Model and Formalism Driven Development of Collaborative Applications

Bertrand DAVID, René CHALON
LIESP Lab
Ecole Centrale de Lyon
69134 Ecully Cedex France
Tel. ++33 472 186581
Bertrand.David@ec-lyon.fr, Rene.Chalon@ec-lyon.fr

ABSTRACT

In this position paper, we explain our approach for collaborative systems development based on a model of cooperative applications and a formalism called ORCHESTRA allowing to express collaborative situations to take into account and a transformation process allowing to “project” ORCHESTRA description on different execution plate-forms elaborated in respect with a generic architecture.

Categories and Subject Descriptors

D.2.1 [Software Engineering]: Requirements/Specifications – Methodologies, D.2.2 [Software Engineering]: Design Tools and Techniques – User interfaces. D.2.11 [Software Engineering]: Domain-specific architectures, description languages H.1.2 [Models and Principles]: User/Machine Systems – Human factors H.5.1 [Multimedia Information Systems] Artificial, Augmented and virtual realities, H.5.2 [User Interfaces]: User-Centered Design, H.5.3 [Group and Organization Interfaces] Computer-Supported Cooperative Work

General Terms

Management, Design, Experimentation, Human Factors.

Keywords

CSCW, Specific description language, MDA inspired elaboration process, transformation process, formalism meta-model.

1. INTRODUCTION

CSCW [1] is a field of interactive computer-based systems which objective is to allow several participants (actors) to work together via a computer-based system to complete cooperatively a task which can be of different natures (design, management, production, learning, etc). Design of this kind of systems is relatively complex because it is not limited to individual activities, but also and mainly to cooperative work of several actors, which can be classified in co-operation, coordination and conversation activities in respect with the definition initially proposed by Ellis [10] and adapted by several other authors [9]. This cooperative work can be done in several cooperative situations characterized initially by Johansen and enhanced by Ellis [11]. At the moment CSCW systems are becoming more and more mobile, context-aware and proactive. We called this kind of cooperative systems Capillary Cooperative Systems (CCS) [7]. We use this term by analogy with the network of blood vessels. The purpose of the Capillary CS is “to extend the capacities provided by co-operative working tools in increasingly fine ramifications, hence they can use fixed workstations and handheld devices". These systems become also pervasive, proactive and ubiquitous. Our final goal is to allow them to evolve in mixed reality environment (mixture of real and digital objects and tools) and to put into practice Ambient Intelligence (AmI) concepts.

2. OUR APPROACH

We are studying design of CSCW systems and we propose an approach and a process, called CoCSys (Collaborative Capillary System) engineering process [8]. Main reason for this more comprehensive process is related to the necessity to collect requirements of the system as appropriate as possible and allow the evolution of this kind of system during its use in relation with the users’ skills, expertise, and the evolution of their perception and the mastery of the system. Our approach is based on Model-Based approach [15], which is characterized by a different way of development: “Rather than programming an interface using a toolkit library, developers would write a specification of the interface in a specialized, high-level specification language. This specification would be automatically translated into an executable program, or interpreted at run-time to generate the appropriate interface.” This approach is used in HCI for several years and become more generally used in other development application fields. OMG adapted a similar approach as new paradigm of development which is called MDA Model-driven architecture [13]. Other acronyms describing similar ways are MDE (Model-Driven Engineering) or MDD (Model Driven Development). In each case specification at concrete, abstract or meta level is privileged before studying the way to produce an executable code. The production is done more or less automatically by transformation or translation of these models. The objective of our approach is to adapt this trend to CSCW. We are proposing a framework for design, implementation and evolution of CCS. As described deeply in [8] this approach is based on 3 main parts: 1/Scenarios Collection, 2/Cooperative Behavior Model (CBM), and 3/Collaborative Architecture; and 3 transformation phases: I/CBM Model Construction, II/CBM Projection on the Collaborative Architecture and III/Evolution.

