Modelling Regulated Social Spaces for Groupware Applications

Carmen Mezura-Godoy and Luis Gerardo Montané-Jiménez

University of Veracruz,
Av. Xalapa, Esq. Manuel Avila Camacho,
CP 91020, Xalapa Ver.
cmezura@uv.mx
lmontane@uv.mx

Abstract. Social regulation is an intrinsic aspect of collaborative work, although it is not considered in the most of cases in groupware applications. Regulation concerns the establishment of working rules and their negotiation. The MARS regulation model enables users to define their common workspace -arena, actors participating in the activity, roles assigned to the actors, interactions between actors and rules governing all the objects in the arena and also in several ones. This paper proposes a service to regulate groupware applications. The service implements a language to describe social aspects. In order to validate our approach, a prototype supporting the MARS model has been implemented.

1 Introduction

In Sociology, *social regulation* is the process which enables groups to create and modify social rules controlling their individual and collective actions[5]. Regulation enables people to describe how they wish to take part in the activity and the minimal conditions of work's execution: in other words, it enables people to create, negotiate and apply rules controlling their collaborative activity. Regulation does not imply a complete description of an activity, but rather a definition of minimal rules, and personal rights and responsibilities, in order to improve the group activity. Regulation inside a group changes during the activity. The rules described at the begining of the activity can be modified like a natural process of evolution of the activity; so, the work rules change along with the activity.

Activities group can be for instance: (a) communication with each other in order to exchange their ideas or points of view, (b) sharing information or their workspace and (c) coordinationof their activity (time, space and resources). Nevetheless, we can observe that people collaborate in several activities at the same time. For instance, members of a research team take part in meetings, in projects, in paper writing, etc., so people take part in several arenas.

Nowadays, groupware tools supporting collaborative work enable users to communicate, coordinate and cooperate in specific tasks [17, 8, 2, 13]. Nevertheless, these tools rarely incorporate social activity aspects. Moreover, they do not enable users to regulate their activity.

Received 02/07/08
Accepted 01/08/08
Final version 11/09/08

We consider important to include a model activity in groupware applications. So, it was proposed a new multi-arena regulation model [15, 14]. This model enables to describe activities in a single workspace and also in several ones.

On this paper we propose a regulation service for groupware tools. It enables developers of groupware applications to create new regulated applications from simple ones. This regulation service enables: to model several arenas and to execute them according to the regulation. In order to validate our model we implemented a software prototype. The prototype illustrates how a groupware application can be regulated, how end-users can describe and modify the regulation during the activity and how the service applies the regulation.

The rest of the paper is organized as follows. Section 2 surveys related work to establish the context of our research. Section 3 describes the MARS regulation model and its associated language. Section 4 presents a regulation service for groupware applications. Section 5 introduces our experimental software prototype. Finally section 6 summarizes the contributions of this paper and introduce some research directions.

2 Related Work

Differents works offer infrastructures allowing the introduction of some social aspects in groupware tools, under the term "coordination policies" *e.g.* [7, 4, 11, 12, 16, 10]. These policies enable to coordinate an activity in terms of access control to the production groupware space (private or shared workspace).

Even though, these systems offer the means to incorporate social aspects of an activity work (participants, roles, policies, rights). However, social aspects are defined at a low level of abstraction (coordination language). In order to enable users to define their activity, we believe that it is necessary to provide them with more powerful mechanisms to facilitate the definition of social aspects of collaborative activities.

Moreover, the models proposed by these systems focus only coordination aspects, rather than on the complete collaborative activity. For instance, the role concept is used to control access to shared resources (production space). For us, role is a collaboration concept: an actor in the activity has a particular role in the group (*e.g.* leader, coordinator, etc.).

In the works previously cited social aspects are limited to activity coordination. We believe that a model for collaborative activities must allow the definition of the activity and its context, which means enabling users to define group members taking part in the activity (their duties, rights and preferences), their social role, tools to facilitate doing their activity, and naturally groupwork rules. Including a complete model of the activity in groupware can aid end-users to use them better and adopt them more easily.

Works like, Locales [6], CooLDA [1], SeeMe [9] as well as participation model (PM) [3] propose activity models for groupware applications. These models are founded in works in social sciences and they offer concepts enabling users to define a group activity in a workspace. This workspace called, a "locale"(Worlds), "support of activity"(CooLDA) or "arena"(PM), represents the group's activity and its context, *i.e* it has the activity components and it establishes necessary conditions for the activity execution. On the other hand the purpose of SeeMe is to support the early phases of developing concepts for socio-technical solutions and to document them with diagrams.

