A Group-based Model for Dynamic Communities

Carmen Morgado
CITI/Dept Informática,
Fac.Ciências e Tecnologia,
Universidade Nova de Lisboa
Quinta da Torre
2829 -516 Caparica, Portugal
cpm@di.fct.unl.pt

José C. Cunha
CITI/Dept Informática,
Fac.Ciências e Tecnologia,
Universidade Nova de Lisboa
Quinta da Torre
2829 -516 Caparica, Portugal
jcc@di.fct.unl.pt

Jorge Custódio
CITI/Dept Informática,
Fac.Ciências e Tecnologia,
Universidade Nova de Lisboa
Quinta da Torre
2829 -516 Caparica, Portugal
jfc@di.fct.unl.pt

Nuno Correia
CITI/Dept Informática,
Fac.Ciências e Tecnologia,
Universidade Nova de Lisboa
Quinta da Torre
2829 -516 Caparica, Portugal
nmc@di.fct.unl.pt

ABSTRACT
In this paper we describe a model that supports group-based abstractions to organize dynamic user communities enabling the modeling of typical interaction patterns that emerge in real time interactive applications. Our model deals with dynamic group membership and combines multiple forms of communication mechanisms inside each group unit. The users can interact with each other or with groups based on message passing, asynchronous event notification and access to a shared space. We identified two kinds of groups, the ones where users explicitly express the desire of join/leave a group (explicit groups), and the ones that are automatically formed and were based on a set of predefined rules that are observed by the users (implicit groups). The distinctive characteristic of the implicit groups is the fact that users do no have any explicit participation on the act of join/leave the group. The membership is based only on the users attributes/profile.

Categories and Subject Descriptors
H.3.4 [Systems and Software]: Distributed systems; H.5.3 [Group and Organization Interfaces]: Asynchronous interaction Synchronous interaction Organizational design

General Terms
Design, Experimentation, Human Factors

Keywords
Group-based model, interactive applications, communication mechanisms

1. INTRODUCTION
It is a fact that humans tend to interact with each other, in order to share information and establish relationships and naturally form groups of interests. This fact related with the generalization of the use of computational devices and the technological advances on the network infrastructure give the users the possibility of being connected all the time to each other. The conjugation of this facts motivates research on the models, abstractions and mechanisms in order to enable a transparent and flexible specification of the interactions between users and group based collaborations.

The relevance of the group model approach and its particular characteristics and properties [5], has been recognized in multiple domains like for instance the CSCW (Computer Supported Cooperative Work), multi agents and more recently interactive Web applications for social networking. With the latter mentioned applications new issues concerning the structuring and interaction aspects have emerged.

Our proposed model tries to simplify the development of interactive applications particularly those where multiple forms of communications between users are needed and where the dynamic formation of small and medium location based communities is a special issue. Our MAGO model - Modeling Applications with a Group Oriented approach, and its computing platform is intended to facilitates the organization, access and sharing of contents by multiple users, and enables their dynamic interactions and organization. We consider groups as confined spaces for user interactions where their attributes/profile can be used to exploit geographic proximity of users as well as locality issues associated with access control and privacy. We identify four main contributions of our work:

- the groups can be seen as confined spaces of interaction and cooperation, that act as autonomous elements of the system, and support structuring and organization of distributed applications;
- the ability to dynamically adjust and control group
environments, such as:

- the support of different forms of interactions that are required by real life applications, including peer-to-peer, multicast, asynchronous event notification, and access to a shared space, internal to each group;
- an easy to use set of primitives to manipulate users, groups, their communications and collaboration.

In the following section we identify possible scenarios and refer to some interactive applications, their main requirements and characteristics in terms of organization and interaction between users. In section 3 we present the MAGO model, its main concepts and classes of functionalities as well as a brief description of the main components of its architecture. In section 4 we describe an example of application that was modeled with the MAGO functionalities/primitives. Finally, conclusions and future directions are presented.

2. APPLICATIONS AND REQUIREMENTS

There are many examples where the use of group structuring mechanisms can be found in distinct application environments, such as:

- school, eg. to model a class behavior, or working groups;
- tourism places, eg. a group visiting a museum, or a family visiting a city;
- airports, eg. to model the behavior of passengers in a flight, or an excursion waiting for a connection; to notify the group of people, who like history, about excursions or invitations to share a taxi to go to some historic spot located near by;
- conference management, to model the sessions participants, or the social event;
- shopping center, eg. to advertise sales of sport articles only for the clients that are interested in sports.

