# *DynG*: Enabling Structured Non-Monolithic Electronic Collaboration

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

## 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 spread dynamically among potentially many groups. The term "non-monolithic" is taken from the negotiation vocabulary (see [9], 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 [3], the issue of non-monolithic collaborative document edition [8]. 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 [1] [2] [4] [5] [7] [12]. According to [6], 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 never been addressed.

In this paper, we present the *DynG* (for Dynamic Groups) prototype which provides support for multiple collaboration protocols for non-monolithic collaboration processes. In section 2, a model for collaboration protocols integrating communicative, "acting", and social aspects is presented, then group actions required to provide support for group dynamics are introduced. In section 3, both the overall architecture and implementation details of the *DynG* prototype are described. Section 4 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, which means that message exchange has to be structured, second, group dynamics have to be addressed.

### 2.1. 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 [13] [11]. 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 [10]. 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 start state, 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 be easily 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.

### 2.2. Group Dynamics

In non-monolithic collaborative processes, groups evolve: a collaborator may join or leave an existing group, a group may split in two or more groups, two or more groups may merge into a single group. Group dynamics may be modeled by a set of *group actions*. The following group actions have been identified:

- `create action`: creates a new group;
- `join action`: adds an author to the set of collaborators of an existing group;
- `merge action`: creates a new group consisting of the union of the set of collaborators of at least two groups;
- `end action`: deletes an existing group;
- `leave action`: removes a collaborator from the set of collaborators of an existing group;
- `split action`: creates at least two groups from an existing group and the union of the sets of collaborators of the created groups equals the set of collaborators of the existing group.

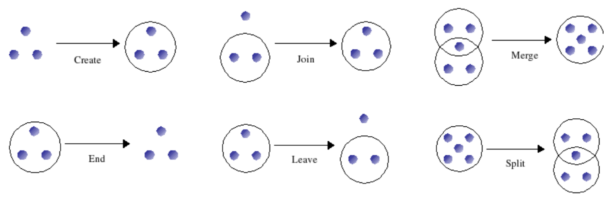


**Figure 1. Group actions**

Group 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.

## 3. The *DynG* Prototype

### 3.1. Overall architecture

The *DynG* (for Dynamic Groups) prototype is an implementation of the formerly presented concepts: collaboration protocols and group actions. It aims at being a platform for implementation of collaborative systems.

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 Core is responsible for the implementation of the model presented in section 2, and is implemented using Java. The Server provides a unified XML-based interface to the Core. The introduction of the Server allows to decouple access to the Core from the Core itself: the Server is responsible for communication with clients and translates requests from clients to access to the Core. The Client is responsible for interaction with the final user and the Server. Such an architecture allows various communication protocols between clients and the server.

The overall architecture is presented on Figure 2. Both the Server and the Client are implemented using Java Servlets. On the client side, 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. 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).

On the server side, three elements may be distinguished: the communication module, the Core, and a repository. The communication model is responsible for translating XML messages sent via HTTP into calls to the Core. The Core provides support for collaboration protocols and group actions. 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 [14] 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 known protocols.