All these works take into account the evolutionary aspect of a group activity. Worlds and CooLDA are based on reflexive models to enable users to modify the activity model in runtime. CooLDA and PM consider redefinition of the activity as part of the activity itself.

An activity model must also make it possible to improve the design of groupware applications and, consequently allow users to better use them. Nevertheless, Worlds and CooLDA do not consider social aspects of activities, like people engagement in the activity and social work rules as PM does. Even though, the PM offers a social regulation model of collaborative activity, none of them enables the idea of "reusing" defined spaces in order to create more complex collaborative spaces.

3 The MARS regulation model and associated language

In this section we present, the MARS *Multi-Arena Regulation model* [15]. This model allows one to represent regulated group activities supported by groupware tools. These regulated group activities can be carried out by members of a group inside a collaborative space or in several ones.

3.1 Elementary concepts

A group activity is defined by interactions taking place in a collaborative space called *arena*. The users executing interactions are called actors. An *actor* represents a person, a software agent or a group. Throughout an activity, the actors handle and produce *objects*, such as documents, files, notes, etc. A family groups actors or objects having the same set of features (*e.g.* "writers", "readers", "books", or "papers" families). During the activity, actors and objects plays different *roles*, depending on the specific interaction they execute or take part. For instance during the "writing" interaction an actor plays the "writer" role, and the object handled in this interaction plays the "draft" role.

In order to regulate their activity, actors define scenarios for each interaction. A *scenario* describes how an interaction is carried out (operations or interactions that must be executed), who can participate in the interaction, and what objects can be handle. The scenarios represent the social protocols taking place in the arena.

In the following let $\mathcal{A}$, $\mathcal{O}$, $\mathcal{R}$, $\mathcal{S}$, $\mathcal{A}_\mathcal{F}$ and $\mathcal{O}_\mathcal{F}$, be the sets of *actors*, *objects*, *roles*, *scenarios*, *actors family* and *objects family* respectively, and α and β two functions returning respectively the family of an actor or an object.

3.2 Interaction, scenario and arena

An "interaction model" defines a regulated interaction inside the arena. It specifies all the families of actors and objects taking part in the interaction, roles attributable to actors and objects during the interaction, and the scenarios describing how the interaction can be carry out.

Definition 1 (Interaction model.)
An interaction model is a tuple $< n_I, E, A_f, O_f, R_s, S_s, \pi, \rho >$, *where* n_I *is the name of the interaction,* E *is a set of interaction states,* $A_f \subseteq \mathcal{A}_{\mathcal{F}}$, $O_f \subseteq \mathcal{O}_{\mathcal{F}}$, $R_s \subseteq \mathcal{R}$, $S_s \subseteq \mathcal{S}$, π *is a relation from* $A_f \longrightarrow R_s$ *and* ρ *is a relation from* $O_f \longrightarrow R_s$
$\square$

Let us imagine the model for the interaction "to publish some document" defined as follows: $<$ *to Publish*, {*active, finished*}, {*manager, collaborators*}, {*paper, document, image*}, {*publisher, published*}, {*scenario To Publish*}, {(*manager publisher*), (*collaborator, publisher*)}, {(*paper, published*)} $>$[1]. This model authorize "to Publish" interaction to "manager" and "collaborators", it limits the objects handled in this interaction to "papers", "images" and "documents", it defines the "publisher" role assigned to the "manager" and "collaborators" and the "published" role assigned to "papers", "images" and "documents". It specifies the "scenario To Publish" as the only scenario describing how this interaction can be carried out.

An "interaction", represents an interaction in execution. An interaction must always be according to an interaction model.

Definition 2 (Interaction.)
Given an interaction model $< n_I, E, A_f, O_f, R_s, S_s, \pi, \rho >$, *an interaction is a tuple* $< n_I, e, A, O, s, \sigma, \omega >$, *where,* n_I *is the name of the interaction,* $e \in E$, $A \subseteq \mathcal{A}$ *and* $\forall a \in A\alpha(a) \in A_f$, $O \subseteq \mathcal{O}$ *and* $\forall o \in O\beta(o) \in O_f$, $s \in S_s$, σ *is a relation from* $A \longrightarrow R_s$, *and* ω *is a relation from* $O \longrightarrow R_s$. $\square$

An interaction representing the actor "carmen" publishing the "enc08" paper in the "writing" arena could be the following: $<$ $toPublish_1$, *active*, *carmen*, *enc08*, $scenarioToPublish_1$,(*carmen, publisher*), (*enc08, published*) $>$.