In this section we begin with a brief overview of interactive applications and their typically interaction patterns. Finally we identify some of the main requirements and functionalities needed by these applications and the classes of functionalities that are offered by our model.

2.1 Interaction patterns found in applications

Many applications that deal with dynamic interactions between users have emerged in the last years, specially applications concerning the management of the collaboration between users [14], the establishment of connections with other users, and the personalization of applications based on user preferences and context-awareness [8, 13]. We have analyzed the applications in terms of functionalities offered to the users, particularly the ones concerning the forms of communication, the collaborations and the formation/organizations of users structures, and the ones concerning the scale.

We have considered the applications in two main domains, this classification was based on the main type of interactions observed in the applications: The domains are: (1) social software; (2) information sharing. In the first one we considered the applications whose main focus is centered on the management of contacts and the interactions between users. In the second domain the focus is on contents and sharing.

2.1.1 Social software

The more relevant group of applications in this domain is the SNS -Social Network Sites. These applications promote interaction and establishment of connections between remote users, and have an interface and a set of functionalities to ease the management of the users profiles and lists of contacts. The services offered by these applications are centered on Web interaction and rely particularly on asynchronous communications, where the interactions between participants are based on message publishing for later reading. Among the most representative examples of this kind of applications are FaceBook, MySpace or Hi5. Although some of these applications, like the FaceBook, have made available functionalities that enable the direct chat with other online users, the main focus is still the management of the lists of users contacts. Based on the user profile, and the semantics of the application, groups of possible contacts are suggested and communities can be formed based on the users interests and on the users list of contacts. But usually these are very generic suggestions where the users must explicitly join these large communities.

Another class of applications, that can be considered in this domain, are the applications that promote the direct communication between remote users. Example of this are ICQ and mIRC and more recently MSN Messenger from Microsoft, Skype, Google Talk and Yahoo messenger, that offer services to allow the direct exchange of messages, in real-time, between registered users. The more recent applications also support communications with a group of users, and audio and video conference. But these groups of communications are associated with a chat and usually do not persist in time after the conversation in over and the members go offline. Another limitation is that they are usually limited to text and file exchange and do not offer simultaneously other methods of interaction between members, like data share or peer-to-peer communication.

With the advances in the mobile devices equipments and network infrastructure some of the above mentioned applications have developed and adapted their services to mobile versions. As examples we have the FaceBook mobile and the Google mobile, allowing to upload photos and notes from a mobile device and also receive and reply to messages.

2.1.2 Information sharing

The main goal for this domain is to offer a set of services that will give the users the possibility of sharing content (text, photos, audio or video). Some applications give the users some tools to characterize the data, like the possibility of associating metadata information to the shared contents. This metadata information can be used as a way to optimize the data search. These kind of applications offer services to organize, catalog and associate comments to photos and also the data search. These kind of applications offer services to organize, catalog and associate comments to photos and also the data search. These kind of applications offer services to organize, catalog and associate comments to photos and also the data search. These kind of applications offer services to organize, catalog and associate comments to photos and also the data search.
vices also had consequences on emergence of a new kind of applications. On these applications the users location and other information associated to the actual environment will trigger changes on the services offered and on the execution of the application. The concept of context\footnote{A possible definition of context is [8]: "any information that can be used to characterize an entity and is considered relevant to interaction between the user and the application"}, is a major issue in these environments. Applications and platforms like Dodgeball [12] allow the distribution of information based on the actual users position. Some examples of applications, specialized to tourism, are: Tech4Tourism [4] where the tourists can navigate through the maps gathering contextualized information (audio, video, photos or explanation texts) based on their actual position and personal interests; InStory [6] allows the users to have more interaction with the system and with each other, sharing information and grouping themselves based on different activities; or Marble [17] where the museum visitors can have access to information concerning the art objects nearby. None of these mobile applications however, have the possibility of dynamic creation of groups or offer an integrated platform for communications and data share.

2.2 Requirements - why a group-based approach

Group models and system supporting collaboration between multiple users have been the subject of intense investigation in the past decades, in a diversity of contexts and application domains. Also, computational platforms supporting group communication protocols have been developed and used to support the coordination of distributed services and applications. More recently, with the evolution of the Web and Mobile computing, there was a trend towards increasingly interactive and multiuser applications towards social networks. Many of these applications already provide several interesting functionalities such as:

- dynamic management of users profiles and contacts;
- flexibility and dynamic creation of communities and groups of users;
- easy update and access to personal and group information;
- manipulation of different kinds of media;
- possibility of interactions and/or task coordination between users.