2.1 Cooperative Behavior Model

We consider that a scenario based approach allows to final users and designers to meet them and discuss together about functionalities of the system to be developed. A scenario describes repetitive activity that should activate an adaptation mechanism which will be recorded and reused. This analytical perception of working situations seems to be possible to catch and to express observers or actors needs. We are asking to give as precise description as possible, i.e. to indicate, if possible, all actors evolving, artifacts used, activities executed and contexts characterizing them (devices used, geographical location, temporal situation …). The designers are in charge to study different scenarios and to construct gradually the Cooperative Behaviour Model (CBM). In the model we find comprehensive collections of actors, artifacts, activities and contexts and also all relations which allow materializing all necessary elements for each activity. Different processes are also explained carrying out dependencies between tasks and their temporal and organizational constraints. This comprehensive model is able to manage the cooperative system behavior and will be used during the implementation process i.e. projection of this model on a particular hardware, network and software architectures. Main elements of the CBM model are:

An **actor**, as instantiation of one or several roles, a role is a basic element of human behavior in the system, which can be qualified as Acting (A), Observing (O) or Editing (E) i.e. observing and acting.

An **activity**, describing an identified work which a role can do, this activity can be also A, O or E, i.e. acting, observing or editing activity.

A **process** expressed as a network composed of process states (PS) and process transitions, which can also be qualified by A, O or E.

An **artifact** can be either a **tool** or an **object**. The tool is an instrument used in the task; the object is either input, support or output of the task, qualified by A, O or E.

A **context** is a collection of three aspects: platform, situation (often logical, physical or geographical location) and user preferences characterizing the context. We take into account several platform examples and elements: laptop, PDA, cellular phone, and also active environmental object (active RFID tag), passive environmental object (passive tag).

In the CBM model all these elements are expressed and interconnected. We can take as example a user's role, which is identified by a name, a type, its participation in different actors, the activities which can be done, the process states and transitions in which their can occur, the artifacts (tools and objects) manipulated and the contexts (platform, situations and user preferences) which applies the role. These interrelations are also needed for other elements of the model. They are explicitly or implicitly described and can change during the system life expressing its adaptation and evolution. List of activities is one of the main components of CBM. This list is obtained from the task tree which can be expressed by CTT [14], an interesting task formalism, and its environment (CTTE) proposed by Paterno. Its extension for cooperative activities [12] aims to express cooperative situations. In CTT, collaboration is expressed by individual task trees and by a collaborative task tree. That is interesting to express tasks, but is insufficient for the more comprehensive view of collaboration, that we need. We consider that tree view of tasks is interesting during the task design phase. However, during the activities organization (definition of effective collaborations), mainly effective activities (leaves of the task tree) are important and their individual or collaborative scope is essential, in relation with effective actors, objects, tools, process states and transitions and contexts. To express this more comprehensive view we propose a formalism called Orchestra.

2.2 ORCHESTRA

The objective of ORCHESTRA [6] is to propose a more comprehensive formalism which is able to express together all main aspects of the CBM. ORCHESTRA adapts musical score notation [16] to our problem of CBM description. For us, the 5 lines of a staff are expressing 5 main aspects of the CBM (Fig. 1), which are: user's role, activity concerned, process state or transition, artifacts involved in the activity and the context. These aspects are expressed on each of their respective line by situating one or several "notes" containing their names. Each note can receive a stem which indicates the participation of the element (acting, observing or editing). We distinguish main actor (double arrow) and secondary actor (simple arrow) as well as active role and passive role: A bar line indicates the separation between independent cooperation episodes. To express repetition of an episode we propose four options: an explicit number of repetitions (n), an undetermined number of iterations (+/*), a contextual end (logical condition), a time dependent end of iteration (relative or absolute time limit). Each cooperation episode expresses a state or a transition in the cooperation process description network. For each cooperation episode, sequential ordering from left to right is implicit temporal option, another order, must be expressed explicitly either by a jump from current period to another one which is named, or by a "procedure call" jump to a named episode then the back to the previous one.

By different types of parenthesis, we indicate explicit relations between participating notes. These parentheses are used to express different situations:

(…) alternatives,
{…} mandatory participation,
[….] optional participation.