A "scenario" describes how an interaction can be executed.

Definition 3 (Scenario.)
A scenario is a tuple $< n_S, Pre, Pos, S_s >$, *where* n_S *is the name of the scenario,* Pre *is a set of preconditions,* Pos *is a set of posconditions and* $S_s \subseteq \mathcal{S}$. $\square$

A scenario for the interaction "to publish a paper" is defined as follows: $<$ *scenarioToPublish*, {(*paper, finished*), (*dateOfPublication*$<$ *deadline*)}, {(*paper, published*)} $>$. This scenario defines two preconditions. The first one evaluates the role of the paper, in this case it must be "finished" in order to be "published", and the second one evaluates the deadline. Finally, the poscondition estiblishes for the paper the role of "published".

[1] We identify in all our examples, actors, objects, roles, scenarios, interaction's models and interaction instances by their name.

An arena defines a group activity, actors, objects, interaction model and interactions.

Definition 4 (Arena.)
An arena is a tuple $< n_E, A_s, O_s, M_s, I_s >$, *where* n_E *is the name of the arena,* $A_s \in \mathcal{A}$, $O_s \in \mathcal{O}$, M_s *is a set of interaction models and* I_s *is a set of interactions.*
□

An example of a writing arena is the following: $<$ *writingArena*, {*carmen*, *luis*}, {*enc*08, *image*1, *cscw*07}, {*toPublishIntM*, *toWriteIntM*}, {$toPublish_1$, $toWrite_2$} $>$ where *writingArena* is the name of the arena, "carmen" and "luis" are the actors who can perform interactions in this arena, "enc08", "image1" and "cscw07" are tha accesibles documents (for publishing or writing interactions), "toPublishIntM" and "toWriteIntM" are the interaction models, the first one for publishing interaction and the second one for writing interaction, and finally "$toPublish_1$" and "$toWrite_2$" are the running interactions.

3.3 View and Complex Arena

A collaborative activity is defined inside an arena. Nevertheless, members of a work group collabortate with other groups or activities. So, people take part in several activities in different spaces, *i.e.* several arenas. For instance the members of a research team collaborate at the same time on projects, on writing articles or documents, on the organization of conferences or work meetings, etc. Each of these collaborative spaces is controlled by specific work rules.

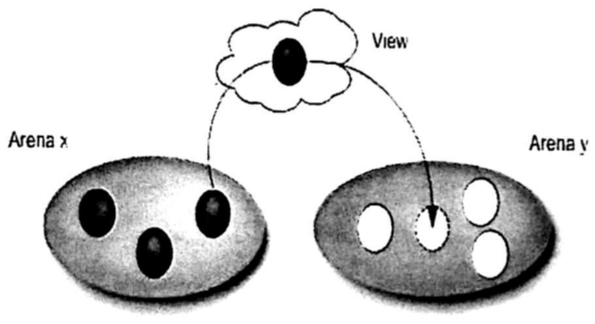


Fig. 1. Cooperation between arenas via the use of views.

A "view" defines, actors, objects and interactions that an arena can share with another (See Fig. 1).

Definition 5 (View.)
Given an arena $< n_E, A_s, O_s, M_s, I_s >$, *a view is a tuple* $< n_E, A_v, O_v, m_v >$, *where* n_E *is the name of the arena that produce the view,* $A_v \subseteq A_s$, $O_v \subseteq O_s$ *and* $m_v \subseteq M_s$. □

Let us imagine, that two arenas will cooperate to make their objects accessible to each other. The "libraryArena" produces a view with the objects it will share with the "writingArena". The view defined by libraryArena is: $< \{edgard, carmen, luis\}$, $\{LNCS2527,\ javaBeans,\ groupwareApplications\}$, $toBorrowIntM >$, where "toBorrowIntM" is the interaction that will be accessible from the writeArena, "edgard", "carmen" and "luis" are the actors with the possibility of borrowing a book from the libraryArena, "LNCS2527", "javaBeans" and "groupwareApplications" are the books from libraryArena accesible from the "writngArena".