These applications also exploit the concepts of virtual groups as a way to enable user interactions in dynamic environments. However, in order to enable their evolution and improvement, we claim there is still a need for exposing adequate group abstractions and programming interfaces.

We have identified the following set of relevant functionalities:

- Need to capture the most typical interactions found in the above applications, namely supporting direct communication between peers, one-to-many communication, and a form of indirect interaction between users through a shared memory space. Such forms of interaction should be coherently integrated in a single model allowing group members to interact by using any combination of the above forms. By supporting distinct semantics for user interaction, it is possible for users to establish communication and coordination according to the needs of each specific application scenario, for example, encompassing synchronous or asynchronous, direct or anonymous, offline or online;

- Need to capture the dynamicity of the application environment, by allowing changes in the set of registered users, as well as changes in the scope and context of their interactions. In order to support these aspects, the model should provide a functionality for handling asynchronous events, as a way to report and disseminate changes in the states of the system and its entities. Additionally, dynamic modifications to the groups membership should be supported by allowing entities to dynamically join and leave groups.

In our model, groups allow distinct and possible complementary interpretations, which we find useful in distinct application contexts:

a) Groups seen as confined or closed interaction spaces requiring entities to join so that only then interaction with other group members is allowed:

b) Groups seen as units for the system organization. A well-defined interface can be associated with a group as a way to separate the specification of the group functionality from the inner details of the group operation. A group can also be seen as a unit of protection and access control mechanisms can be enforced in order to implement security and protection strategies.

c) Groups provide a way to structure the access to shared information repositories. By encapsulating a shared memory internal to each group, and allowing a generic or meta-level representation of the data it is possible to give access to the representation of different information contents, such as photos, videos, audio or any kind of text reference to the data. It is also desirable to establish and manage the relationship between the data elements in the group shared memory and the information in external or remote information systems, allowing access to external databases and persistent information repositories. This is one of the important aspects to enable a structured and flexible access to multimedia contents as required by all existing interactive applications.

All the above functionalities were integrated in our model MAGO - Modeling Applications with a Group Oriented approach, that will be described in the following section. We have developed a working prototype of a Group Oriented architecture supporting the model, and have used the prototype to support the development of small-scale dynamic interactive applications, as described in section 3.2.

3. MODEL AND SYSTEM ARCHITECTURE

This section describes the MAGO model and its supporting architecture.
3.1 The MAGO model

The MAGO model allows to express an application in terms of multiple interacting entities. Two main types of entities are supported:

(i) elementary entities represent computational units executing a program (in general these correspond to processes, objects or agents);

(ii) groups are collective entities corresponding to collections of cooperating entities. In the MAGO model, users are represented by elementary entities, each being modeled as an active object in Java.

Groups allow to model the dynamic characteristics of distributed applications, involving multiple entities joining and leaving the system. A group also acts as an interaction space, supporting distinct forms of communication among its members, namely, direct point-to-point, one-to-many, and shared memory. Synchronous and asynchronous semantics are supported for each of the above forms of interaction. In particular, concerning one-to-many interactions, an asynchronous event dissemination mechanism is supported in the form of a publish-subscribe model. This plays an important role in handling dynamic changes in system configuration.

The above characteristics are implemented on top of existing group communication platforms [3, 1, 16, 11, 2], by building a higher-level communication abstraction supporting a form of shared state or shared memory within each group. In our work we developed an architecture that supports the transparent management of the consistency of views that are observed by the members of each group, concerning the events related to messages, events, and access to shared state. A causality precedence relationship among such events is also preserved.

The primitives supported by the MAGO model can be organized in the following categories:

1. **Entity and group management.**

   On creating an elementary entity, its defining characteristics and attributes are registered into the system. Also, when creating a new group its defining characteristics are specified, such as its name, the admission policy, the maximum number of members admitted, the type of group (explicit or implicit), and the group activation and deactivation deadlines.

2. **Group interactions.**

   Interactions supported within a group include the following: point-to-point, one-to-many, and shared space.

   - Direct point-to-point communication, based on a call to an interface method of an entity or a group.
   - Event dissemination, based on a publish-subscribe mechanism [9, 18] is used to multicast messages to the group members. Group members can subscribe to any event type that has been advertised on the group, and receive a notification whenever an event of that type is published by an event producer. An event can be associated to the sending of messages, or to the notification of changes in the system state.
   - Group shared space: This is based on the concept of a tuple space as proposed in Linda [10] with corresponding primitives to insert, remove and consult tuples. Additionally there is primitive (find) that allows the search of multiple tuples that match a set of specified search parameters. Each tuple has a name, information about the owner, the type and other application specific information. Changes in the tuples in the shared space can also be notified to the interested group members through the event dissemination mechanism.