Different key signatures or annotations are expressing collaboration properties like synchronous or asynchronous collaborations, collaboration modes and styles of coordination (computational 💻 or social 😐, implicit ● or explicit ▬):

@ - **Asynchronous** with infinite answer delay
@@ - Asynchronous with limited answer delay corresponding to "**on call**" participation
& - Synchronous "**in-meeting**" cooperation
&& - Synchronous "**in-depth**" cooperation

In synchronous collaboration two different participations must be distinguished:

- **instantaneous**, short term collaboration, called also implicit and expressed by ● i.e. vote activity,
- **long term** participation, long term collaboration, called also explicit and expressed by ▬ i.e. sketching activity.

In the first case (vote activity) an implicit collaboration is appropriate (short exclusive access to the shared space), in the second case (sketching) explicit participation must be asked and allowed (long-term access to the shared space) either by **social coordination** (😐), i.e. one of human actors is in charge of this coordination or a **computational** (💻) one i.e. the computer fulfil it. We express graphically instantaneous collaboration by a **dot** over concerned chords and for long term collaboration we use a horizontal line ▬ and a symbol expressing social or computational coordination (😐, 💻) i.e. coordination made by one of the actors or by interaction (asking for, receiving and returning exclusive access right to shared space).

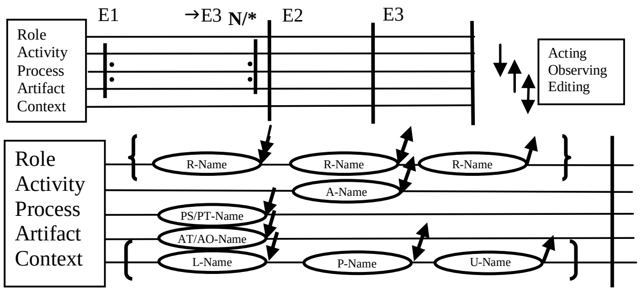
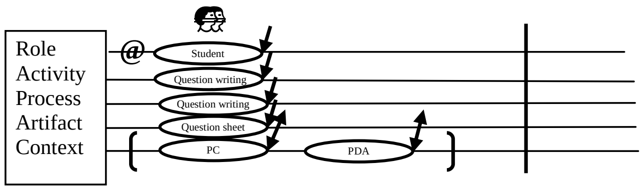


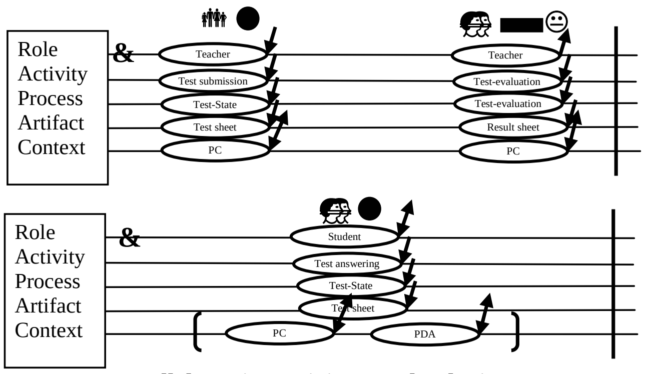
Fig. 1. ORCHESTRA main concepts

Another important notion in CSCW is **awareness**. Its objective is to allow to different actors to know (or not) what has been done by an actor. It is important to decide statically (by the designer) or dynamically by the actor himself the scope of information propagation to other actors. For static way we propose to express awareness in ORCHESTRA formalism. Special marks are proposed:

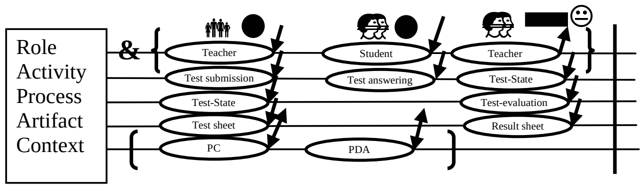
- for **no awareness**,
- for **partial awareness** (for specific actors),
- for **overall awareness** (for all actors).



A – Individual activity: question to teacher writing



B – Collaborative activity - In-depth view: test preparation, submission and evaluation



C - Collaborative activity - Synthetic view
Fig. 2. Different ORCHESTRA descriptions: individual, cooperative, in-depth or synthetic views

In fig. 2 we show an example of description. To explain more deeply ORCHESTRA formalism, we give in annex 1 its metamodel.