Arenas importing remote objects from other arenas are called "complex arenas".

Definition 6 (Complex arena.)
Given an arena $< n_X, A_s, O_s, M_s, I_s >$ *and a view* $< n_Y, A_v, O_v, m_v >$, *a complex arena is a tuple* $< n_X, A_s \bigcup A_v, O_s \bigcup O_v, M_s \bigcup m_v, I_s >$. □

The "writingArena" is an example of a complex arena because it imported remote objects from the "libraryArena": $< writingArena, \{carmen, luis, edgard\}$, $\{enc08,\ cscw07,\ image1,\ image2,\ LNCS2527,\ javaBeans,\ groupwareApplication\}$, $\{toPublishIntM, toWriteIntM\}$, $\{toPublish_1, toWrite_2\} >$.

3.4 Arena and view operators

The arenas evolve according to the creation and the execution of interactions. For this reason, we defined operators which allow to manage arena and view objects. Each of these operators ensures the passage of an arena or a view from one coherent state to another, always respecting the arena regulation.

Table 1. Operators enabling the addition and deletion of actors, objects, interactions and model interactions to/from arenas.

Operator	Enter	Exit
addActor	(E, a)	$E' = (n, A \bigcup \{a\}, O, I, M)$
deleteActor	(E, a)	$E' = (n, A - \{a\}, O, I, M)$
addRobject	(E, o)	$E' = (n, A, O \bigcup \{o\}, I, M)$
deleteRobject	(E, o)	$E' = (n, A, O - \{o\}, I, M)$
addModelInt	(E, m)	$E' = n_E, A, O, I, M \bigcup \{m\})$
deleteModelInt	(E, m)	$E' = (n_E, A, O, I, M - \{m\})$
addInteraction	(E, i)	$E' = (n_E, A, O, I \bigcup i, M)$
addInteraction	(E, i)	$E' = (n_E, A, O, I - \{i\}, M)$

Table 1 summarizes the operations to allow adding and removing actors, objects, interaction models and interactions to/from an arena. For instance, for a given arena $E =< n, A, O, I >$ where: n, is the identifier of the arena, "A" is the set of actors, "O" the set of objects, "I" the set interactions and "M" the set of model interaction, the arena resulting after applying the operator "addActor(E,a)", is $E' =< n, A \bigcup \{a\}, O, I, M)$. The operation "addActor" verifies that the family of the actor to be added is defined in the arena.

We defined two operations to enable to cooperate arenas. These operators allow them to share their objects by export and import views.

Operator 1 (Export View.)
Given an arena $E =< n, A, O, I, M >$, a set $A_v \subseteq A$, a set $O_v \subseteq O$ and a set $M_v \subseteq M$, the result of $ExportView(n, A_v, O_v, M_v)$, is the following view $V =< n, A_v, O_v, M_v >$. □

Table 2. Operators enabling the addition and deletion of actors, objects and model interactions to/from views.

Operator	Enter	Exit
addViewActor	(V, a)	$V' = (n, A \bigcup \{a\}, O)$
deleteViewActor	(V, a)	$V' = (n, A - \{a\}, O)$
addViewRobject	(V, o)	$V' = (n, A, O \bigcup \{o\})$
deleteViewRobject	(V, o)	$V' = (n, A, O - \{o\})$
addViewModelInt	(V, m)	$V_M = (n, A, O, M_M \bigcup \{m\})$
deleteViewModelInt	(V, m)	$V_M = (n, A, O, M_M - \{m\})$

Table 2 summarizes the operations to allow adding and removing actors, objects and interaction models to/from a view. For instance, for a given view $V =< n, A, O, M >$, where: "n" is the identifier of the view, "A" is the set of actors to be exported, "o" is the set of objects and "M" is the set of interaction models, the view resulting after applying the operator "addViewActor(V,a)" is $V' =< n, A \bigcup \{a\}, O, M) >$

Operator 2 (Import View.)
Given an arena $E =< n_x, A, O, I, M >$, and a view $V = (n_y, A_v, O_v, M_v)$, the result of $ImportView(E, V)$ is an arena $E' =< n_x, A \bigcup A_v, O \bigcup O_v, I, M \bigcup M_v >$.
□

Consider the following "writing arena": $writingArena =< \{carmen, luis\}$, $\{enc08paper\}$, $\{toWriteMInt, toReviewMInt\}$, $\{toWrite_1, toWrite_2\} >$. This arena defines an space for joint paper's writing. Inside this arena, "carmen" and "luis" can "write" and "review" a paper for the Enc2008. Now, we want to enable these actors to access the univertisty library, in order to ease the paper's documentation.

