Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3142-3159
submitted: 23/7/08, accepted: 29/10/08, appeared: 1/11/08 © J.UCS

Taxonomy for Integrating Models in the Development of Interactive Groupware Systems

William J. Giraldo
([)Systems and Computer Engineering, University of Quindío, Quindío, Colombia
wjgiraldo@uniquindio.edu.co)

Ana I. Molina
(Department of Information Technologies and Systems. Castilla – La Mancha University
Spain
AnaIsabel.Molina@uclm.es)

Cesar A. Collazos
(IDIS Research Group, University of Cauca, Popayán, Colombia
ccollazo@unicauca.edu.co)

Manuel Ortega, Miguel A. Redondo
(Department of Information Technologies and Systems. Castilla – La Mancha University
Spain
{Manuel.Ortega, Miguel.Redondo}@uclm.es)

Abstract: This paper describes the taxonomy for designing interactive groupware systems. The taxonomy defines the objectives, methods and principles for classifying models and facilitates their integration. In particular, we show the integration process of models in two notations such as CIAN, which considers collaboration and human-computer interaction issues, and UML, which allows specifying the functionality of groupware systems. The proposed integration process is based on a software tool, called CIAT, developed to put our proposal into practice.

Keywords: model-based development, groupware, computer-human interaction, taxonomy, methodological proposal
Categories: D.2.2, H.5.3, H.1.1

1 Introduction

In this paper we propose a taxonomical approach for Model Based User Interface Development of Collaborative Applications. This proposal relates technologies such as Enterprise Architecture, Model Driven Architecture (MDA), meta-modelling approach, domain specific methodology (DSM), model transformation and framework-based development, etc. It supports the interface design of groupware applications, enabling integration with software development processes through UML notation.

Multidisciplinary teams face the challenge of balancing multiple interdependent, and sometimes conflicting, aspects in their designs. They require specifying different views, abstractions, abstraction levels, granularity and levels of detail. Usually, there

are different stakeholders sharing designs and models in various domains [Gutwin et al. 1998; Molina et al. 2006]. They use the more appropriate language according to their role. Each developer represents the system in a more effective manner by using adequate, readable, comprehensible and expressive notations supporting their job. For example: UML activity diagrams are adequate and provide good expressive power to describe activities. However, task models are more adequate to design usable interfaces [Paternò 2001]. Each specific language allows developers to perceive themselves as working directly with domain concepts [Kelly et al. 2008].

Nowadays, there are proposals such as task modelling, sketches, graphical templates, standard modelling languages, domain specific languages, etc.. All these come from various disciplines such as Software Engineering (SE), CSCW (Computer Supported Cooperative Work), and Usability Engineering (UE). However, there is still a gap between the development process of the functionality of CSCW systems and the development of their user interface , particularly, proposals that combine group work applications and interactive aspects [Molina et al. 2007]. Our goal is to reduce this gap. We propose to use CIAN1 [Molina et al. 2006] (a specific notation for modelling interactive and group work issues) for developing the user interface of groupware systems and the UML language to modelling system functionality. Therefore, we need to integrate information that is specified in CIAN models with the information gathered in UML models. Only part of the diagrams and part of the information specified in them is useful for our integration purposes. This integration proposal is done through an integration layer, which was subsequently generalised for defining a more generic taxonomy. We implement a software tool based on the eclipse framework and a case study.

This paper is organized in the following way: Section 2 introduces our methodological approach for designing interactive groupware applications, presenting a brief explanation of its stages and the aspects that can be specified in each one. Also, some aspects of the CIAN notation are described in this section. Section 3 introduces our integration proposal, especially the integration layer that supports it. This section presents an example in which a case study is used. Section 4 presents some related works and introduces the main concepts and methods used to specify the taxonomy. Also, the integration proposal is introduced. This section describes how the integration proposal could be generalized to be used in a other contexts. Finally, the conclusions and further work are presented.

2 CIAM: A Methodological Approach for User Interface Development of Collaborative Applications

In this section we present the stages in our methodological approach. CIAM considers interactive groupware modelling in two ways: group-centered modelling and process-centered modelling. First, the social relations are studied and an organizational scheme is specified. Next, the group work is modelled. The modelling becomes more user-centered when we go deeper into the abstraction level in which interactive tasks are modelled [Beyer et al. 1998]. In other words, a dialog arises between an individual

[1] CIAN Notation is commented in the whole text.

user and the application. In this way, collaborative aspects (groups, processes) and interactive (individual) modelling problems are tackled jointly. CIAM guides designers in creating conceptual specifications of the main aspects that define the presentation layer in CSCW systems. The stages of this proposal (see *Figure 1*) and their objectives are enumerated as follows and, in the next section, the proposed conceptual framework is explained.

Sociogram Development. In this phase, the organization structure is modelled, as well as the relationships between its members. Organization Members belong to these categories: *roles, actors and software agents*, or in the aforementioned associations, forming *groups* or *work teams* consisting of several roles. The elements in these diagrams can be interconnected by means of three kinds of basic relationships (*inheritance, performance* and *association*).

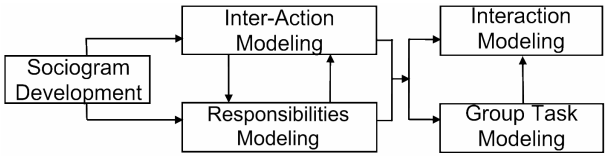

Figure 1: CIAM methodological proposal stages

Inter-Action Modelling. In this phase, the main tasks that define the group work in the previously defined organization are described. For each process, the roles involved, the data manipulated and the products generated are specified. Each task must be classified in one of the following categories: *group* or *individual tasks*. Tasks will be interconnected by means of several kinds of relationships that, in many cases, can be interpreted as dependence.

