DYNG: A MULTI-PROTOCOL COLLABORATIVE SYSTEM

Thomas Huriaux, Willy Picard
Department of Information Technology
The Poznań University of Economics
ul. Mansfelda 4, 60-854 Poznań, Poland
{thomas.huriaux, picard}@kti.ae.poznan.pl

Abstract Existing systems supporting collaboration processes typically implement a single, fixed collaboration protocol, and collaboration process takes place inside a single group. In this paper, we present in details the *DynG* prototype which provides support for multiple collaboration protocols for non-monolithic collaboration processes, i.e. collaboration processes in which collaboration is spread among many groups. Collaboration protocols used by the *DynG* prototype includes communicative, "acting", and social aspects of collaboration processes, and the introduction of group actions provides support for group dynamics and helps to structure collaboration processes.

Keywords: CSCW, structured collaboration, non-monolithic collaboration, collaboration protocols, group dynamics, communication, social aspects, voting, electronic negotiations

1. Introduction

From prehistoric tribes to trade unions, group structure has always been at the heart of human activities. Grouping their competences, humans are able to achieve great projects, from pyramids to railroad infrastructure construction. The keyword for group activities is *collaboration.* Collaboration is the process of sharing competences to achieve a common goal.

To a recent past, the collaboration process was limited by the requirement of a single location. People involved in a collaboration process needed to meet to exchange information. In reality, people are generally spread on large geographical area. Meetings are difficult to organize, because of schedule incompatibilities, and costly in terms of time and money.

Telecommunication networks provide a partial solution to the former problem. Telecommunication networks let collaborators be spread over various locations. The use of telephone allows collaborators to exchange information

via voice communication. Documents can be exchanged via fax in a graphical format. Local area networks (LAN) are the basis of electronic information exchange inside enterprises, while wide area networks (WAN) – in between enterprises.

With the rise of telecommunication networks, collaboration models that rationalize the collaboration process have been developed. Most of them are document oriented, i.e. the fundamental object of the collaboration process is one or more documents. In enterprises' intranets, collaboration tools are currently widely used for sharing files, for group scheduling or for document collaborative writing.

Traditionally, research in electronic support for collaboration has concentrated on collaboration processes confined inside a single group. Few attention has been accorded to the case of non-monolithic collaboration processes, i.e. processes in which the collaborative activities are dynamically spread among potentially many groups. The term "non-monolithic" is taken from the negotiation vocabulary (see [Raiffa et al., 2002], pp. 4-5, 389-406), where a non-monolithic negotiation process is a negotiation process in which some parties do not behave as a unitary decision entity, i.e. a party consisting of many persons with various perceptions and goals.

In the field of computer support for collaborative work (CSCW), some works have addressed the issue of the group data organization in a dynamic way [Ettorre et al., 2003], the issue of non-monolithic collaborative document edition [Picard, 2004]. These works are usually poorly formalized and focus on very limited applications. In the field of electronic negotiations, some works addressed the issue of negotiation protocols [Benyoucef and Keller, 2000] [Cellary et al., 1998] [Hung and Mao, 2002] [Kersten and Lo, 2003] [Kim and Segev, 2003] [Schoop and Quix, 2001]. According to [Kersten et al., 2004], a negotiation protocol is "a formal model, often represented by a set of rules, which govern software processing, decision-making and communication tasks, and imposes restrictions on activities through the specification of permissible inputs and actions". One may consider a negotiation protocol as a collaboration protocol. Works in the field of electronic negotiations are usually limited to monolithic negotiations, or address a single user's point of view and do not provide support for group collaboration. To our best knowledge, the issue of support for both structured and non-monolithic collaboration processes has only been addressed in our previous work [Picard and Huriaux, 2005] [Picard, 2005]. In these two articles, a model for structured collaboration procotols has been presented. In this paper, the main focus is on a detailed presentation of the prototype implementing the model mentioned above.

In this paper, we present the *DynG* (for Dynamic Groups) prototype which provides support for multiple collaboration protocols for non-monolithic col-

laboration processes. In section 2, the theoretical background – i.e our previous work on a model for collaboration protocols integrating communicative, "acting", and social aspects – is presented. Next, the concept of action is refined with the introduction of a classification of types of actions. In section 4, both the overall architecture and implementation details of the *DynG* prototype are described. Next a complete example of the use of *DynG* to support a collaboration process is detailed. Section 6 concludes the paper.