3. Transformational process

From ORCHESTRA the target of our process is a generic software architecture composed of three layers decomposition as a generic framework for cooperative mobile pervasive systems. The top layer corresponds to the collaborative application level. It contains all the cooperative software used by the actors. This level is totally user-oriented, which means that it manages interaction control and proposes interfaces for notification and access controls. It uses multi-user services provided by a second layer. This one is a generic layer located between application layer and the distributed system layer. This layer contains common reusable elements of groupware activities and acts as an operating system dedicated to groups. It supports collaborative work by managing sessions and users, provides generic cooperative tools (e.g. telepointer) and is responsible for concurrency control. It also implements notification protocols and provides access control mechanisms. The last layer is essentially in charge of message multicast and consistency control. Usually, it is a computer-oriented layer which provides transparent mechanisms for communication and synchronization of distributed components.
We developed a cooperative middleware called SMAC (Services for Mobile Applications and Collaborations) that implements the two lower layers (groupware services and distributed system) of this conceptual cooperative architecture (see Fig.3).

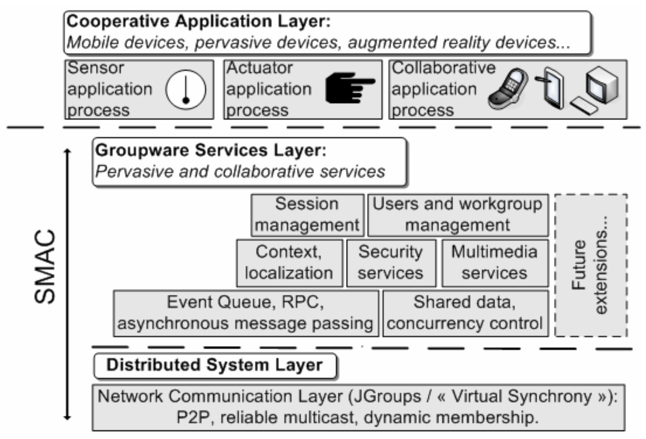

Cooperative Application Layer:
Mobile devices, pervasive devices, augmented reality devices...
Sensor application process
Actuator application process
Collaborative application process
SMAC
Groupware Services Layer:
Pervasive and collaborative services
Session management
Users and workgroup management
Context, localization
Security services
Multimedia services
Event Queue, RPC, asynchronous message passing
Shared data, concurrency control
Future extensions...
Distributed System Layer
Network Communication Layer (JGroups / « Virtual Synchrony »): P2P, reliable multicast, dynamic membership.

Fig. 3. Three layer collaborative architecture structure.

As we target mobile devices, we have strong constraints for the choice of technology for the Distributed System Layer. On one hand, synchronous cooperation is hard to implement with lightweight clients, and on the other hand heavy distributed objects systems such as Corba or J2EE are not available on mobile devices. Currently we choose to base SMAC on the Virtual Synchrony distributed programming model [3, 5], by using a version of JGroups [2] specially implemented for the J2ME / CDC Java Virtual Machine. Although the Virtual Synchrony programming model does have some limitations regarding specific mobile CSCW scenarios (mainly: it does not scale well to a large number of concurrent users, and it is not very adapted to situations involving lots of connections and disconnections), it does fit well with the kind of scenarios that we are experimenting, and it provides convenient and powerful abstractions of cooperating processes that need to keep coherent states. Above this layer, the Groupware Services Layer is composed of a core framework of Java classes onto which we can plug specific groupware services as needed. Currently, only a subset of these services is implemented, mainly the classes corresponding to the notions of collaboration, cooperation episode and session, and the classes representing users and groups. This provides a minimal system that handles synchronous collaboration, as well as persistence of collaboration states between sessions. The relation between ORCHESTRA and the generic architecture is the following: Information coming from the ORCHESTRA description concerning roles, activities, process, artefacts and context is “projected” on this architecture. This projection concerns either application layer or collaborative layer, whose core classes are summarized in annex 2. Information about role and actors is manipulated at the collaborative layer, as well as at the application layer, where the corresponding user interface is proposed. ORCHESTRA concept of activity is translated to SMAC in two different ways. An application specific activity, called semantic activity, is located at the application layer, for generic activity its location is naturally at the collaborative layer. Concerning cooperation processes management expressed in ORCHESTRA by episodes and their orchestration, their corresponding SMAC classes are using an adaptive workflow engine. ORCHESTRA artefacts are either tools or objects, generic or semantic. Their mapping to SMAC is done either at application layer (for semantic artefacts) or at collaborative layer for generic ones. Tools are used or activated at application layer and objects are manipulated by services located either at application layer or at collaboration layer depending of their specificity or genericity. Context description expressed by ORCHESTRA is used at physical level concerning hardware platform description, at distribution layer concerning software level description and either at collaborative layer or application layer concerning location adjustment and user preferences. Main mechanisms used during this transformation are XML encoding and decoding of information manipulated in ORCHESTRA editor and interpretation engine, which is able to read these XML files and execute appropriate code either generated from this description or corresponding attachments doing the link with existing code at collaboration layer or specifically developed code at application layer. According to platform adaptation mechanisms we are able to produce appropriate interfaces in regard with hardware platform used i.e. laptop, PDA or Smartphone.