Responsibilities Modelling. In this phase, the individual and shared responsibilities are modelled. We can see that the specified information in this phase is supplemented by that of the previous one. Both models must be consistent with each other.

Group Tasks Modelling. In this stage the group tasks identified in the previous stage are described in a more detailed way. There are two different kinds of tasks, which must be modelled in differentiated ways: **(a)** *Cooperative Tasks* are specified by means of the so-called *responsibilities decomposition graph*, in which subtasks make up the group task, so that, at a lower abstraction level, only an *individual task* must appear. **(b)** *Collaborative Task* modelling includes specification of the roles involved, as well as the data model objects manipulated by the work team (that is, the *shared context* specification). Shared context is defined as the set of *objects* that are visible to the user set, as well as the *actions* that can be executed on them. Once the objects that make up the *shared context* have been decided, it is necessary to fragment this information into three different parts: the objects and/or attributes manipulated in the *collaborative visualization area*, the ones which appear in the *individual visualization area* and the ones that make up the *exclusive edition segment* (a subset in the data model that is accessed in an exclusive way by only one application user at a time).

Interaction Modelling. In the last phase, interactive aspects of the application are modelled. An interaction model for each individual task detected in the diverse phases of the gradual refinement process is created. An interactive task decomposition tree in CTT [Paternò et al. 1997] is developed. The interactive model is directly derived from the shared context definition. Our methodological approach includes the way of obtaining this model from the shared context modelling [Molina et al. 2007].

CIAM is an approach based on Model Driven Development (MDD), which promotes the use of models to simplify the complexity of groupware design [Frankel 2004]. The different stakeholders can understand each model using their own point of view without worrying about syntax or other platform-specific issues. CIAM is supported by a notation called CIAN (Collaborative Interactive Applications Notation). This notation is a simplification of another notation for workflow modelling, called APM (Action Port Model), proposed by Carlsen [Carlsen 1998]. This notation has been enriched to support differentiated modelling of cooperative and collaborative tasks, although it has been simplified in some aspects (to characterize a task, only the task identification, the roles involved and the objects manipulated are included).

3 The integration proposal

There are proposal to integrate the user interface design with UML. Paternó [Paternò 2001] integrates the CTT diagrams and use cases. Trætteberg [Trætteberg 2002] proposes a framework for classifying user interface design representations, presents languages for modelling domain, task and dialog, and he suggests how these modelling languages may be integrated with UML. But these proposals are limited by the existence of semantic correspondences between their own notations and UML, both at the conceptual and structure levels. They are exploiting UML extensibility mechanisms to support their solutions. Instead, our integration proposal of models in UML and CIAN is done through an integration layer. First, the CIAN diagrams are done in order to specify the collaborative interface. This model specifies the collaboration, the work of users, the objects, the passage of information, the coordination of activities, the relationship between interfaces and tasks, etc. This information populates the integration layer. Then, some modelling elements in UML are generated by mean of model transformations. Subsequently, the design is continued in UML in order to specify the functionality. The Review Conference System was used as a case study. This example has been chosen because it is referenced in literature and it is used in several approaches, it is extracted from [Schwabe et al. 2001].

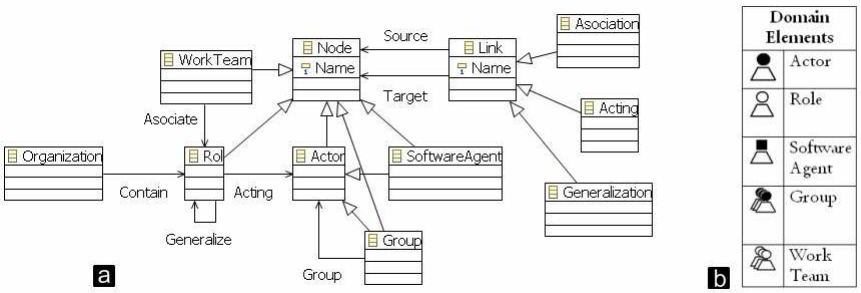

Figure 2: Sociogram metamodel.

A model transformations specification requires that the languages to be used in the transformation process are normalized. Therefore, our integration process begins with a ***Normalization of the languages***: MOF and the EMF Plug-in were used to normalize the CIAN metamodel. Only the abstract syntax is modified. The concrete initial syntax is preserved. *Figure 2* shows the Sociogram metamodel. In this case, the specific domain is the modelling of the organization structure. The domain concepts are {actor, role, software Agent, group, work Team} (*Figure 2*). We have specified the same modelling concepts in the metamodel. In this same sense, we have also specified the necessary relationships. The Sociogram is intended to be used to model collaboration, cooperation and interactive aspects. This differs from UML business actor and system actor models because the Sociogram provides semantic to specify the dynamics of the organization. However, if we model structural aspects of the organization, we can find some relationships and interchange points between them. The semantic of the Sociogram is expressed directly with abstract syntax and OCL sentences. By means of this metamodel, we have developed the modelling tool for this diagram by using GEF. In this example, we have the following roles: PC-Chair, PC-Member, Reviewer, Author and Co-Author. Figure 3(left) shows the structure of the organization. A PC-Member can be considered a specialization of a Reviewer, since he/she carries out the same work (revising), but specialized in carrying out another, wider, group of tasks or responsibilities. In addition, we can see that the PC-Chair and PC-Members' roles are associated. This indicates that there are tasks in which both, with their respective responsibilities, take part.

The integration layer structure which we propose is based on the Zachman Framework [Zachman 1987]. This Framework proposes a systematic taxonomy which allows us to associate concepts which describe the real world (domain concepts) with those which describe their information system (modelling concepts) and its subsequent implementation [Sowa et al. 1992]. This taxonomy is defined in two dimensions organized in perspectives and views. We use only the business model, system model and technology model perspectives and the data, function, network and people views. The intersection of views and perspectives leads to 12 modelling cells, (Figure 3(right)). Each cell provides a container for models that address a particular perspective and view. The Framework provides a representation from different points of view, different levels of granularity, generality and abstraction.

