# A Design Model Applied to Development of AmI Systems

R.F. Arroyo[1], J.L. Garrido[1], M. Gea[1], and P.A. Haya[2]

[1] Universidad de Granada
{robfram,mgea,jgarrido}@ugr.es
[2] Universidad Autónoma de Madrid
Pablo.Haya@uam.es

**Abstract.** Ambient intelligence (AmI) represents a promising paradigm for group-centred collaborative interaction with the surrounding environment. The complexity for AmI designs is closely connected with the mechanism for describing their inherent features. What would be interesting is a method which is capable of describing these properties in a straightforward way. Task modelling techniques are a suitable method for AmI systems. This paper describes a new design and implementation proposal for developing AmI systems, starting from the conceptual and methodological frameworks proposed by AMENITIES, a methodology based on task and behaviour models for the study and development of cooperative systems, extending it with inherent AmI features. With respect to the implementation of AmI systems, an intermediate software layer supporting common functional requirements is supplied in order to simplify their development. The overall scheme therefore simplifies the analysis and development of such systems. These features are shown in a case study of a collaborative e-learning AmI system.

## 1 Introduction

Ambient intelligence (AmI) has become the next step in the approach of user-centred applications which integrate technology into an omnipresent and transparent infrastructure intended to implement intelligent environments. AmI systems attach special importance to friendliness, performance and support for human interaction [3]. High quality information must therefore be available (irrespective of the specific user, their physical location, the time and the device used) in order to enable people and devices to interact with each other and with the environment. Task modelling [14] is a useful technique for describing interactive/collaborative systems, and it transforms user activities and their required data into structured knowledge fragments which are based on tasks. In general, a task is described on the basis of a set of actions to be performed, information that is required for these actions, and the actors (playing roles) who perform them. Normally, however, no context information is taken into account in task modelling, or at least, it is not represented in a appropiate way. . The benefits arising from the use of methodological frameworks that include and integrate

(as one more part) task analysis and modelling can be even greater. In [10], a context layer is proposed which is implemented by using a blackboard model-based middleware and maintains a global data structure for relevant information about the world model.

Section 2 of this paper discusses the AmI system features. Section 3 briefly introduces the conceptual and methodological framework (called AMENITIES) that is the basis for our proposal. Section 4 describes the proposed design for developing AmI systems. Section 5 shows how this design model is implemented. Section 6 describes the application of the proposal to a specific e-learning system. The final section summarises the main contributions and outlines our future lines of research.

## 2 AmI Systems Features

In this section, we shall review the AmI system features in order to provide a better understanding of this kind of system. These systems are *context-aware*, which means that "any information can be used to characterise the situation of an entity" [2].

Most AmI scenarios are oriented towards groupwork and this results in collaborative spaces. These active spaces mean that people work on a constantly changing context [1], and this has led us to speak about *dynamic spaces*. One requirement is *proactiveness* and this is defined as the system's ability to make its own decisions and to start the actions which it considers appropriate without the user's attention having to focus on the interaction dialog; new user behaviour models have been proposed [12] for this. *Shared knowledge* is a very important issue since user context is distributed throughout the environment [18]. The development of applications to solve everyday tasks is also important and one example is the Stick-e Notes [13]. *Usefulness* is therefore also an important factor for these systems. A complementary theory comes from situated social interaction [17] which analyses both human interaction with other participants and computational devices and also the influence between them. Although theories and ontological methods are suitable for a better understanding of AmI spaces, it is also necessary to take into account these concepts in design models, and so conceptual and methodological frameworks should be adopted. Scenario-based design [5] is a well-known technique for problem domain understanding and requirement elicitation, and this has sometimes been proposed as an alternative method for task-based design. Non-expert users, however, may find it difficult to manage its natural language.

## 3 Conceptual and Methodological Framework

AMENITIES methodology framework has been adopted as a conceptual and methodological framework. This methodology is based on task models and user behaviour and has been specially devised for the study and development of

cooperative systems [6]. In practice, it has been successfully applied to several collaborative systems [8,7].