2. Structuring Non-Monolithic Collaboration Processes

In non-monolithic collaborative processes, collaboration always occurs inside a group. Even when a single collaborator works alone, it may be considered as a group consisting of only herself/himself. Therefore, it may be stated that *a group is a non-empty set of collaborators*. An other aspect of this kind of collaboration is that collaborators are collaborating via message exchange. As we would like to structure non-monolithic collaboration processes, we have to address two issues: first, a mechanism to structure collaboration inside a given group has to be proposed, i.e. message exchange has to be structured, second, actions occuring inside a group have to be addressed.

Collaboration Protocols

Three elements may be distinguished in collaborative processes: a communicative aspect, an "acting" aspect, and a social aspect.

Communication is a major component of collaboration as collaborators need to exchange information to achieve their common goal [Weigand et al., 2003] [Schoop et al., 2003]. The acting aspect of collaboration concerns the fact that collaborators not only exchange information to reach their common goal, but also act to achieve it. Finally, the social aspect of collaborative processes concerns relationships among collaborators, the perceptions they have of others collaborators.

Let's take an example to illustrate the communicative, acting and social aspects of collaborative processes. Let's assume that a parent is reading a fairy tale to her/his child. They collaborate: their common goal being usually to spend some pleasant time together. They communicate: the child may ask why the wolf is so bad at the three little pigs, and the parent answers, or at least tries. They act: the child may point the wolf on a picture in the book, the parent turns pages. The parent and the child are obviously playing different social roles.

The concept of *behavioral unit* captures all three aspects – communicative, acting, and social – of collaborative processes.

Behavioral unit a behavioral unit is a triplet:

(UserRole, MessageType, Action)

- The *UserRole* addresses the social aspect. In the case of the former example, two *UserRoles* may be distinguished: *Parent* and *Child.*
- The *MessageType* addresses the communicative aspect. The introduction of message types allows to limit ambiguousness of communication [Schoop, 2001]. In the case of the former example, three *MessageTypes* may be distinguished: *Question*, *SureAnswer* or *PotentialAnswer*. Intentions of the collaborator can be clearer with an adapted message type. The message "the wolf is fundamentally bad" may be a *SureAnswer* or a *PotentialAnswer*, depending on the confidence of the person answering the question. In this case, the introduction of the adapted message type permits to evaluate the credibility/veracity of exchanged data.
- The *Action* addresses the acting aspect. In the case of the former example, two *Actions* may be distinguished: *PointingTheWolf* and *TurningPage*.

In the proposed model, collaboration processes result from exchange of behavioral units among collaborators. Collaborators are exchanging behavioral units, sending typed messages and acting, in a given role. Exchange of behavioral units causes the evolution of the group in which collaborators are working: each sent behavioral unit causes a transition of the group from a past state to a new state.

Transition A transition is a triplet:

(BehavioralUnit, SourceState, DestinationState)

In the case of the former example, let's define a transition that may occur after the child has asked a question, i.e. the group is in *WaitingForAnswer* state. The transition leads to the *Reading* state. The behavioral unit involved in the presented transition may be the following: *(Parent, SureAnswer, TurningPage)*.

It is now possible to define *collaboration protocols*, which may be used to structure collaboration processes.

Collaboration protocol A collaboration protocol consists of a set of transitions, a set of start states, and a set of terminating states.

One may notice that a protocol is a variant of finite state machines. A finite state machine (FSM) is usually defined as "a model of computation

consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state". The set of states of the FSM can easily be deduced from the set of transitions of the protocol. The start state occurs in both the FSM and the protocol. The input alphabet of the FSM is the set of behavioral units which appear in all transitions of the protocols. Finally, the transition function of the FSM is defined by the set of transitions of the protocol. The only difference between FSMs and collaboration protocols is the existence of terminating states for protocols.

A collaboration protocol is a template definition for a set of collaboration processes. Using an analogy with object-oriented programming, one may say that a collaboration protocol is to a protocol instance what a class is to an object. In a given group, a given protocol instance regulates collaboration among group members.

Protocol instance A protocol instance is a triplet:

(Protocol, CurrentState, UserToRoleMapping)

The *UserToRoleMapping* is a function which associates a *UserRole* with a given user.