For that, the "libraryArena" exports a view (with actors, objects and interaction) to the "writing arena", as the following: $ExportView =< libraryArena$, $\{carmen, luis, edgard\}$, $\{cscw02, LNCS2517, criwg05\}$, $toBorrowIntMt$ $>$. The operation $ImportView$ adds to the "writing arena" the objects of the library's view. The new "writing arena" is: $<$ $writingArena$, $\{carmen, luis$ $edgard\}$, $\{enc08paper, cscw02, LNSC2527, criwg05\}$, $\{toWriteMInt, toReview$ $MInt, toBorrowMInt\}$, $\{toWrite_1, toWrite_2\}$ $>$.

As we can see, in a multi-arena context, arenas have "local" and "remote" objects. So, in order to enable arenas handling with remote objects a propagation of operators is then necessary. We identified several constraints that must be satisfied during the application of the operators in order to ensure coherence between arenas [14].

3.5 Collaborative Regulation Language: CoRaL

In order to describe scenarios with the MARS model a language named CoRaL was proposed. Remember that a scenario defines how an interaction can be execute, it specifies through pre and pos conditions: i) who can participate in the interaction, ii) what objetcs can be manipulated, iii) what role has an actor or object during the interaction and in same cases iv) has references to another scenarios.

Let be E, the set of all strings representing names of: arenas, actor, families of actors and objects, roles and interactions. In the following we describe the syntax of CoRaL lenguage in BNF notation.

<statement> ::= <expression> <operator> <expession> ";"
<expression>::= <keyword> ":{"<elements> "}"
<expression>::= ! <keyword> ":{"<elements> "}"
<operator>::= :: | ->
<keyword>::= "Arena" | "Interaction" | "Actor" | "Actor family" |
"Object family" | "Object" | "Role"
<element>::= any string on E

In CoRaL a "statement" describes pre and pos conditions. A "statement" is formed by two expressions and an operator. The operator "::" is used to represent preconditions and the operator "->" is used to represent posconditions. An "expresion" is formed by a "keyword" and "element", where the "keyword" is a string identifing a component of the model (*e.g.* Arena, Interaction or Actor) and the "element" is a string representing an identifier of E elements (*e.g.* "to Publish" or "carmen").

For instance, the precondition *Actor:{"carmen"} ::Arena:{"writingArena"}*, states that carmen must be defined as an actor in the arena, in order to execute an interaction inside the "writingArena" arena.

4 Regulation service

In order to regulate a groupware aplication we introduce a regulation service based on the generic regulated architecture proposed on [14]. This architecture proposes the construction of a regulated collaborative application from two components: the component application and the regulation's component (See Fig. 2). At the application layer we observe the functions of the groupware applications on regulated way and at the regulation layer are described the collaboration spaces (arenas) and rules associated to all the possible interactions (scenarios). Whenever a user invoke a function, a request to the service of regulation is carried out. At the regulation layer the rules are verified and the corresponding scenario is executed.

The regulation service is composed by four components: (i) arena manager, (ii) CoRaL parser, (iii) regulation engine and (iv) regulation observer.

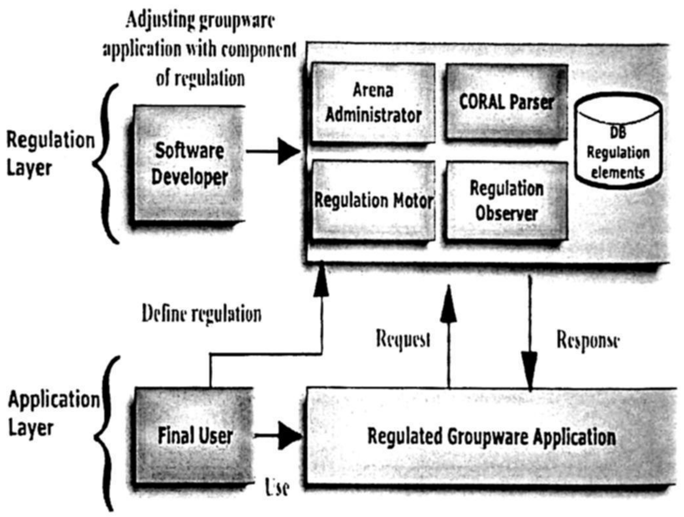


Fig. 2. Regulation service for groupware applications.