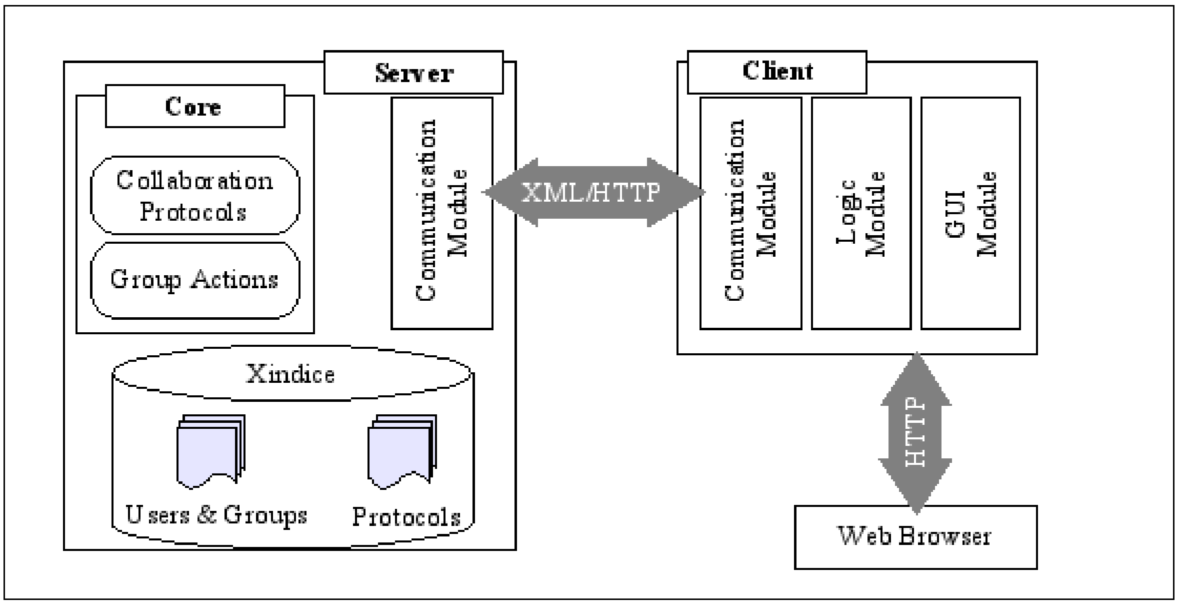


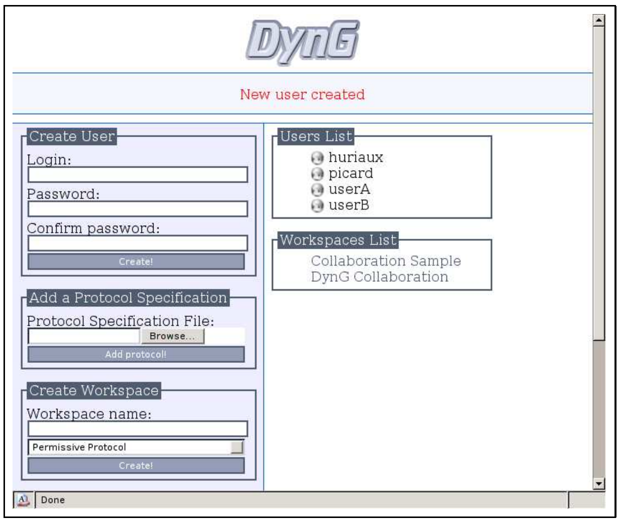
**Figure 2. Overall architecture of the *DynG* prototype**

## 3.2. Implementation details

The introduction of protocol instances 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 the collaboration group. As a consequence, the GUI module must be highly dynamic as it has to propose to the user only behavioral units that are available at the current state and for the role of the given user. Therefore, depending both on the role of the user and the current state of the collaboration process, the graphical user interface (the HTML pages) will not only be different but also allow a user 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 in 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 page provides an interface to set the `UserToRoleMapping` inside the collaboration process, but only at the top level, i.e. the workspace level.

Once a collaboration process is initialized, a collaborator can login and collaborate with other collaborators of the collaboration process. The Main page give the collaborator the list of groups, the group she/he is working on, and the messages sent to this group. To describe dynamic creation of the interface, let's take an example: we assume that the workspace for a collaboration process has been successfully


DynG

New user created

Create User
Login:
Password:
Confirm password:
Create!

Users List
huriaux
picard
userA
userB

Workspaces List
Collaboration Sample
DynG Collaboration

Add a Protocol Specification
Protocol Specification File:
Browse...
Add protocol!