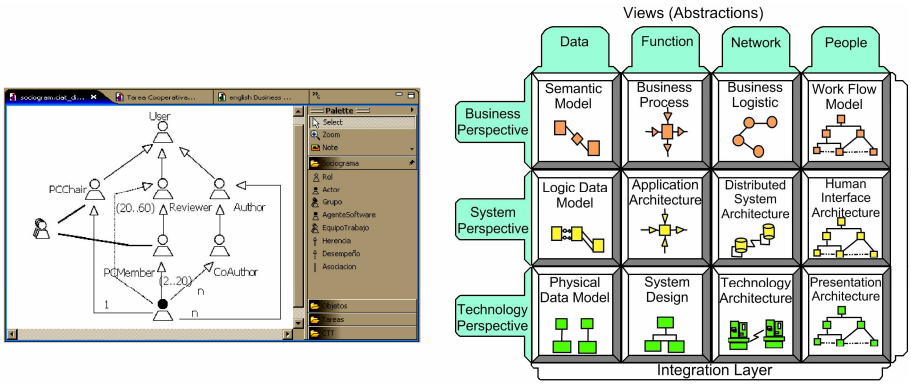

Figure 3: Sociogram model of case of study (left). Integration layer(right).

#	Description
1	The mapping between the use cases and the task models can be based on the following basic transformations [Lu et al. 1999]: (a) The use cases represent the highest levels of abstraction in the hierarchical task models. (b) The "uses" relations can be interpreted as temporal order expressions (in particular a sequence connection). (c) The "extends" relations indicate optional behaviours. This situation can also be specified in a task model. (d) Temporal dependencies are related to post conditions and preconditions in activities diagram.
2	Business entities provide domain information for activities. An Inter_Action model consists of a set of tasks carried out in a certain order and considering certain data or temporal restrictions among them. For each task, the roles involved, the data manipulated, and the product obtained as a result of the task are specified. For the data specified in the context of a task we can specify the access modifiers to the objects, which can be *reading, writing* or *creation*
3	The tasks in the model are interconnected by means of several kinds of relationships that can be interpreted as dependencies: *temporal dependencies* (order relationship), *data dependencies* (when tasks need data manipulated by previous tasks) and *notification dependencies* (when it is necessary for a certain event to occur so that the work flow continues). The dependencies act like preconditions and post conditions into activities diagram. It allows designers to define task attributes such as the category, the type, the objects manipulated, frequency, and time requested for performance. Inter-Action Model is more expressive than Activity diagrams and use cases in order to design logistic models
4	In the Business Object Model is defined the interaction between actors and domain objects. UML do not provide semantic for storing object access information. In CIAN, activities are enriched with information about access modifiers of objects, which is stored later into responsibilities table.

Table 1: Interchange Points for integrating CIAM and UML Models.

Finding interchange points: The interchange points are pieces of information from each language which provide structural or syntactical correspondence between them. We found some interchange points between UML and CIAM. In the next table, we present a summary of these interchange points. As a result, we define the following

modelling concepts: Data view {Entity, Relation}, Function view {Process, Input_Output}, People view {Person, Association}, and Time view {Event, Cycle}. These concepts are to store information from CIAN model into the integration layer. Concepts and their relationships are depicted in *Figure 4*. Besides, they are related to each other to integrate information from the models. For example, the variable event is linked to the variable cycle, person and entity.

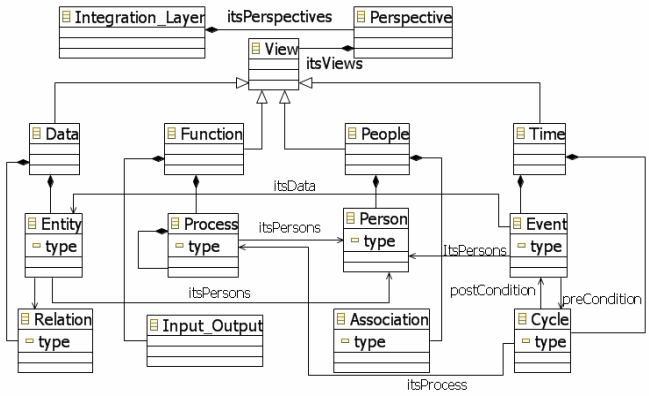

Figure 4: Integration layer base metamodel

Defining the base metamodel: The information into cells of integration layer must be related to each other in two directions, views and perspectives. Therefore, a base metamodel should be specified. A reduced version of the integration layer metamodel is presented in *Figure 4*. Each column of the integration layer defines a set of concepts to store information used in the integration. This metamodel supports the information of the interchange points. For example, the variable event is related with the variables Cycle, Person, and Entity. These relationships are specified in order to establish a link between temporal dependencies into the Inter_Action diagram in CIAN and preconditions and postconditions into activities diagram in UML, like it is explained within the interchange points one and three.