AMENITIES shows similarities to other methodologies (e.g. COMMONKADS [16]), they start from a conceptual structure and build behavioural models in the different phases of its life cycle. But unlike COMMONKADS, AMENITIES is a less wide methodology in which the most relevant information related to concepts such as task, role, organization, ..., is included in a unique hierarchical behaviour model (UML statechart). It makes easy the information access, and even provides a better understanding of the complete system. Thereby, this methodology proposes the building of a system model as starting point for the software development [6,7]. Abstract concepts present in AmI systems are part of those present in AMENITIES. Due to space limitations, this paper treats only with the following concepts:

*Law* (which is defined as a restriction imposed by the system enabling the set of possible behaviours to be dynamically adjusted, allowing or denying for example the execution of user tasks), *precondition* (defined as a set of restrictions to be checked, *subactivity* (as a combination of the actions carried out by active or passive entities), *event* (which is defined as a significant occurrence or happening to a task or program), *role* (which is a set of related tasks to be carried out), and *actor* (which is a user, program, or entity with certain acquired capabilities like skills, category, etc. that can play a role in the execution of, or responsibility for, actions).

## 4 Design Model

### 4.1 Basis of the Proposal

The abstract concept of *object* is the mainstay of the design model proposed in this paper. We can define an **object** as the *most elemental abstraction of any entity*, and this can be physical or not. Every entity is considered as an object, regardless of its nature or condition; it therefore establishes the basis for a homogeneous representation. As the aim is to design dynamic environments, we must establish a way to associate behaviours and properties to objects. We are interested in specifying what characterise the behaviour of an object, and in order to achieve this objective, we use the *interface* concept. We define an **interface** as a *specification of a set of properties and/or methods that an object exports*.

Physical objects are abstracted into logical objects which shall be represented in the system. We then proceed to separate the object properties from the part representing its behaviour. This distinction allows us to divide the model into two classes of components: the *implementation* of an object (simply called object), separated from its *specification* (interfaces). Figure 1 shows a diagram where the abstraction stratification has been performed. Interfaces can be associated with any object but not with each other. Since the objects can be related with any number of objects and interfaces, an object can be related with various objects in order to compose new objects, and an object can be related with multiple

interfaces to complete its behaviour. We can express the connectivity of objects and interfaces with Figure 1. This illustrates an example consisting of a device (`computer001`), comprising a physical computer placed somewhere in the real system, encapsulated into a logical object (`terminal`), which we will add as information to the system; this terminal is used as a mail client and instant messaging client, so it will implement at least two different interfaces which are related to these functions (`emailClient` and `IMClient` interfaces).

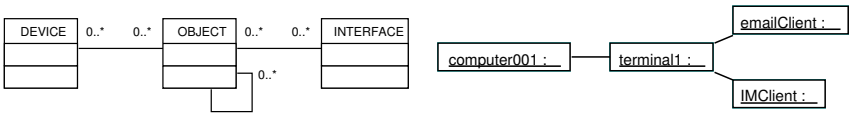


**Fig. 1.** Composition of devices, objects and interfaces. Example of relationship between them.

### 4.2 Model Entities

In the following section, we shall describe some of the main entities that shall be dealt with in Section 6.

**Law** This has the following properties: *self-information* for identifying 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 system activity. If this *logical expression* has not been specified, then the AND operator between preconditions is assumed by default.

**Precondition** This comprises *self-information*, a set of *restrictions* specifying a list of particular attributes that candidate objects to be chosen must have in order to carry out 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 obligatory nature can be used as a preference criterion for the candidate object choice.

**Event** The information about the specific use of the event is specified in the link connecting this event with the object that uses it. This specification consists of the *type* of relation with the event, i.e. whether the event is being sent or received; a *roles* expression including at least one role or a composition of some of these using an exclusive OR operator, or the reserved word `any` followed by a list of

roles to be excluded; and a list of *parameters* that might be necessary for certain events.

## 5 Implementation of the Model

The previously proposed design is mainly oriented to AmI systems in particular and has been used in the U-CAT (ubiquitous collaborative adaptive training) project to build intelligent active e-learning spaces. It has been implemented using a blackboard-based architectural design. This architecture (as a middleware between the physical world and the context-aware applications) is based on a paradigm called the blackboard.

The blackboard stores the prominent information that is available about the environment at any time. There are two different kinds of clients that interact with the blackboard: producers and consumers. The producers modify the information stored on the blackboard. The Consumers can either consult the blackboard to see if there are any new changes or they can subscribe to blackboard modifications whereby they are notified of any modification. One of the consumers included by default is called the solver When all the restrictions and events of a law have been fulfilled, new changes are generated on the blackboard. These changes can fire new laws, and can become actions that affect the physical environment or context-aware applications subscribed on the blackboard.