3. Refining the Concept of Action

Inside a group, many actions can occur, modifying the structure of this group, creating new groups, or having no real influence on the main structure of the collaboration. Three types of actions have been identified:

1 neutral actions;

2 actions modifying the structure of groups;

3 actions modifying the structure of collaboration processes.

Prior to a description of these three types of actions, let's define the concept of *structure of a group*. In the later, the structure of a group will refer to a pair that consists of the set of collaborators within the group and the mapping between users and roles.

By analogy, let's define the concept of *structure of a collaboration process*. In the later, the structure of a collaboration process will refer to a pair that consists of the set of groups within the collaboration process and the mapping between groups and protocols.

Neutral Actions

Neutral actions have no effect, neither on the structure of the collaboration, nor on the structure of the group in which the action is processed. Therefore,

different existing groups inside the collaboration process are not influenced by these actions. Their structures remain the same, no new group is created. Furthermore, the group inside which the action was processed keeps its own structure unchanged, i.e. the same users with the same roles.

In the case of purely communicative behavioral units, i.e. behavioral units which aim only at exchanging information, associated actions should be neutral ones. However, neutral actions should not be limited to the case of purely communicative behavioral units: if a file is modified during the collaboration process via an *edit* action, the *edit* action has no effect on the structure of neither the collaboration process nor the group.

Actions Modifying the Structure of a Group

As it has already been mentioned, a group is composed of one or more users, each of them playing a given role. However, a user may have his/her role changed during the collaboration process. Actions causing such changes modify the structure of a group.

A user (1) with a role (2) is giving to a user (3) with a role (4) a new role (5). Therefore, five parameters have to be taken in account:

1 the user executing the action;

2 the role of the user executing the action;

3 the user having his/her role changed;

4 the role of the user having his/her role changed;

5 the newly attributed role.

One may notice that the case in which a user changes his/her own role is just a special case of the generic one presented above. Indeed, an action allowing a user to change his/her role is just an action in which the values of the first and the third parameters are the same.

Actions Modifying the Structure of a Collaboration Process

The structure of non-monolithic collaborative processes is usually highly dynamic: groups are created and deleted in a dynamic way. Group dynamics is the result of the execution of actions modifying the structure of the collaboration model: a collaborator may for instance decide to execute an action *createANewGroup*. Two kind of actions may be distinguised modifying the structure of a collaboration process: actions may modify either the set of groups – called *group actions* – or the mapping between groups and protocols – called *protocol dynamic actions*.

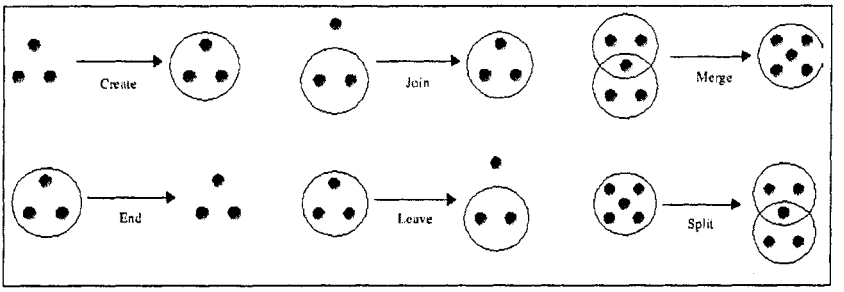

Figure 1. Group actions

Group Actions. The following group actions have been identified:

- *join* action: adds at least one collaborator to the set of collaborators of an existing group;
- *quit* action: removes at least one collaborator from the set of collaborators of an existing group;
- *split* action: splits an existing group in two or more new groups and the union of the sets of the collaborators of the created groups equals the set of collaborators of the existing group;
- *merge* action: creates a new group consisting of the union of the set of collaborators of at least two groups;
- *create* action: creates new group;
- *end* action: deletes an existing group.

These actions are illustrated on Figure 1. Dots represent collaborators while circles represent groups. One may notice that, as shown on Figure 1 for the *split* and *merge* actions, a given collaborator may participate at a given time in many groups.

Group actions may either modify only the group in which the action is processed – e.g. the *quit* action – or involve other groups – e.g. the *merge* action.

Protocol Dynamic Actions. This type of action is required in two cases: to allow a group to change its protocol during the collaboration process, and to design and implement parameterized protocols.