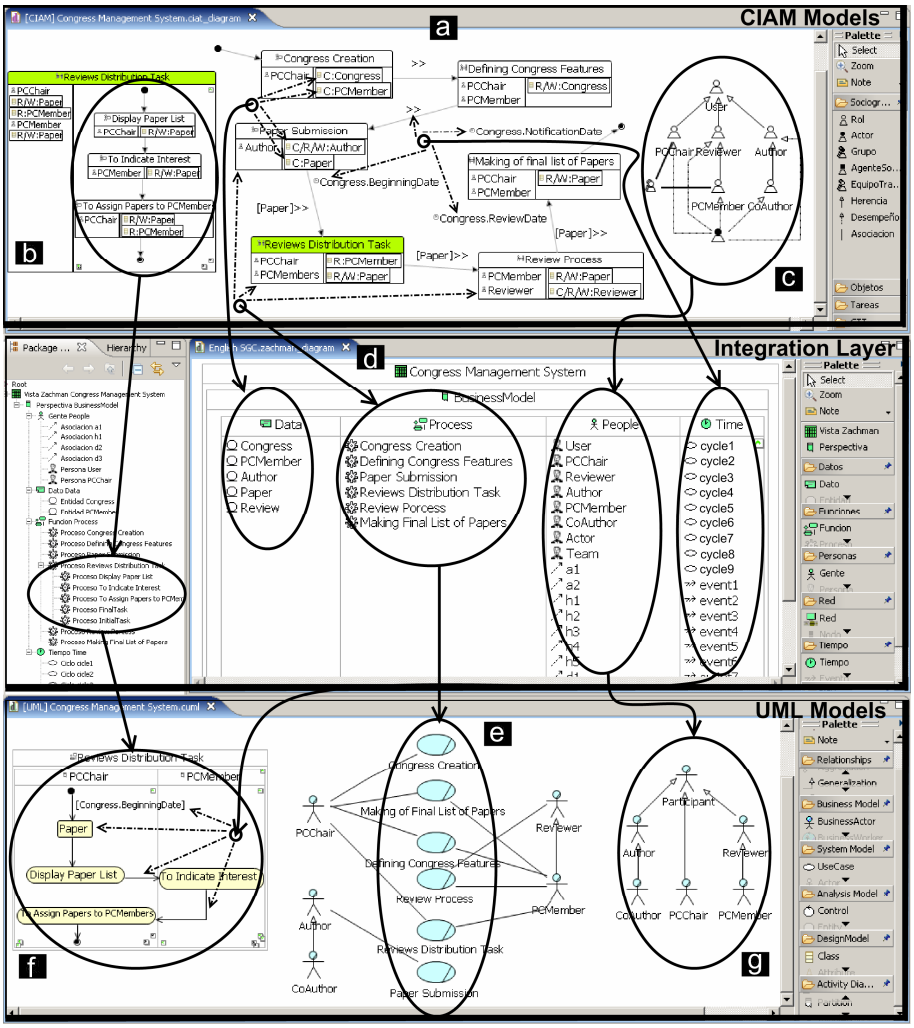

Figure 5: Transformation process by using CIAT.

Define transformations: The ATLAS Transformation Language (ATL) is used to implement transformations between models. The ATL plug-in is used to transform CIAM diagrams into a set of model elements in the integration layer. *Figure 5* illustrates the integration between the CIAN models and UML models by using the CIAT tool. The information about roles and relationships in the Sociogram is extracted through the transformation, which is used in the "Business Model" and "System Model" perspectives and the "People" view, mainly; see *Figure 5*(c). There is no direct translation of the acting and association relationships of CIAN into UML. However, this information should be stored in order to generate other artefacts. The Inter_Action diagram (see *Figure 5*(a)) illustrates the macro-activities of the system and their interdependencies. This model is essential, because certain temporary information (precedence and coordination information) is represented. This

information can be enriched by using information related to the domain (which is extracted from the models of the ES process). This diagram provides information about the pre-conditions, post-conditions, messages and data required or generated by the activities. UML lacks a diagram of this type. The process of transformation and integration is controlled through the integration layer metamodel. The first transformation uses the CIAN metamodel as the input metamodel and the taxonomy metamodel as the output metamodel. The second transformation uses the taxonomy metamodel as the input metamodel and the UML metamodel as the output metamodel. CIAT recognizes these three metamodels and it is possible to edit models using editors for each one. The validation of this proposal is carried out by developing a series of tools that allow the development of models of an interactive groupware system. The purpose of this validation is to verify compliance with the goals outlined in this paper. The development of modelling tools and the transformations were done in the Eclipse environment. This validation is supported by a tool, called CIAT (*Collaborative Interactive Applications Tool*). CIAT [Giraldo et al. 2008] is an Eclipse-based tool to support designers and engineers by creating models based on CIAN notation. This software tool supports the interface design of groupware applications, enabling integration with software processes through UML notation. Eclipse Framework provides tools for guiding the software modelling by using metamodel concepts [Moore et al. 2004]. By using the EMF (*Eclipse Modelling Framework*) and GMF (*Graphical Editing Framework*), we design the CIAT tool as an Eclipse Plug-in.

4 Taxonomy for interactive groupware systems

CSCW finds its bases on the concepts of collaboration, communication, cooperation and coordination, among others. These concepts have been related to the space and time, which has led to different classifications of CSCW tools. One of the first classifications was proposed by Johansen [Johansen 1988]. From here, other proposals adding new categories, which interrelate the aforementioned basic concepts of CSCW and the space and time, arise. However, it is not always possible to locate simple tools, and even less complex systems, into those categories. Penichet [Penichet et al. 2007] presents a taxonomy in which it is possible to classify a function, a tool or a system[2] regarding the spatial-temporal characteristics and the characteristic of CSCW systems such as collaboration, communication and coordination, *Figure 6*(a). Its proposal, up to a point, removes some of the discrepancies presented in the previous classifications, and allows separating into independent categories not only the aspects of collaboration, communication and coordination, but also the time and placing where these aspects are developed. All these classifications, or taxonomies, have been used to classify functions or subsystems that provide support to collaborative work, however, have not been used to classify models and modelling concepts of this kind of systems. We propose to create the taxonomy for this purpose [Giraldo et al. 2008]. Our taxonomy differs from the rest in the fact that it provides a

[2] This distinction is necessary because some of the services needed in CSCW can stand as functionality, as an embedded software component or as a system or software application by itself. A chat is an example.

framework for the classification of groupware systems based on diverse aspects or facets that are considered during its modelling. In addition, it promotes the appropriate integration between information expressed in different models.