- **Arena manager:** Through this component it is possible to define regulation elements such as: arenas in simple or complex form, actors participating on arenas, family of actors/objects, interaction models and views. It includes funtions to add/delete/update elements inside arenas. It also allows to describe scenarios for interactions. The scenario includes, the actor participating on the interaction, the roles asigned to the actors and the sequence of individual actions executed by actors.

- **CoRaL parser:** The parser verifies that the scenarios described for each interaction are lexically and syntactically correct. The parser acts by request of the regulation engine or the arena manager.
- **Regulation engine:** The engine creates, deletes, up-dates and executes interactions according to the scenarios described using CoRaL. For that, it applies operators defined on section 3. So, it must verify the origin of the objects taking part in the interaction in order to know what regulation need to be applied (local or remote). If the interaction concerns remote objects, the regulation engine must send the interaction to the other arena and in the other arena it must apply its own regulation.
- **Regulation observer:** This component implements an observer of all the actions executed inside the arenas according to the rules established by the actors. It could be a software agent that can propose new rules for the group activity based on its observation and analysis.

5 Prototype

In order to demostrate the feasibility of our approach, we developed a prototype implementing the MARS model. This prototype was developed with the Microsoft .Net Framework 2.0 and works with he IIS Web server and SQL Server Express. The regulation service was implemented using a MVS architecture. This regulation service was proved on a groupware application developed at the Faculty of Computer Science of the University of Veracruz. This application, named GroupwareFei, offers tools for communication, coordination and cooperation such as: chat, forum, group agenda and space to share documents. It also offers functions to allow users to create a group and to add/delete group members (See Fig. 3).

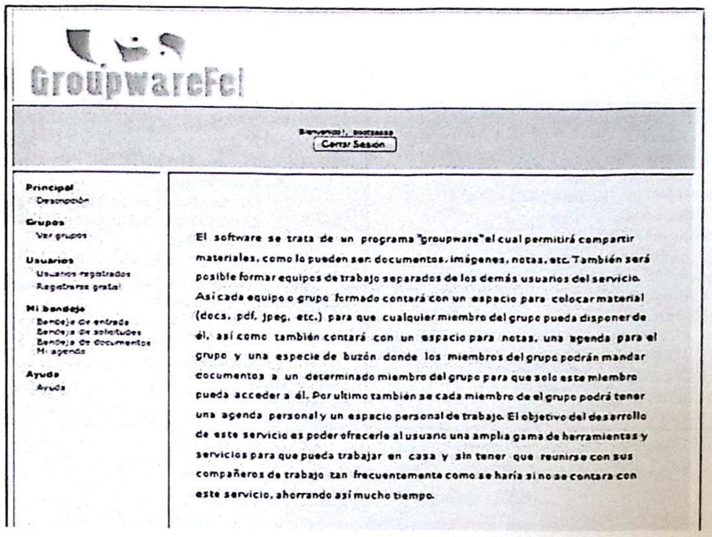
In order to regulate the application it was necessary: i) to identify the functions of the application to be regulated, ii) to adapt the application in order to request the regulation service and iii) to define the scenarios using CoRAL.

5.1 Idetifying functions and adapting the application

The functionalities idetified were "to create a group", "to modify a group", "to share a document", "to communicate using chat", "to send messages to a group or a partner" and "to use a group agenda". Each one was associated to the regulation service in order to define the regulation model *i.e* the arena and all its elements. For instance, the function "to create a group" was associated to the operation "create arena". The association is specifiedt by developers.