A protocol is parameterized if it may have parameters whose values are specified during the collaboration process. An example of such a protocol could be a protocol in which an action is used to specify how many messages can be sent before going to the next step of the collaboration process.

4. The *DynG* Prototype

Overall architecture

The *DynG* (for *Dynamic Groups*) prototype is an implementation of the formerly presented concepts: collaboration protocols and actions. It aims at being a platform supporting structured non-monolithic collaboration processes.

The *DynG* prototype consists of three parts: the *DynG* Core, the *DynG* Server, and the *DynG* Client (the term *DynG* will be omitted in the rest of the paper to improve readability).

The overall architecture is presented on Figure 2.

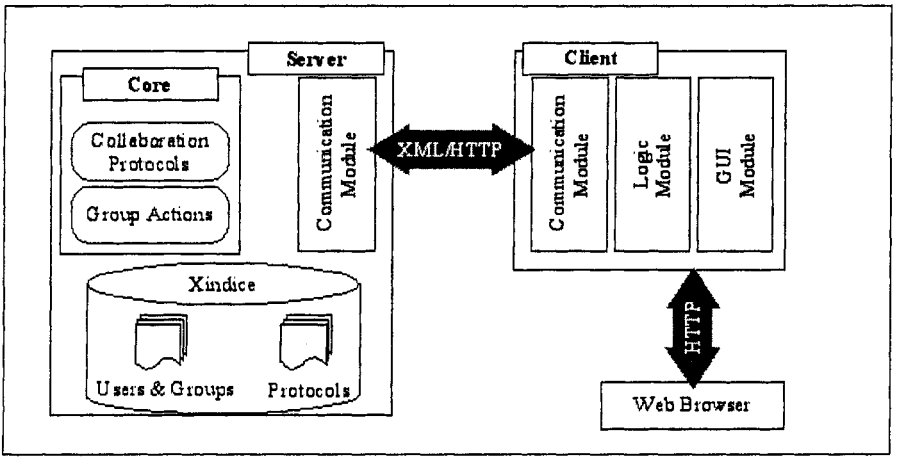

Figure 2. Overall architecture of the *DynG* prototype

DynG Client

The Client is a Java Servlet aiming at providing an interface to the users. Each HTTP request coming from the users' web browser is passed to the logic module. The logic module may exchange information with the Server via the communication module. The communication module translates Java objects created by the user's request into XML messages to be sent to the server, and translates the responses from the server, which are also in XML format, into Java objects understandable by the Servlet. When the logic module has finished its work, it redirects to the GUI module which is responsible for generating dynamic HTML pages for the final user. The GUI module consists of a set of Java Server Pages (JSP).

During one request, many XML messages are exchanged between the Server and the Client. The user may require only information about the state of the

collaboration, or may perform one action modifying the state of this collaboration. Each basic request requires one message. For example, three distinct requests between the Client and the Server are required to know in which groups the user belongs, to get the collaborators inside the group the user is working on and to get the potential actions the user may perform.

DynG Server

On the server side, three elements may be distinguished: the communication module, the Core, and a repository. The communication module is responsible for translating XML messages received via HTTP from the Client into calls to the Core, and to create response from Java objects into XML messages to send back to the Client.

The Core provides support for collaboration protocols and group actions and is responsible to maintain the state of the collaboration: existing users, protocols, groups, etc. The Core manages the collaboration processes according to protocols stored in the repository. In the current implementation of the *DynG* Server, the Xindice native XML database [Xindice, 2005] is used as a repository but it would be possible to use other storage mechanisms, such as file systems or relational databases. The repository is responsible for storing not only information concerning users and groups, but also protocols, exchanged messages, etc.

The *DynG* Administration

The introduction of protocols allows collaboration processes to be structured. At the implementation level, the introduction of protocol instances allows to restrict the set of possible behavioral units in a given state of a given group. As a consequence, the GUI module must be highly dynamic as it has to display to the user only behavioral units that are available at the current state and for his/her role. Therefore, depending both on the role of the user and the current state of the collaboration process, the graphical user interface (HTML pages) will be different, allowing him/her to perform different behavioral units.