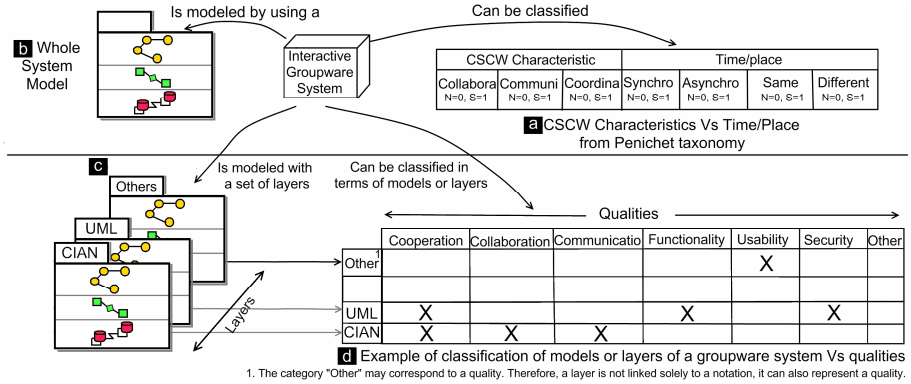

Figure 6: Classification of the qualities of an interactive groupware system.

The classification method is based on the assumption that an interactive groupware system can be classified and, therefore, modelled through one or more abstraction layers representing either several families or sets of specifications. This idea, expressed graphically in *Figure 6* (d), leads to the definition of our proposal. Modelling concepts can be shared by different layers, since each one is simply a realization[3] (delimitation, or abstraction) of one or more qualities[4]. Therefore, layers can be aligned so that when combined give a more complete view of the same modelling concept. In *Figure 6* (c) there is a layer –*Others*- that aims to integrate all models related to the "*usability*" of the system. Instead, the "*cooperation*" is supported by the UML and CIAN layers. In this example, each layer, CIAN and UML, contains the whole of all their respective models; however, a layer could contain diagrams that are supporting "*cooperation*" either in a specific language or in different languages. In systems with only one layer, this necessarily represents all their qualities, *Figure 6* (b). So each layer could be a stand alone software component. This hypothesis suggests that a CSCW system can be replaced by a set of software components in order to support one or more of their qualities.

Our taxonomy includes not only the taxonomic organization of the information of models but the methods, rules, and principles for classifying, organizing, and integrating the modelling concepts used in the specification of a groupware system. In that way, our proposal is similar to a methodology specifying not only the "what" (Taxonomic organization) but the "how" (methods) classifying the information in groupware systems.

[3] Realization is a mechanism used in the RUP to delimit or demarcate the set of modeling concepts that implement a specification. This mechanism is used mainly in use cases. A realization, therefore, is a view of all classes that implement functionality. A class can participate in various realizations

[4] Property, attribute, character, trait, characteristic, or aspects that make the specification of a system be fleshed.

4.1 Main concepts: Taxonomic Organization.

The taxonomy uses diverse concepts or categories in order to classify the information into an interactive groupware system. The main concepts are listed in the next table.

Concept	*Description*
Layer	A ***layer*** is a set of diagrams organized according to a particular criterion, for example: diagrams modelled with the same notation, diagrams representing a particular abstraction, diagrams representing a quality indicator, etc.
Perspective	A ***perspective*** is an architectural representation at a specific abstraction level and represents a set of logical or physical constraints that may affect the development of a system at that level. A key issue in software architectures is the support to handle different levels of abstraction. Perspective and viewpoint from MDA is the same.
View	The concept of ***view***, or abstraction, is a mechanism used by designers to understand a specific system aspect. The abstraction is the tool that enables software developers to manage the complexity of their developments. That is why we focus, first, on abstractions, and later on implementations that are derived from these abstractions [Kaisler 2005]. For example, the data view provides information about models of the domain system to be developed. On the other hand, the function view includes models which represent the processes and functions of the system. To capture all the requirements of a software system is necessary to provide multiple views.
Cell	A ***cell*** is a container for models that address a particular perspective and view. Models in each cell are specified by a domain specific language. Cells contain variables associated a concrete domain objects. These variables are completely independent from variables in other cells.

Table 2: Taxonomic organization.

4.2 Classification method

We propose a method for classifying, organizing, and integrating the modelling concepts used in the specification of a groupware system. Initially, our interest focuses on the integration of some models in UML and CIAN; however, the method can be applied to a large number of notations, each one is suited to specify various aspects of the system.

The integration or separation is carried out by using one or more integration layers, whose purpose is to store the useful and relevant information in each notation used for these purposes. A way to combine information directly from UML and CIAN models by using a layer of integration is showed in Figure 7 (a). The common information of model elements on both modelling notations is classified and organized in this layer in different perspectives and views.

This integration process could be accomplished in different ways. Some of the possible integration scenarios are :(1) first, diagrams in CIAN are done in order to specify the collaborative interface. This model specifies collaboration, work of users, objects, passage of information, coordination of activities, relationship between interfaces and tasks, etc. Subsequently, the design is continued in UML in order to specify the functionality. *Figure 7* (a) illustrates this process. (2) It began its design in

UML so as to specify the functionality. Subsequently, it continues the design in CIAN with the aim of specifying the collaborative interface. *Figure 7*(b) illustrates this process. (3) It combines the two previous stages. Some transformations in the integration layer are necessary to synchronize the models, which are developed in parallel, *Figure 7* (c).