4. CONCLUSION

In this paper, we outlined a new formalism called ORCHESTRA, which objective is to provide a graphical expression of Cooperative Behavior Model. CBM, elaborated from a collection of scenarios, as a reference for the transformation process allowing different implementations. As it is important to associate different actors to this constructive process, we propose a formalism which could be used during initial discussions as well as during the implementation and adaptation process. We elaborated a set of reusable patterns which are useful to accelerate and do design process more powerful. We propose to use them in a pattern oriented walkthrough, in which patterns are considered as best practices, as a collection constituting an inspiration sourcebook and as a use guide. Of course ORCHESTRA explains a global view of cooperation. An in-depth view is necessary to describe completely the content of “notes” with the help of an editor. ORCHESTRA has been tested in several case studies and we may continue to upgrade it by new concepts as result of these tests. The connection with mixed reality has not been described in this paper, even if we are currently working on it [4].

5. REFERENCES

[1] Andriessen, J.H.E.: Working with Groupware: Understanding and Evaluating Collaboration Technology. Springer, CSCW Series (2003) 206 p.

[2] Ban B. Design and Implementation of a Reliable Group Communication Toolkit for Java. Cornell University (1998)

[3] Birman K. Reliable Distributed Systems: Technologies, Web Services, and Applications. Springer (2005)

[4] Chalon, R., David, B.: IRVO: an Interaction Model for designing Collaborative Mixed Reality Systems, HCI International 2005, Las Vegas, USA, 22-27 July (2005)

[5] Chockler G.V., Keidar I. and Vitenberg R.. Group Communication Specifications: A Comprehensive Study. ACM Computing Surveys, 4(33):1-43, December (2001)

[6] David, B., Chalon, R., Delotte, O., Masserey, G., Imbert, M.: ORCHESTRA: formalism to express mobile cooperative applications. LNCS, Vol. 4154, Springer, (2006) 163-178

[7] David, B., Chalon, R., Vaisman, G., Delotte, O.: Capillary CSCW. In Proceedings of HCI International, Crète (2003)

[8] David, B., Delotte, O., Chalon, R.: Model-Driven Engineering of Cooperative Systems. In proceedings of HCI International 2005, Las Vegas, USA, 22-27 July (2005)

[9] David, B., IHM pour les collecticiels. In Réseaux et Systèmes Répartis, Hermès, Paris, vol. 13 (2001) 169–206

[10] Ellis, C., Gibbs, S.J., Rein, G.L.: Groupware: some issues and experiences. In Communications of the ACM, vol. 34, n° 1, (1991) 38–58

[11] Ellis, C., Wainer, J.: A conceptual model of Groupware, In Proceedings of CSCW'94, ACM Press, (1994) 79–88