3. **Implicit groups.**

   MAGO provides a higher level model for group management that enables the automatic management of groups based on user profiles and attributes (implicit groups). These functionalities rely on a search engine, always looking for members whose profiles match a set of specified rules. The implicit group membership is defined depending on the users currently on the system, their characteristics, and the application defined rules. These rules are specified on group creation as simple conjunctions of the user’s attributes, and they define the conditions that must be satisfied by potential candidates to group membership. The evaluation of the satisfaction of such rules is triggered by the event mechanism, possibly leading to the joining or leaving of an implicit group in three different situations:

   - when a new implicit group is created, all matching user profiles are considered as candidates for joining;
   - when a new entity enters/register in the system, its defined attributes are matched against current group definitions;
   - when an entity changes its attributes and preferences, all matching group definitions are considered as candidates. This can lead to an implicit action of joining a group, as well as of leaving an implicit group when the entity no longer satisfies that group membership criteria.

3.2 MAGO supporting architecture

A layered architecture supports the MAGO model as shown in figure 1. A working prototype was implemented in Java, and used to support the experimentation with real applications.

The bottom layer supports the basic communication and operating system functionalities including the management of the connections between the devices supporting user interaction.

The middleware layer - JGroupSpace API [7] - provides a Java-based programming interface for group management, and group communication based on messages, shared spaces and events. In this system a process is an active entity that executes a Java program that can join groups. This middleware layer was implemented on top of the JGroups API [2].

An information system gives access and manages the persistent data manipulated by the system, which in the current prototype is supported through an interface to an ORACLE Database system. The information system stores the application defined contents and the information shared through the groups. It also stores the information related to the system entities and overall system organization, such as the
user’s profiles and the group’s attributes which are used to create and manage the implicit groups.

In our model we have considered several classes of functionalities that can be seen on figure 2, that were translated into primitives offered to the application developers.

A working prototype of MAGO architecture was implemented in Java, and used to support the experimentation with real applications, as briefly illustrated in the following.

4. APPLICATIONS

As we have mentioned before, there are a diversity of scenarios where groups of entities can be used to ease the modeling and development of applications. Examples of such application scenarios arise in working places, travels, tourism spots, shopping centers or any places where “social interactions” exist. These situations, although different, show a set of similar needs and functionalities.

Usually what users want is to: form groups (in an explicit or implicit mode), to establish interactions and to share information. Places like a school campus is an example of a place where the users may tend to use groups as a way to interact, share information to achieve a common goal, to play a game or simply as a mean to meet people that share similar preferences.

In the following, as an example, we present a brief description of an application (school campus scenario) modeled with MAGO. All the users (members of the system) have previously register themselves on the school campus system.

4.1 A case study: school campus

In this application scenario many users have computational devices, particularly laptops and pdas, and have a predisposition to establish groups because they share the same space.

The establishment and generation of groups is usually based on the users professional activities and/or personal characteristics, interests and hobbies, like for instance (3):

- group of teachers or computer science teacher (CS teachers),
- group of computer science student (Joe Group),
- players of the football team (FTeam),
- groups associated with classes or courses (Programming (P1) Class, Math Class),
- group of users of room 123.

Or groups associated with the users preferences and/or profiles like for instances:

- fans of action movies,
- students looking for a room/apartment to rent/share,
- students looking for a part-time job,
- people that have cars and are going to town,
- people that need a ride to town.

These groups are used for spreading information (for example: about a change on a date of a work deliver), for scheduling meetings (ex: agreement on a reunion data) and share media data (ex: put a video and/or slides of a class that will be available to the group of students of that specific class).

4.1.1 Explicit groups

For example to form and manage the explicit group for the user Joe (Joe Group), a typical sequence of MAGO actions that could be performed is:
first step. Create the group "JoeG", this action is launched by user Joe and is made by invoking the create primitive.

\[ joegID = create(joeID, "JoeG", \text{explicit, by\_creator}, 5) \]

This will create a group limited to 5 members and where new members admission is given only by the group creator Joe;

second step. Join the group "JoeG", this action is performed by group candidates by invoking a join primitive.

\[ (join(maryID, joegID) \]

After this action Mary will became a member of "JoeG", if join granted the authorization;

third step. After a group creation, the members can use it to share and access group data, to disseminate information to all group members, or simply to communicate directly to share and access group data, to disseminate information third step.