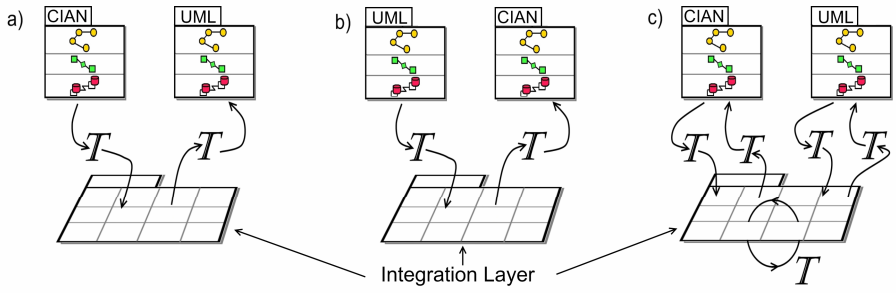

Figure 7: Integration scenarios of CIAN and UML.

The classification method establishes the principles, rules and steps in order to formalize the organization of the layers, languages and the information into the interactive groupware system. In this sense, we suggest the use of metamodelling to specify both abstract and concrete syntax for each language. The abstract syntax denotes the structure and grammatical rules of a language. The concrete deals with notational symbols and the representational form used by the language [Kelly et al. 2008]. There are layers which aim to store information related to a particular aspect. These layers are often populated with information specified in multiple languages or these are storing complementary information of modelling objects. These layers have only abstract syntax, because the stored information does not necessarily have a visual sense and it is only for purposes of integration. However, other layers have both concrete and abstract syntax.

The taxonomy defines a series of steps both for classification and for integration of information. The following is a brief description of these steps:

1) Normalization of the languages: In order to provide a workspace, the sharing models, the model integration, etc., each language must be defined formally. This normalization is done by means of metamodels. These metamodels should be formalized keeping only one domain in mind. Our proposal allows developers to work in their usual languages; however, it is possible that there is a need to make changes for each language. We propose to fit only the abstract syntax respecting the concrete syntax.

2) Formalization of the integration layer: A layer can be specified to store the information that a notation exposes to the other layers for integration purposes. An integration layer can be seen as a mapping that provides specifications for transformation of a language into other language. An integration layer must support the specification of different views, abstractions, abstraction levels, granularity and levels of detail. The formalization of the integration layer requires a similar process to that one conducted by the domain specific languages. The main steps are:

(2a) Defining the integration layer structure: The integration layer is defined as is aforementioned. Each cell contains information that belongs exclusively to that cell and that is different from other cell.

(2b) Defining the focus of architecture: This classification uses perspectives enabling designers to establish independence between different levels of abstraction, however, it is necessary to have a solid architecture that allows its subsequent integration. MDA (*Model Driven Architecture*) [Miller et al. 2003] is an architecture that promotes design guided by models and, as can be seen in *Figure 8* (c), there is a relationship between the perspectives and levels of MDA. Frankel et al [Frankel et al. 2003] describe the mapping between Zachman Framework and MDA.

(2c) Defining the rules: To obtain integrity, uniqueness, consistency and recursion of the information specified, taxonomy defines a series of rules. Thus, the seven rules of the Zachman Framework have been adopted and refined [Sowa et al. 1992]. Examples of these rules are: (R2) all of the cells in each column-*view*- are guided by a single metamodel. (R5) The composition or integration of all models of the cells in a row is a complete model from this perspective. (R7) The logic is recursive

(2d) Defining the variables of integration layer: The variables are a set of classifiers associate to modelling concepts that belong to each language. These variables should be classified according to a perspective and a view associated with a specific cell. Variables in a cell are completely independent of variables in other cells. These variables generalize the modelling concepts in order to provide a matching between information from different languages. In other words, the interchange points are pieces of information from each language which provide structural or syntactical correspondence between them; while, the variables are modelling concepts for modelling these correspondences.

(2e) Defining the base metamodel: The information into cells of integration layer must be related to each other in two directions, views and perspectives. Therefore, a base metamodel should be specified (*Figure 8*(a)). This metamodel control the models cells consistency into the same view -*rule 2*- and it is necessary for the integration or composition of the models into cells of the same row -*rule 5*- performing an integration role at perspective level. Each column –*view*- has a simple, basic metamodel which represents an abstraction from the real-world for convenience of the design [Sowa et al. 1992]. Although, the concepts in a view are representing different things in each perspective, they are related each other in the same manner. Because they belong to the same abstraction. (i.e. Business objects, analysis classes and design classes belong to different viewpoint however they have the same metamodel). It is possible to specify a different base metamodel for each integration layer, which depends on the nature of the family of languages (DSL) which is specifying. For example, by integrating UML and CIAN it could provide a single integration layer to store common information useful for integration. However, it could have an integration layer for each notation, providing an additional benefit because each notation may expose the information provided to the other one and not just to one in particular. Multiple integration layers can coexist in a system. -See *Figure 8* (e)-. An integration layer can be associated directly to a layer -see *Figure 6*-, a notation or a one o various qualities. This represents a new dimension, which is defined by grouping integration layers needed in an interactive groupware system. Each level in this dimension represents a group or family of specific domain

languages used to modelling one or more qualities of the system. *Figure 8*(e). The *Figure 8*(c) illustrates the rule two and five, these rules are necessary in order to define model consistency in each view and perspective.