The flexibility and simplicity of the blackboard architecture make it a very suitable solution for environments where the configuration changes frequently. This is the case of AmI scenarios. 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 that 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. A consumer can then make a request to the blackboard and retrieve the change since it has been stored. On the other, 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 so development is easier. The following operations are provided by the blackboard: *obtain information* (either a whole entity or the value of a property), *change information* (enabling the value of a property to be modified), *add new information* (allowing new entities to be added), *remove information* (whereby entities can be deleted from the model), *subscribe* (which allows a client to be notified of any changes on the blackboard), and *unsubscribe* (to remove an existing subscription).

The information stored on the blackboard is represented by means of a data structure which includes metamodel entities such as laws, preconditions, roles, subactivities, etc., in addition to physical and logical environmental elements such as devices, persons, rooms, interfaces, etc. The information is then accessed in a transparent way independently of its nature. A client wanting to know the

precondition of a certain activity or to consult the state of the slide projector installed in a classroom can therefore use the blackboard to access both types of information.

## 6 Case of Study

Applying the proposal to an e-learning AmI system, one fragment of a complex scenario related to the task of teaching a class might be described in the following way: *"When a teacher called Mairi wants to teach her class, she asks the system for a suitable classroom. About fifty students are expected to attend, so a classroom with a capacity for more than fifty would be the best choice"*. Amongst other entities, this scenario includes actors (Mairi), roles (Teacher), laws, etc. For example, the law will comprise a precondition on the classroom, the one which will generate an event for initiating Mairi's class in a suitable classroom. The precondition itself comprises an optional restriction on the classroom capacity, with a capacity of over fifty being preferable. As we mentioned before, information must be specified in XML for it to be capable of interacting with the blackboard according to our software implementation. The XML specification stores all the necessary information into the blackboard, as it was modelled. The solver functionality itself must be implemented in the blackboard, as a producer-consumer client subscribed to it. Thus, the goal of the XML specification is to provide a textual, standard representation for the system model. Figure 2 shows the specification of certain entities in XML.

It can be observed that an XML fragment defines an object (using `entity`) of law type (determined by `type="10"`), comprising a precondition (`PreRoom`) and an action to be performed (`ActionClassMairi`). We then define the precondition (specified as `type="11"`) that contains a restriction on the capacity attribute (`name="Capacity"`) to be greater than fifty (attribute value `>50`). As it is not mandatory, the obligatory attribute is set to false (i.e. `name="mandatory"` is set to `no`). The action is defined as another entity with another specific type (`type="17"`), and it consists of a set of instructions specified on a language that the solver can process, in this case, it generates the event `StartClassMairi` with the class chosen by the precondition `PreRoom` (`SendEvent ( StartClassMairi , PreRoom)`). This event is specified in the figure below, where information about the action entity `ActionClassMairi` is responsible for generating this event (using >> in relation `name=">>:any:class"`, and the specification of entity in `destination="ActionClassMairi"`). When all this XML information has been stored on the blackboard, clients processing information stored on the blackboard can operate with the data dynamically. This enables the solver to select required objects for subsequent activities and state preferences between possible candidates according to our model's preconditions and laws.

## 7 Conclusions and Future Work

In this paper, we have focused on the design of a specific type of AmI system and its application to e-learning. We presented a proposal for a task-driven design

```
<entity name="Law" id="1001" type="10">
  <property name="[Info]">
    <paramSet name="Self-Information" id="001">
      <param name="Name"> Mairi Class </param>
    </paramSet>
  </property>
  <relation name="precondition"
            destination="PreRoom" id="201">
  </relation>
  <relation name="action"
            destination="ActionClassMairi" id="202">
  </relation>
</entity>
<entity name="PreRoom" id="2001" type="11">
  <property name="[Info]">
    <paramSet name="Self-Information" id="001">
      <param name="Name"> Room prec. </param>
    </paramSet>
  </property>
  <property name="[Restrictions]">
    <paramSet name="Restriction1" id="101">
      <param name="Capacity"> >50 </param>
      <param name="mandatory"> no </param>
    </paramSet>
  </property>
</entity>
<entity name="ActionClassMairi" id="3001" type="17">
  <property name="Actions">
    <paramSet name="Action block" id="301">
      <param name="Instructions">
        SendEvent (StartClassMairi, PreRoom)
      </param>
    </paramSet>
  </property>
</entity>
<entity name="StartClassMairi" id="4001" type="15">
  <relation name=">>:any:class"
            destination="ActionClassMairi" id="401">
  </relation>
</entity>
```