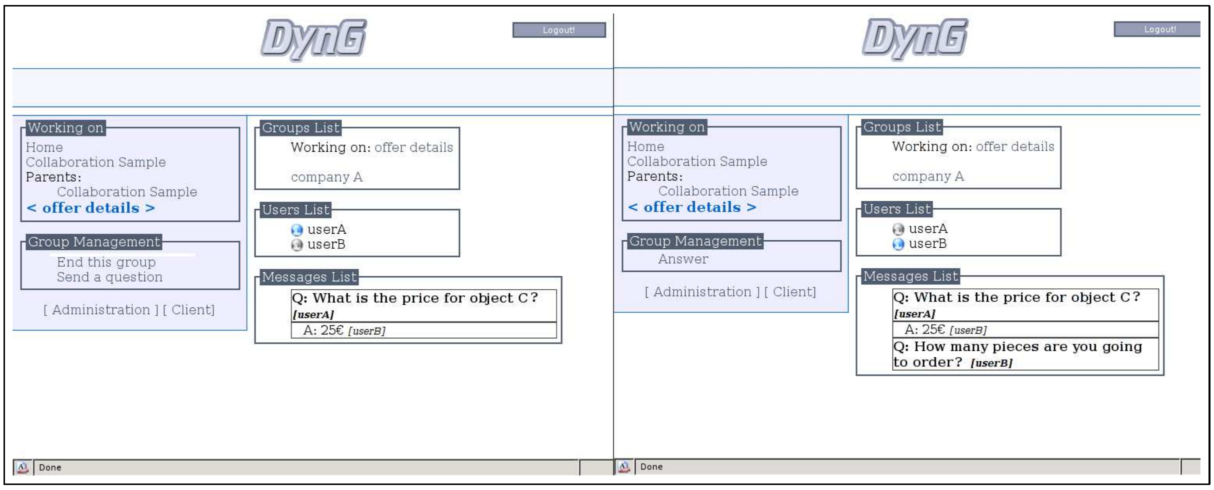
Create Workspace
Workspace name:
Permissive Protocol
Create!

Done

**Figure 3. Administration web page**

created, and that user *A* has created inside this workspace a group *G* to ask user *B* information. Group *G* will use a basic protocol consisting of a question followed by an answer. Group activity can be ended each time the last sent question has been answered.

We thus have formally the three following behavioral units, according to the definition given in section 2.1, i.e. the triplet
`(UserRole, MessageType, Action)`:



Working on
Home
Collaboration Sample
Parents:
Collaboration Sample
< offer details >

Group Management
End this group
Send a question

[ Administration ] [ Client]

Groups List
Working on: offer details
company A

Users List
userA
userB

Messages List
Q: What is the price for object C? [userA]
A: 25€ [userB]

Working on
Home
Collaboration Sample
Parents:
Collaboration Sample
< offer details >

Group Management
Answer

[ Administration ] [ Client]

Groups List
Working on: offer details
company A

Users List
userA
userB

Messages List
Q: What is the price for object C? [userA]
A: 25€ [userB]
Q: How many pieces are you going to order? [userB]

**Figure 4. Question and answer protocol in *DynG***

- `AskInformation = (BasicRole, Question, Asking)`
- `AnswerQuestion = (BasicRole, Answer, Answering)`
- `EndGroup = (BasicRole, Success, Ending)`

And the three following transitions, i.e. triplets `(BehavioralUnit, SourceState, DestinationState)`:

- `(AskInformation, WaitingForQuestion, WaitingForAnswer)`
- `(AnswerQuestion, WaitingForAnswer, WaitingForQuestion)`
- `(EndGroup, WaitingForQuestion, EndState)`

To complete the instance of this protocol, we also need a UserToRole mapping. In this case, there is only one role:

- `(UserA, BasicRole)`
- `(UserB, BasicRole)`

On the left side of Figure 4, the user interface is presented in the case when a question and an answer to the question have been sent.

As specified in the presented protocol, it is not possible to answer a question that has already been answered. When a question has been answered, one may only send a new question or end the group activity, as we may notice in the "Group Management" part of the graphical interface. Let's ask for more information, by sending a new question. The user interface is presented on the right side of Figure 4. Now that a question has been asked, one may only answer the question, according to protocol specifications. It may be noticed that the graphical interface has changed, as the "Group Management" component is generated according to the protocol.

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, we may get abilities to perform other behavioral units available in this new working group, depending on the protocol ruling this group and the role of the collaborator inside this 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.

## 4. 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 takes place between 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". 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

[1] M. Benyoucef and R. K. Keller. 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, 2000.

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

[3] M. Ettorre, L. Pontieri, M. Ruffolo, P. Rullo, and D. Sacca. 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, 2003.

[4] P. Hung and J.-Y. Mao. 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, 2002.

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

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

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

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

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

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

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

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

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

[14] Xindice. http://xml.apache.org/xindice/.