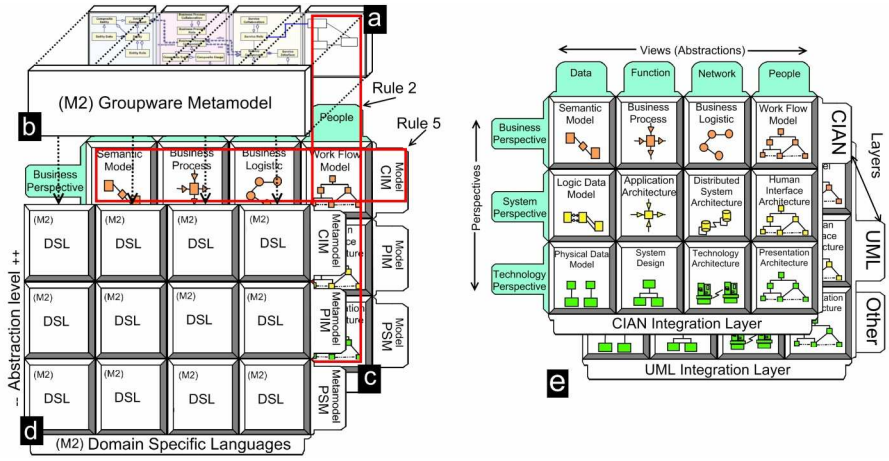

Figure 8: Base metamodel definition. Multiple integration layers.

(f) ***Defining the metamodels of the cells***: The integration layer should, at least, consist of modelling elements representing quality-specific concepts at various levels of abstractions. These modelling elements are modelled with domain-specific languages (DSL). A challenge is the need to integrate cells to obtain a holistic view of a design. One way to integrate these cells is to define a base metamodel that describes the relationships among concepts defined in the different cells. In this same manner, the abstract syntax of each cell is defined by a specific metamodel. This set of metamodels is restricted by the base metamodel –see rule 2-. Each integration layer metamodel has an extended abstract syntax based on the shared modelling concepts from the languages. i.e. CIAN and UML. However, a concrete syntax is not always necessary in each cell. MDA provides the conceptual structure for specifying the notations or domain specific languages (DSL) used in every cell in the integration layer. Additionally, it allows to implement this notation by means of software tools by using specific applications within an architectural approach [Frankel 2004]. Therefore, each one of these models of the cells is related to their respective metamodel (DSL).

(g) ***Defining transformations***: All models into MDA are related as they are based on a more abstract metamodel called MOF (*Meta Object Facility*) [Miller et al. 2003]. MOF facilitates the definition of the necessary transformations to integrate models. We define transformations both model to model (M2M) and Model to Text (M2T). These can be made between two different layers, in a single layer, horizontally - same perspective- or vertical-same view. The integration between UML and CIAN layers is possible because these are consistent with the MDA levels.

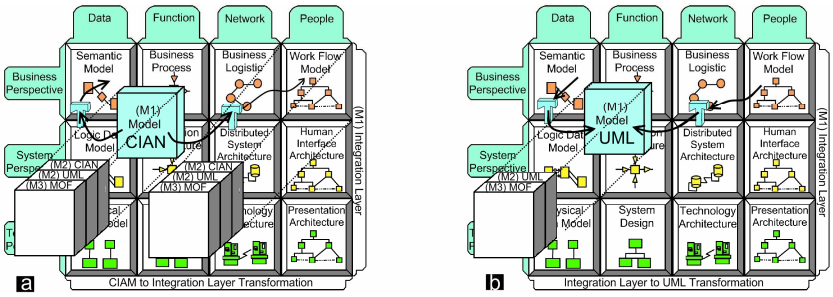

Figure 9: Integration between CIAM and UML

MDD proposes model transformations to reduce the complexity of software design [Frankel 2004; Jouault et al. 2006]. The integration of models in UML and CIAN is done through an integration layer; see *Figure 9*. The integration layer is populated by using transformations applied to CIAN models; see *Figure 9*(a). The structure of notations is represented by some boxes containing metamodels at M2 and M3 levels. *Figure 9*. The cell that contains the CIAN diagram -Sociogram- lies in the level M1 (Model); in addition, the notation CIAN which is defined as a UML Profile lies in the level M2 (metamodel). The transformations have as input metamodel CIAN and as output metamodel DSL defined for these cells. In the *Figure 9*(b) the process to transform models from the integration layer to generate UML diagrams is shown. UML diagrams fully specified not always are possible; therefore, the generated information serves as a starting point for the subsequent modelling in UML.

5 Conclusions

The development of systems to support group work is a complex task, among other reasons, because of the nature of the groups involved in this process, whose members are often from different areas of knowledge. They have different needs depending on their perspective and understand the artefacts manipulated from a specific point of view. Similarly, when they develop the software system, there are different aspects or qualities (usability, support to collaboration, functionality, etc.) to be enhanced. Moreover, the possibility of working with different abstractions (views) facilitates the management of complexity of the development of this kind of systems.

Contemplating all these possibilities has led to a proposal based on the definition of a three-dimensional taxonomy (*perspective-view-layer*) that facilitates the integration of the notations used by various members of the development team, by supporting various modelling aspects and qualities, as well as the definition of transformations between them. From another point of view, this taxonomy can also be used as an evaluation framework for classifying notations. It would possible, in this case, to define metrics, indicators, or indexes of coverage for each notation. in this manner a developer can evaluate if each notation provides modelling elements in the required qualities, perspectives and/or views.

In particular, this paper has shown a proposal for integrating modelling information from two notations such as UML, which provide a suitable support for the modelling

of system functionality and CIAN, which focuses on modelling collaboration and interaction with the user. By extrapolating the results of integration of both proposals to other notations in the literature, we could conclude that the taxonomy proposed makes it possible the connection between the proposals from the field of Software Engineering and the Human computer interaction.

It has been developed a software tool known as CIAT, which implements the ideas presented here. This tool not only allows the editing of models in CIAN notation, but also performs integration and transformation of the standard UML models, using the taxonomy proposal as an intermediary artefact.

Acknowledgments

This work has been supported by Universidad del Quindío, Castilla – La Mancha University and the Junta de Comunidades de Castilla – La Mancha in the projects AULA-T (PBI08-0069), M-CUIDE (TC20080552) and mGUIDE (PBC08-0006-512). Also, by the project CICYT TIN2008-06596-C02-2.