After a group creation, the members can use it to join granted the authorization;

new members admission is given only by the group creator Joe;

This will create a group limited to 5 members and where the matching rule that must be verified by all the the members will be "nationality italian". In this case the create primitive will have to specify the attributes list

\[ list = [\text{nationality}, "\text{italian}"] \]

So the create primitive will have the form:

\[ italienID = create(man1, "\text{ItalianG}\), \text{implicit,},\text{list}) \]

The implicit groups are also useful to create groups associated to location, like for instance the group of people that are near "Building1" (\[ list = [\text{local}, "\text{Building1}\)"]. With this group created, each time a system user arrives at the specified location, his/her location attributes changes and automatically will became a member of that group. This group can be used to give information about an event that is taking place on that location, or even to the users communicate and share data with each other.

4.1.2 Implicit groups use

If, for instance a school manager want to create a group of all the school members that have italian nationality, he/she should create an implicit group, where the matching rule that must be verified by all the members will be "nationality italian". In this case the create primitive will have to specify the attributes list

\[ list = [\text{nationality}, "\text{italian}"] \]

So the create primitive will have the form:

\[ italienID = create(man1, "\text{ItalianG}\), \text{implicit,},\text{list}) \]

The implicit groups are also useful to create groups associated to location, like for instance the group of people that are near "Building1" (\[ list = [\text{local}, "\text{Building1}\)"]. With this group created, each time a system user arrives at the specified location, his/her location attributes changes and automatically will became a member of that group. This group can be used to give information about an event that is taking place on that location, or even to the users communicate and share data with each other.

4.1.3 Use in a mobile game scenario

We also defined a mobile game scenario where groups of users compete with each other and try to be the fastest on collecting virtual artifacts that were previously defined and distributed all over the campus by the game manager (figure 4). The team that collects more artifacts wins the game.

For modeling this particular application we defined several groups: one for each team and one for the game space.

In a similar way the players can collect clues instead of artifacts that help progressing in the game. During the game the team players can also communicate directly with each other or with the teams (\[ send()\) to leave clues for their team members:

\[ update(player1, team1, \text{public, notify, ObjectClue}) \]

The game manager can also communicate at any time with all the players and teams, using the same primitives. For instance for sending a message to a group team:

\[ send(master, team2, spread, "\text{You have to collect more objects}\)"

Or to all the game players for instance to announce the end of the game a send primitive will be used directed to the group game:

\[ send(master, game, spread, "\text{Game Over}\)"

5. CONCLUSIONS

Our experimentation has confirmed the suitability of the model functionalities. A distinctive aspect of this model is the integration of the mentioned forms of interaction in the context of dynamic groups. Our model relies on previous research from the distributed systems community, namely...
concerning the proposal of the virtual synchrony model [5], in order to ensure the consistency of the views observed by the group members, in the presence of concurrent events related to processes joining and leaving a group, and to the interactions among the group members.

While the model functionalities proved to be very useful for application development, and their implementation proved to be feasible for small or medium scale applications (dozens of entities), we claim there is still a need for a higher-level of abstraction in order to enable the modeling of more complex, dynamic and large-scale applications. As a matter of fact, in modern applications, access to information is increasingly made more complex due to the so-called "data deluge" related to massive amounts of information. The ability to organize or structure the information space can, at least partially, rely upon a group model. In this view, groups can be associated with some meta-information described their functionalities and/or their associated information contents. In this way, users can select the most appropriate groups to contact or to join, depending on their immediate or current interests. And by having associated the group shared memory concept in our model with possibly external information repositories or databases we allow the user to selectively walk through a complex information system, by using groups as "front-ends" to access the repositories of the physical data.

It is also a fact that most applications exhibit a highly dynamic behavior regarding new users joining, users changing their immediate interests or profiles, and also changes in the contents of the information repositories. All of these changes can be related to events triggered and detected by the system. Thus it becomes possible to automate (at least partially) the task of identifying changes in user profiles and/or component attributes, in order to trigger a search for the relevant entities that should be notified of such changes. In particular, when a new user joins an application scenario, or some registered user changes the current profile, it is possible for the system to automatically identify the relevant groups that such user could possibly be interested to join. This idea led to the concept of "implicit groups" that we have incorporated in the MAGO model, as described in the system and illustrated with some real examples on the case study. We claim that this concept opens interesting opportunities for large-scale and complex interactive applications and

6. REFERENCES