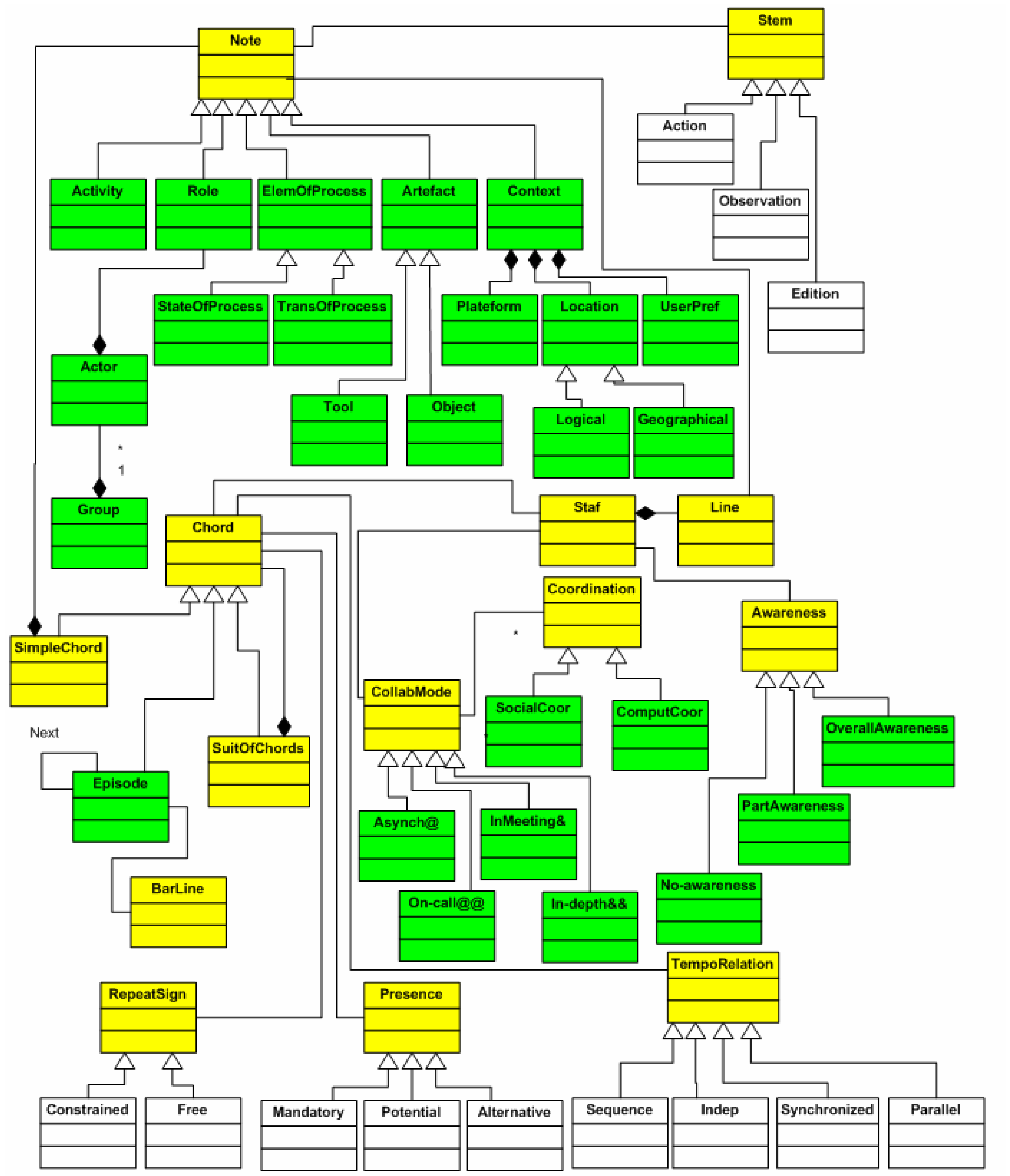
[12] Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. In IEEE Transactions on SE, vol. 28, n. 9.