References

[Beyer et al. 1998] Beyer, H. and K. Holtzblatt: Contextual Design: Defining Customer-Centered Systems. 1998. Morgan Kaufmann Publishers Inc.

[Carlsen 1998] Carlsen, S.: Action Port Model: A Mixed Paradigm Conceptual Workflow Modeling Language. Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems. 1998.

[Frankel 2004] Frankel, D. S.: An MDA Manifesto. MDA Journal. 2004. www.bptrends.com/publicationfiles/05-04%20COL%20IBM%20Manifesto%20-%20Frankel%20-3.pdf.

[Frankel et al. 2003] Frankel, D. S., P. Harmon, J. Mukerji, J. Odell, M. Owen, P. Rivitt, M. Rosen and R. M. Soley: The Zachman Framework and the OMG's Model Driven Architecture. MDA Journal. 2003. http://www.bptrends.com/publicationfiles/09%2D03%20WP%20Mapping%20MDA%20to%20Zachman%20Framework%2Epdf.

[Giraldo et al. 2008] Giraldo, W. J., A. I. Molina, C. A. Collazos, M. Ortega and M. A. Redondo: CIAT, A Model-Based Tool for designing Groupware User Interfaces using CIAM. Computer-Aided Design of User Interfaces VI, CADUI 2008. 2008. Albacete España. Springer Verlag.

[Giraldo et al. 2008] Giraldo, W. J., A. I. Molina, M. Ortega and C. A. Collazos: Una propuesta metodológica basada en taxonomías para el desarrollo de sistemas groupware interactivos. IX Congreso Internacional de Interacción Persona-Ordenador (INTERACCIÓN'2008). 2008. Albacete (España).

[Gutwin et al. 1998] Gutwin, C. and S. Greenberg: Design for Individuals, Design for Groups: Tradeoffs between power and workspace awareness. ACM CSCW'98. 1998. Seattle. ACM Press.

[Johansen 1988] Johansen, R.: Groupware: Computer support for business teams. 1988. New York: The Free Press.

[Jouault et al. 2006] Jouault, F. and I. Kurtev: On the architectural alignment of ATL and QVT. Proceedings of the 2006 ACM symposium on Applied computing. 2006. Dijon, France. ACM.

[Kaisler 2005] Kaisler, S. H.: Software Paradigms. 2005. John Wiley & Sons, Inc.

[Kelly et al. 2008] Kelly, S. and J.-P. Tolvanen: Domain-Specific Modeling Enabling Full Code Generation. 2008. New Jersey. John Wiley & Sons.

[Lu et al. 1999] Lu, S., C. Paris and K. Vander Linden: Towards the automatic generation of task models from object oriented diagrams. In Engineering for Human-Computer Interaction. 1999. Boston. Kluwer academic publishers.

[Miller et al. 2003] Miller, J. and J. Mukerji.: MDA Guide Version 1.0.1. Retrieve: 08-07-2007, from: http://www.omg.org/docs/omg/03-06-01.pdf.

[Molina et al. 2007] Molina, A. I., W. J. Giraldo, M. A. Redondo and M. Ortega: A proposal of integration of the GUI development of groupware applications into the Software Development Process. 13th International Workshop on Groupware (CRIWG'2007). 2007. Bariloche (Argentina). Lecture Notes in Computer Science. Springer-Verlag.

[Molina et al. 2006] Molina, A. I., M. A. Redondo and M. Ortega: A conceptual and methodological framework for modeling interactive groupware applications. 12th International Workshop on Groupware (CRIWG 2006). 2006. Valladolid. Spain. Springer-Verlag (LNCS).

[Molina et al. 2007] Molina, A. I., M. A. Redondo, M. Ortega and U. Hope: CIAM: A methodology for the development of groupware user interfaces. Journal of Universal Computer Science(JUCS). 2007. Designing the Human-Computer Interaction: Trends and Challenges.

[Moore et al. 2004] Moore, B., D. Dean, A. Gerber, G. Wagenknecht and P. Vanderheyden: Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. 2004. ibm.com/redbooks.

[Paternò 2001] Paternò, F.: Towards a UML for Interactive Systems. 8th International Conference on Engineering for Human-Computer Interaction. 2001.

[Paternò 2004] Paternò, F.: ConcurTaskTrees: An Engineered Notation for Task Models. The Handbook Of Task Analysis For HCI. 2004.

[Paternò et al. 1997] Paternò, F., C. Mancini and Meniconi.: ConcurTaskTree: A diagrammatic notation for specifying task models. IFIP TC 13 International Conference on Human-Computer Interaction Interact'97. 1997. Sydney. Kluwer Academic Publishers.

[Penichet et al. 2007] Penichet, V. M. R., I. Marin, J. A. Gallud, M. D. Lozano and R. Tesoriero: A Classification Method for CSCW Systems. Electronic Notes in Theoretical Computer Science. 2007. Elsevier Science Publishers B.

[Schwabe et al. 2001] Schwabe, D. and G. Rossi: Conference Review System in OOHDM. International Workshop on Web Oriented Software Technology. 2001.

[Sowa et al. 1992] Sowa, J. F. and J. A. Zachman: Extending and formalizing the framework for information systems architecture. IBM Syst. J. 1992.

[Trætteberg 2002] Trætteberg, H.: Using User Interface Models in Design. Computer-Aided Design of User Interfaces III, CADUI 2002. 2002. Valenciennes, France. Kluwer.

[Welie et al. 2003] Welie, M. v. and G. v. d. Veer: Groupware Task Analysis. Handbook Of Cognitive Task Design. 2003.

[Zachman 1987] Zachman, J. A.: A Framework For Information Systems Architecture. IBM Ssystems Journal. 1987.