5.2 Defining scenarios

The *arena manager* offers an interface that allows the creation of all the instances of the regulation model and the definition of the scenarios. Both actions are carried out by end-users. The instanciation of the regulation model is representing



El software se trata de un programa "groupware" el cual permitirá compartir materiales, como lo pueden ser documentos, imágenes, notas, etc. También será posible formar equipos de trabajo separados de los demás usuarios del servicio. Asi cada equipo o grupo formado contará con un espacio para colocar material (docs, pdf, jpeg, etc.) para que cualquier miembro del grupo pueda disponer de él, así como también contará con un espacio para notas, una agenda para el grupo y una especie de buzón donde los miembros del grupo podrán mandar documentos a un determinado miembro del grupo para que solo este miembro pueda acceder a él. Por ultimo tambien se cada miembro de el grupo podrá tener una agenda personal y un espacio personal de trabajo. El objetivo del desarrollo de este servicio es poder ofrecerle al usuario una amplia gama de herramientas y servicios para que pueda trabajar en casa y sin tener que reunirse con sus compañeros de trabajo tan frecuentemente como se haría si no se contara con este servicio, ahorrando así mucho tiempo.

Fig. 3. Regulated GroupwareFei application.

internally using XML. In Fig. 4 it is possible to observe the definition of a "writing arena". It contains families of actors, actors partcipating inside the arena, objects, roles assigned to actors and objects, interactions and the scenarios.

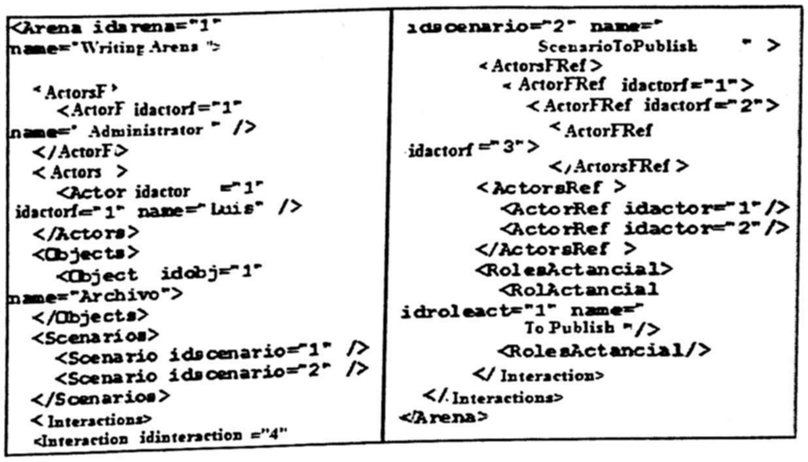
The *parser* validates the scenarios described by the users. The scenarios must be described according to the elements of the arena. For instance, in the following we describe some preconditions of a particular scenario:

1. *Actor family:"manager" -> Interaction:"to Publish";*
2. *Actor family: "collaborator" -> !Interaction:"to Publish";*
3. *Actor family:"visitor" -> !Interaction:"to Publish";*

We observe that the possibility to publish a document is only defined for the "manager".

5.3 Execution of the regulated application

The *regulation engine* is called by the application when a functionality is executed. This engine (i) verifies the interaction associated to the functionality in the regulation model, (ii) verifies the elements associated to the interaction and (iii) execute the scenario. The end-users execute the application according to the group rules, they can modify the elements of the arena. For instance: to add /delete an actor, to modify a scenario, to change roles, etc.



```
<Arena idarena="1"
name="Writing Arena ">

  <ActorsF>
    <ActorF idactorf="1"
name=" Administrator " />
  </ActorF>
  < Actors >
    <Actor idactor   ="1"
idactorf="1" name="Luis" />
  </Actors>
  <Objects>
    <Object  idobj="1"
name="Archivo">
  </Objects>
  <Scenarios>
    <Scenario idscenario="1" />
    <Scenario idscenario="2" />
  </Scenarios>
  < Interactions>
  <Interaction idinteraction ="4"
```

```
idscenario="2" name="
           ScenarioToPublish     " >
      < ActorsFRef >
        < ActorFRef  idactorf="1">
          < ActorFRef idactorf="2">
            < ActorFRef
idactorf =" 3" >
            </ActorsFRef >
      < ActorsRef >
        <ActorRef idactor="1"/>
        <ActorRef idactor="2"/>
      </ActorsRef >
      <RolesActancial>
        <RolActancial
idroleact="1" name="
           To Publish "/>
        <RolesActancial/>
      </ Interaction>
  </ Interactions>
</Arena>
```

Fig. 4. Definition of an arena for GroupwareFeiUV

6 Conclusions

Regulation is a natural social aspect of collaboration. In order to collaborate, people need to establish minimal rules, personal rights, preferences, availabilities and responsibilities in order to carry out their activity. Nevertheless, groupware tools rarely incorporate regulation. We believe, that if groupware applications are designed to support group activities, they must consider the model of this activity. This has two advantages: on the one hand, it make easy to developers the implementation of these applications and, on the other hand, it enables end-users to better adapt and use them.