First of all, the collaboration module has to be initialized to set the collaboration main protocol, i.e. the protocol that rules the collaboration process, to add concerned users, etc. The HTML page presented on the left side of Figure 3 shows how collaboration processes are administrated, and what is needed for a collaboration process to start. As a remark, collaboration processes take place in "workspaces" in the *DynG* prototype. A second HTML page, presented on the right side of Figure 3 provides an interface to assign roles to users inside the collaboration process, but only at the top level, i.e. at the workspace level.

Once a collaboration process is initialized, a collaborator can login and collaborate with other persons involved in the collaboration process.

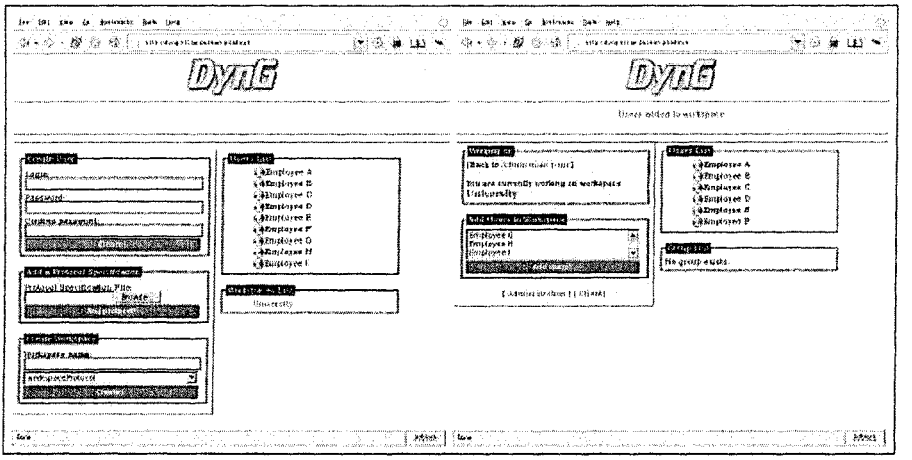

Figure 3. *DynG* administration

5. Example: Election of a University Rector

The goal of the current section is to present the potential use of *DynG* to support a real collaboration process. The collaboration process chosen for this example is the electoral process of the Rector of a university. The collaboration process is a slightly simplified version of the process existing in our university. To improve readability of this article, and because the communicative facet of this process is limited (except for the campain), the message types will be omitted.

Description of the Electoral Process

The pre-requisite is that candidates for the Rector position are known. Then, the electoral process is composed of three main phases: candidatures for the electoral chamber, votes for the electoral chamber, votes for the Rector. The two first steps will be fully described below. For the last step – the votes for the Rector – only elected members of the electoral chamber are concerned.

The process will be divided into two protocols: the candidatures and first votes steps grouped together, and the second votes in another protocol. In other words, the first protocol models the election of the electoral chamber, while the second one models the election of the Rector by the electoral chamber.

To assist the collaboration, an agent has been added. The software agent is considered as a normal user, i.e. it is sending behavioral units according to its role and the voting protocol.

The voting protocol is graphically represented on Figure 4, where rectangles are state, arrows are transition, and tabular are different behavorial units associated to a given transition.

Candidatures for the electoral chamber During this step, each employee of the university may be candidate. But the candidature must be proposed by somebody else, which is a candidate or not. Once somebody has been presented as a potential candidate, he/she can accept to be candidate or not. When the minimal number of candidates is reached, the agent will go through a transition going to a similar situation, except that now the commission manager can decide to stop the candidature period and to move to the vote for the members of the electoral chamber.

Votes for the electoral chamber During this procedure, each employee, candidate or not, may vote for up to the number of candidates to be elected. This "round" is finished either by the agent when everybody has voted, or when the commission manager decide it. If enough candidates have been elected, by reaching a pre-defined majority, then the vote ends. Otherwise a new "round" is started, after the removal of a pre-defined number of candidates.

The protocol is "tuned" by an action specifying how many members the electoral chamber consits of and how many candidates are removed after each round of the voting process. This action is a *protocol dynamic action.* This action is part of the behavioral unit denoted *Set vote specifications* in Figure 4.

During the last step of the vote, each member of the electoral chamber starts with the same default role, and can perform only one action: to vote. To be formally precise, this action is composed of two parts: one part changes the "default" role of the voter into a role "voted", while the second part adds one voice to the chosen Rector candidate.