**Fig. 2.** Specification in XML language.

which is able to capture the special features of AmI systems with an existing solver engine. It is implemented using a client-server architecture that uses a solver engine to provide the required functionality for this kind of system. Clients perform operations by means of remote procedure calls on the blackboard using HTTP transport protocol and XML language for the message format. Active spaces have a wide range of technologies available for communicating with the physical world [10], which can lead to the use of components with limited processing capacities, such as sensors and actuators available in domotic networks. For these cases, the blackboard has specific controllers that enable communication with several kinds of devices, thereby avoiding additional complexity for the other clients.

Our future research shall be directed towards making more general environments, which describe context-sensitive information in greater detail, and adding solving capacities to improve proactiveness, dynamic and collaborative spaces. To date, the AMENITIES methodology has also been extended with new concepts and the corresponding attributes.

## 8 Acknowledgements

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

## References

1. R. Aldunate, M. Nussbaum, R. González, 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" ACM CSCW 2002.

2. A.K. Dey, G.D. Abowd, P.J. Brown, N. Davies, M. Smith, P. Steegels: Towards a better understanding of context and context-awareness. Workshop of Context-Awareness (CHI-2000).
3. C.K. Hess, R.H. Campbell: An application of a context-aware file system. Pers Ubiquit Comput (2003) 7: 339-352
4. S. Card, T. Moran, A. Newell. The Psychology of Human-Computer Interaction. Hillsdale, NJ: Erlbaum, 1983
5. J.M. Carroll. Five reasons for scenario-based design. Interacting with Computers 13 (2000) pp 43-60.
6. Garrido J.L., Gea M. & Rodríguez M.L.: Requirements Engineering in Cooperative Systems. Requirements Engineering for Socio-Technical Systems. Chapter XIV. IDEA GROUP INC. (USA), 226-244, (2005).
7. Garrido, J.L., Paderewski, P., Rodríguez, M.L., Hornos, M.J. & Noguera, M.: A Software Architecture Intended to Design High Quality Groupware Applications. 4th International Workshop on System/Software Architectures (IWSSA'05) - Proceedings of the 2005 International Conference on Software Engineering Research and Practice (SERP'05) , LAS VEGAS (USA), 59-65, (2005).
8. M. Gea, J.L. Garrido, F.L. Gutiérrez, R. Cobos, X. Alamán: Representación del comportamiento dinámico en modelos colaborativos: aplicación a la gestión del conocimiento compartido. Revista Iberoamericana de Inteligencia Artificial, Vol 24, 2004
9. P.A. Haya, X. Alamán, G. Montoro: A Comparative Study of Communication Infrastructures for the Implementation of Ubiquitous Computing. UPGRADE, The European Journal for the Informatics Professional, Vol 2, 5, 2001
10. P. A. Haya, G. Montoro, X. Alamán. A prototype of a context-based architecture for intelligent home environments. International Conference on Cooperative Information Systems (CoopIS 2004), Lecture Notes in Computer Science (LNCS 3290), 2004.
11. G. Montoro, P.A. Haya, X. Alamán. Context adaptive interaction with an automatically created spoken interface for intelligent environments. IFIP Conference on Intelligence in Communication Systems (INTELLCOMM 04). Lecture Notes in Computer Science (LNCS-3283). 2004
12. N. Oliver. Towards Perceptual Intelligence: Statistical Modeling of Human Individual and Interactive Behaviors. PHD Thesis. MIT Media Lab, 2000
13. J. Pascoe, Nick Ryan, and David Morse: Issues in Developing Context-Aware Computing. HUC'99, LNCS 1707, pp. 208-221, 1999.
14. F. Paternò: Model-Based Design and Evaluation of Interactive Applications. Springer-Verlag, Nov, 1999
15. G. Riva, F. Vatalaro, F. Davide, M. Alcañiz.: Ambient Intelligence. IOS Press, 2005
16. Schreiber, A. et al: Engineering of Knowledge and Management - The CommonKADS Methodology. MIT Press, USA (2000)
17. A. Takeuchi, T. Naito: Situated Facial Displays: Towards Social Interaction. SIGCHI Conference on Human factors in computing systems, 1995
18. M.R. Tazari, M. Grimm, M. Finke. Modeling user context. 10th International Conference on Human-Computer Interaction (HCII), June 2003.