has been devised to reflect the particularities of AmI systems. This design encourages the use of context, regarding the advantages of a conceptual design model based on tasks.

A design model (based on elemental patterns among entities) is proposed allowing us to model and implement collaborative tasks in learning AmI environments on the basis of crucial concepts related to collaboration, context, and proactivity.

In relation to the implementation of the system, the blackboard interaction paradigm provides flexible design and deployment procedures. A new client can be added and removed dynamically without affecting the configuration of the clients already present in the AmI system. Consequently, the development of new components is a process that can be done in an independently way. Each new client manages a part of the whole model, without concerning which other clients are already interested in the same part.

But, it is worth noting that an immediate problem arising when two or more different components try to update the same part of the model concurrently. Since each client is not aware about the existence of the rest of the components, it is necessary to have a mechanism that coordinates the access to the blackboard [Haya 06].

Future work is oriented to explore more general scenarios, fulfilling the implementation of this collaborative model on the blackboard. We are also interested in adding task restriction thus solving capabilities to increase the proactiveness.

Further works will be driven to provide means to facilitate the representation and management of the exposed concepts such as users, activities and resources in collaborative intelligent environments using context-aware information for task scheduling. One of the main features to achieve is to facilitate the coordination of different users when doing collaborative activities from diverse locations though different devices. We aim to provide dynamic task scheduling and decision support, which facilitates human coordination and collaboration. A theoretical prototype is being developed for this approach [Arroyo 07].

Acknowledgements

This research is partially supported by a Spanish R&D Project TIN2004-03140, Ubiquitous Collaborative Adaptive Training (U-CAT).

References

[Aldunate 02] Aldunate, R. Nussbaum, M. and González, R. An Agent Based Middleware for supporting Spontaneous Collaboration among Co-Located, Mobile and not Necessarily Known People. Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments, CSCW 2002, New Orleans, USA, November 2002.

[Arroyo 07] Arroyo, R.F., Gea, M., Garrido, J. L., Haya, P.A. , Carro, R.M. Smart Group-Aware Navigation for Active Spaces. Fifth International Workshop on Authoring of Adaptive and Adaptable Hypermedia at the 11th International Conference on User Modeling (UM'2007). In press.

[Campbell 03] Hess, C. and Campbell, R.: An application of a context-aware file system. Personal Ubiquitous Comput. 7, 6 (Dec. 2003), 339-352. 2003. DOI= http://dx.doi.org/10.1007/s00779-003-0250-y.

[Card 00] Card, S. K., Newell, A., and Moran, T. P.: The Psychology of Human- Computer Interaction. Lawrence Erlbaum Associates, Inc. 2000.

[Dey 00] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P.:. Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing (Karlsruhe, Germany, September 27 - 29, 1999). H. Gellersen, Ed. Lecture Notes In Computer Science, vol. 1707. Springer-Verlag, London, 304-307.

[Garrido 05] Garrido J.L., Gea M. Rodrguez M.L.: Requirements Engineering in Cooperative Systems. Requirements Engineering for Socio-Technical Systems. Chapter XIV. IDEA GROUP INC. (USA), (2005) 226-244

[Haya 04] Haya, P. A., Montoro, G., Alamán, X.: A Prototype of a Context-Based Architecture for Intelligent Home Environments. CoopIS/DOA/ODBASE (1) 2004: 477-491

[Haya 06] Haya, P.A., Montoro, G., Esquivel, A., García-Herranz, M., Alamán, X.: A Mechanism for Solving Conflicts in Ambient Intelligent Environments. Journal of Universal Computer Science, 12, 3. ISSN 0948-6968. 2006.. 284-296.

[Mellor 04] Mellor, S. J., Scott , K., Uhl, A., Weise, D.: MDA Distilled, Addison-Wesley, 2004

[Nardi 95] Nardi, B. A. : Activity theory and human-computer interaction. In Context and Consciousness: Activity theory and Human-Computer interaction, B. A. Nardi, Ed. Massachusetts Institute of Technology, Cambridge, MA, 1995. 7-16.

[Nylander 03] Nylander, J.: E-clasroom - 2003 Award Winner. Larry L. Sautter Award. 2003 DOI= http://www.cnc.ucr.edu/sautter/index.php?content=2003/recipients.html

[Paterno 99] Paterno, F. 1999 Model-Based Design and Evaluation of Interactive Applications. 1st. Springer-Verlag.

[Paterno 03] Paternò, F.: ConcurTaskTrees: An Engineered Notation for Task Models. In The Handbook of Task Analysis for Human-Computer Interaction, pp.483-503, Lawrence Erlbaum Associates, Mahwah, 2003

[Riva 05] IST Advisory Group. Ambient intelligence: from vision to reality. In G. Riva, F. Vatalaro, F. Davide, and M. Alca niz, editors, Ambient Intelligence. IOS Press, 2005.

[William 04] Griswold, W. G., Shanahan, P., Brown, S. W., Boyer, R., Ratto, M., Shapiro, R. B., and Truong, T. M.: ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing. Computer 37, 10 (Oct. 2004), 73-81. DOI= http://dx.doi.org/10.1109/MC.2004.149

[Veer 00] Veer, G.C., Welie, M.: Task Based Groupware Design: putting theory into practice, In Proceedings of DIS 2000 , 17-19 August 2000, New York