Use of *DynG* to Support the Vote Process

The Main page displays to the collaborator the list of groups, the group she/he is working on, and the messages sent to this group. The situation at the beginning of the electoral processes is shown on the left side of Figure 5. In this situation, the only allowed action is to propose a candidate, as shown in the "Group Management" part of the interface. The messages present the different steps: the settings of the vote specification and the candidates already proposed, with their refusal or acceptance.

Once a candidate has been proposed, no action can be performed, except for the new potential candidate who can either accept or refuse his/her candidature. This situation is presented on the right part of Figure 5.

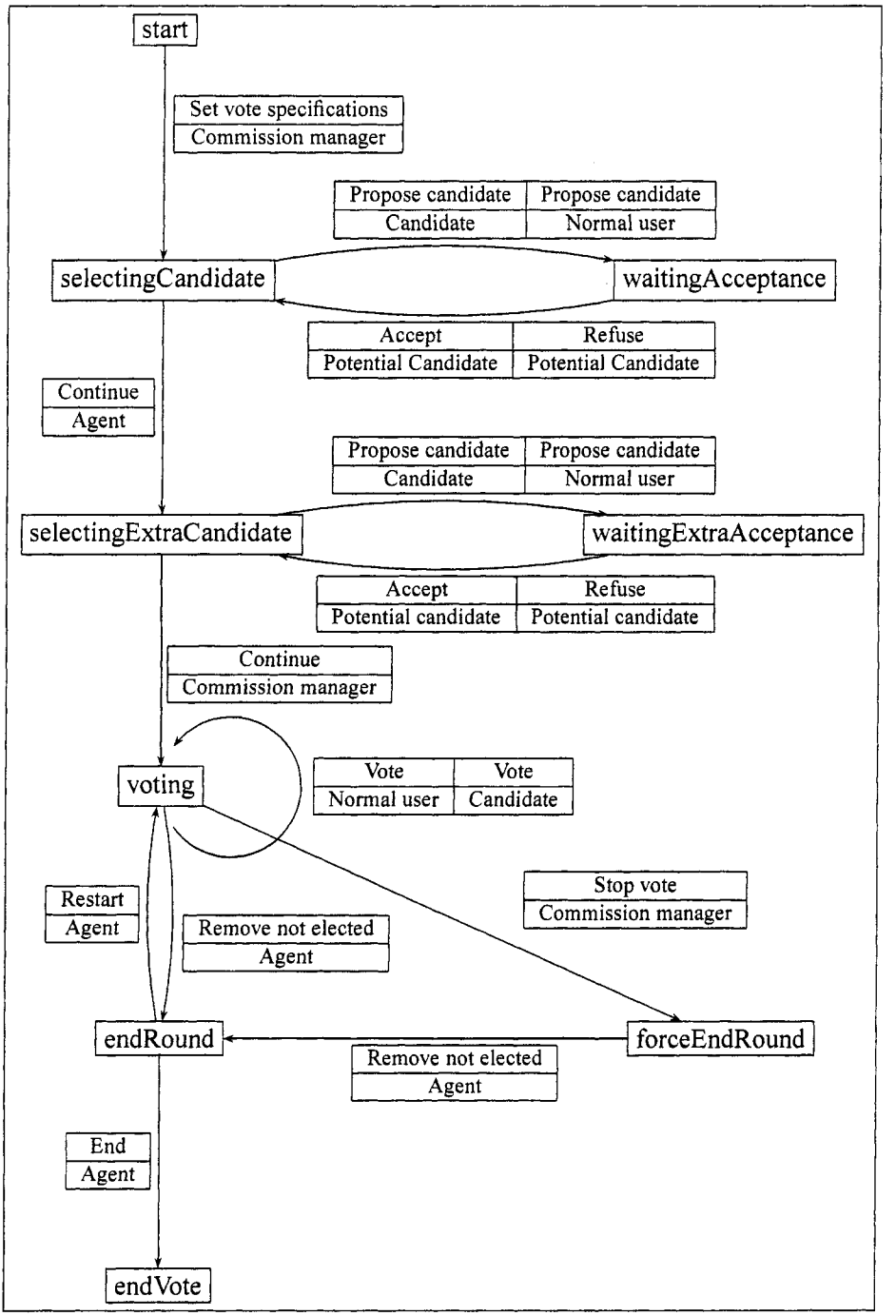

Figure 4. Graph representing the electoral protocol

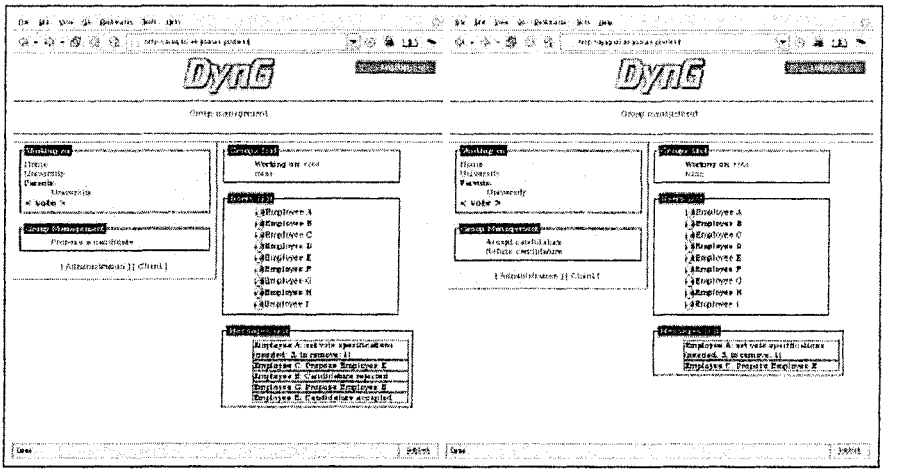

Figure 5. *DynG* client: the candidature period

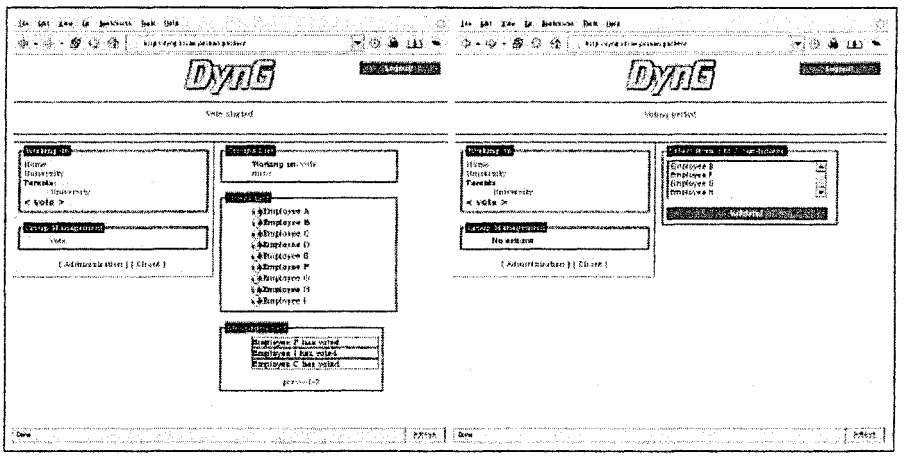

Figure 6. *DynG* client: the vote period

The same situation is repeated until the commission manager decides that there are enough candidates. The voting period then starts. As shown on the left part of Figure 6, the only action each user can now perform is to vote. Once a user decides to vote, she/he then has to choose for whom she/he will vote (right part of Figure 6) and then they cannot perform any action until the end of the vote.

It should be kept in mind that a given protocol rules a given group, and that various groups may be ruled by different protocols. Therefore, by clicking on

the name of an other group in the group list, a user may get abilities to perform other behavioral units available in the new working group, depending on the protocol ruling the group and the role of the collaborator inside the group.

At last, the "Working on" component of GUI allows collaborators to get an overview of the group dynamics, by presenting both the parents and the children groups of the working group. The "Working on" component may be use to browse groups the collaborator belongs to.

6. Conclusions

The introduction of collaboration protocols and group actions allows to provide computer support to non-monolithic collaboration processes. To our best knowledge, it is the first model for electronic support for non-monolithic collaborative processes.

It would be possible to build complex support systems for complex collaborative processes using the framework provided by the DynG prototype. The design of systems for non-monolithic collaboration processes may be resumed in the following steps: first, the roles involved in the collaboration process have to be identified. Next, the required actions have to be implemented. Then, message types should be defined. Therefore, behavioral units may be defined. Finally, collaboration protocol(s) may be specified.