This paper proposed a regulation service based on MARS, a multi-arena regulation model. This model enables to define the necessary interactions to carry out an activity in a single arena, but also it enables to define interactions taking place in several arenas. In order to develop regulated groupware applications, is also proposed an architecture composed by a regulation and aplication layer.

In order to validate our approach, we implemented a .net-based prototype. This prototype implements the multi-arena regulation model and controls the execution of interactions according to arena regulation. It allows to model the functions of the groupware applications in terms of interactions, to instaciate the regulation model and to describe scenarios for interactios. Developers adapt the functions of the application and end-users instanciate the model and define the scenarios. In this prototype the regulation service only includes the arena manager, the CoRaL parser and the regulation engine.

We observe that interaction can play an active role on the activity, so we explore the possibility to incorpore MAS technology not only for interaction but also for modelling the regulation observer. Our future work includes also modelling scenarios in a more detailed way in order to regulate complex interactions.

Acknowledgements

The authors thank the anonymous reviewers of this paper for their useful comments. This work has been supported by PROMEP (Project number: PROMEP/ 103.5/08/1073).

References

1. L. A. and B. Gregory. Inter-activities management for supporting cooperative software development. In *Proceedings of the Fourtheen International Conference on Information Systems Development ISD 2005*, pages 11–20. Springer Verlang, 2005.
2. Bscw: Be smart cooperate worldwide, 2008. http://public.bscw.de/en/index.html.
3. M. Christian, V. Laurence, F. Christine. and D. G. LDL: a language to model collaborative learning activities. In *ED-MEDIA 2006*, 2006.
4. M. Cortez and P. Mishra. DCWPL : A programming Language for Describing Collaboration Work. In *ACM Conference on Computer Supported Cooperative Work, CSCW'96*, Cambridge MA, USA, November 1996.
5. J. Couet and A. Davie. *Dictionnaire de l'essentiel en sociologie*. Edition Liris, Paris, France, 1998.
6. C. Dave. *The Locales Framework: Understanding and Designing for Wicked Problems*. Kluwer Academic Publishers, 2003.
7. W. K. Edwards. Policies and roles in collaborative applications. In *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 11–20, New York, NY, USA, 1996. ACM.
8. Groove: Microsoft office groove, 2007. http://office.microsoft.com/es-es/groove/default.aspx.
9. T. Herrmann. SeeMe in a nutshell. the semi-structured socio-technical modeling method. Report, Department Information and Technology Manangement (IMTM), University of Bochum, Germany, 2006. https://web-imtm.iaw.ruhr-uni-bochum.de/pub/bscw.cgi/0/208299/30621/30621.pdf.
10. I. Jahnke, C. Ritterskamp, and T. Herrmann. Sociotechnical roles for sociotechnical systems: a perspective from social and computer science. In *AAAi Fall Symposium, 8. Symposium: Roles, an interdisciplinary perspective*, Menlo Park/California(USA): AAAI Press, 2005. AAAI.
11. D. Li and R. Muntz. COCA: collaborative objects coordination architecture. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 179–188, New York, NY, USA, 1998. ACM.
12. D. Li, Z. Wang, and R. R. Muntz. "got coca?" a new perspective in building electronic meeting systems. In *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration*, pages 89–98, New York, NY, USA, 1999. ACM.
13. R. Li and D. Li. A new operational transformation framework for real-time group editors. *IEEE Trans. Parallel Distrib. Syst.*, (3):307–319, 2007.
14. C. Mezura-Godoy. *Une architecture pour le support de la régulation dans les collecticiels*. PhD thesis, Université de Savoie, Chambéry, Francia. 2003.
15. C. Mezura-Godoy, S. Talbot, and M. Riveill. Mars: Modelling arenas to regulate collaborative spaces. *Lecture Notes in Computer Science*, 2806/2003:10–25, June 2003.

16. M. R. Morris, K. Ryall, C. Shen, C. Forlines, and F. Vernier. Beyond "social protocols": multi-user coordination policies for co-located groupware. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 262–265, New York, NY, USA, 2004. ACM.
17. Zimbra: Zimbra collaboration suite (zcs), 2008. http://www.zimbra.com/.