[13] Object Management Group, http://www.omg.org/mda/

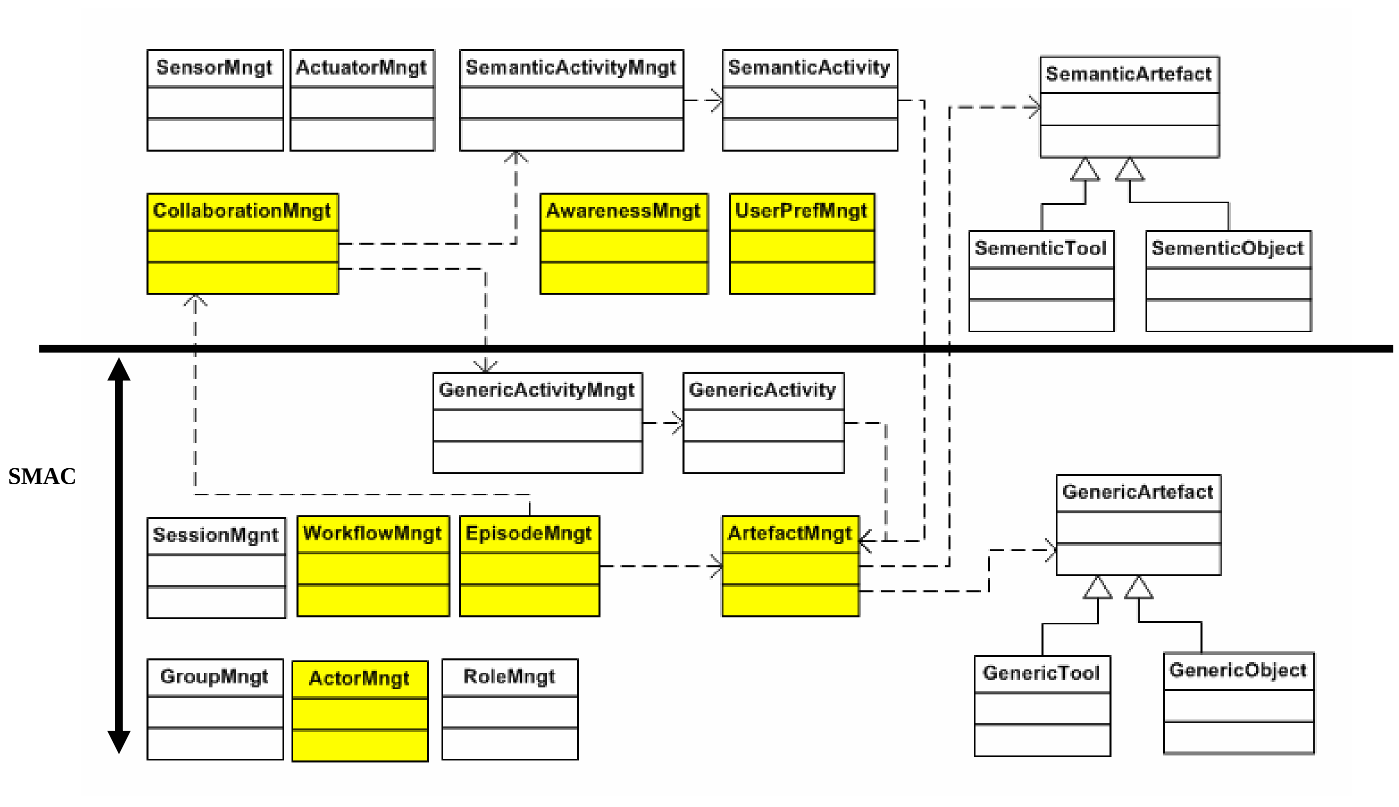
[14] Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Applied Computing Series, Springer -Verlag (2000)

[15] Szekely, P.: Retrospective and Challenges for Model-Based Interface Development. In: Vanderdonckt, J. (eds): CADUI'96, 5-7 June 1996, Namur (1996)

[16] Stewart, D., The Musician’s Guide to Reading and Writing Music. Backbeat (1999) 117 p.



Annex 1: ORCHESTRA metamodel. Main ORCHESTRA classes are in yellow (light gray in black and white) and CBM classes are in green (dark gray in black and white).



Annex 2. SMAC and Cooperative Application Layer core classes.