The presented model could be used in a broad spectrum of potential applications. The presented model may for instance be applied to non-monolithic negotiations, such as international negotiations or business-to-business contract establishment. Another field of applications is the legislative process in which various political parties, potentially presenting various opinions, collaborate in order to establish laws in form of new or modified legal acts. The presented model could also be used to design support systems for collaborative documentation edition processes that often take place among business actors.

Among future works, it would be interesting to investigate the possibilities to embed a protocol instance into another protocol instance. This would allow to modularize protocols, to design protocols using smaller protocols, to develop "protocol libraries". In the example presented in this paper, one may notice that the candidature proposal phase can be seen as a "subprotocol" which could be reused in other protocols, or even many times within a single protocol.

Another field which could be the object of future works is the concept of role. The addition of relationships between various roles, such as inheritance or composition, would be an interesting work to be done.

References

[Benyoucef and Keller, 2000] Benyoucef, M. and Keller, R. K. (2000). An evaluation of formalisms for negotiations in e-commerce. In *DCW '00: Proceedings of the Third International Workshop on Distributed Communities on the Web*, pages 45–54. Springer-Verlag.

[Cellary et al., 1998] Cellary, W., Picard, W., and Wieczerzycki, W. (1998). Web-based business-to-business negotiation support. In *Int. Conference on Electronic Commerce EC-98*, Hamburg, Germany.

[Ettorre et al., 2003] Ettorre, M., Pontieri, L., Ruffolo, M., Rullo, P., and Sacca, D. (2003). A prototypal environment for collaborative work within a research organization. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 274–279. IEEE Computer Society.

[Hung and Mao, 2002] Hung, P. and Mao, J.-Y. (2002). Modeling of e-negotiation activities with petri nets. In *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 1*. IEEE Computer Society.

[Kersten and Lo, 2003] Kersten, G. E. and Lo, G. (2003). Aspire: an integrated negotiation support system and software agents for e-business negotiation. *International Journal of Internet and Enterprise Management*, 1(3).

[Kersten et al., 2004] Kersten, G. E., Strecker, S., and Law, K. P. (2004). Protocols for electronic negotiation systems: Theoretical foundations and design issues. In Bauknecht, K., Bichler, M., and Pröll, B., editors, *EC-Web*, volume 3182 of *Lecture Notes in Computer Science*, pages 106–115. Springer.

[Kim and Segev, 2003] Kim, J. B. and Segev, A. (2003). A framework for dynamic ebusiness negotiation processes. In *CEC*, pages 84–91. IEEE Computer Society.

[Picard, 2004] Picard, W. (2004). Towards support systems for non-monolithic collaborative document edition: The document-group-message model. In *DEXA Workshops*, pages 266–270. IEEE Computer Society.

[Picard, 2005] Picard, W. (2005). Towards support systems for non-monolithic electronic negotiations. *Special Issue of the Journal of Decision Systems on e-Negotiations*. To appear.

[Picard and Huriaux, 2005] Picard, W. and Huriaux, T. (2005). Dyng: Enabling structured non-monolithic electronic collaboration. In *The 9th International Conference on CSCW in Design*, Coventry, UK. To appear.

[Raiffa et al., 2002] Raiffa, H., Richardson, J., and Matcalfe, D. (2002). *Negotiation Analysis, The Science and Art of Collaborative Decision Making*. The Belknap Press of Harvard University Press.

[Schoop, 2001] Schoop, M. (2001). An introduction to the language-action perspective. *SIGGROUP Bull.*, 22(2):3–8.

[Schoop et al., 2003] Schoop, M., Jertila, A., and List, T. (2003). Negoisst: a negotiation support system for electronic business-to-business negotiations in e-commerce. *Data Knowl. Eng.*, 47(3):371–401.

[Schoop and Quix, 2001] Schoop, M. and Quix, C. (2001). Doc.com: a framework for effective negotiation support in electronic marketplaces. *Comput. Networks*, 37(2):153–170.

[Weigand et al., 2003] Weigand, H., Schoop, M., Moor, A. D., and Dignum, F. (2003). B2b negotiation support: The need for a communication perspective. In *Group Decision and Negotiation 12*, pages 3–29.

[Xindice, 2005] Xindice (2005). http://xml.apache.org/xindice/.