BAT: Brainstorm Assistance Tool

António Lopes, Sofia Esteves, Bruno Gonçalves, Luís Botelho
"We, the Body and the Mind" Research Group of ADETTI
Avenida das Forças Armadas, Edifício ISCTE
1600 Lisboa, Portugal
Tel.: +351-217826480
{antonio.lopes,sofia.esteves,bruno.goncalves,luis.botelho}@we-b-mind.org

ABSTRACT

This paper describes BAT (Brainstorm Assistance Tool), a computational system for supporting distributed brainstorms based on a well-defined structured graphical interface. BAT was created within the *KnowledgeAndCulture.org* project, whose objective is to create a set of tools to facilitate the human resources' activity of team formation for cooperative tasks. In this case, the brainstorm-like electronic meeting was the chosen cooperative task. This paper describes BAT, which monitors and records the interactions that take place between several participants of distributed brainstorms. The data generated by BAT is used in agent-based simulation to assist the selection of the best team composition to perform specific cooperative tasks. The system can be used through a graphical language defined for general purpose Brainstorms or through the addition of plug-ins for domain-specific Brainstorms. In this paper we present the overall system design, including the underlying brainstorm model and the output generation process.

1 INTRODUCTION

The *KnowledgeAndCulture.org* [1] project addresses the organizational problem of team formation for cooperative work. The goal is the formation of the best team of people to perform a specified cooperative task, given the set of available organizational actors. When the number of available people is large, the problem of selecting the best team among all the possibilities would consume an enormous amount of resources if all possible team compositions were considered.

In some advanced resourceful organizations this decision problem involves highly specialized psychologists that manually create cognitive maps of all the available people [2]. These maps provide information regarding compatibilities and possible conflicts among them. Through the thorough analysis of those maps, a relatively small set of the most promising teams is selected. The selected possibilities are further evaluated until the best team is eventually identified. The described decision process involves the work of highly specialized well paid workers during considerably large periods of time, thus turning it into a long and very expensive process.

To this end, the *KnowledgeAndCulture.org* project has defined three different stages:

1. Design and develop a computational system for distributed brainstorms; which monitors and records interactions between real organizational actors in cooperative problem solving tasks;
2. Design and develop learning algorithms that create the models of organizational actors from the outputs generated by the computational system developed in stage 1;
3. Use the learnt models (created in stage 2) in a multi-agent system that simulates and evaluates the performance of all possible team compositions, in order to determine the most adequate for the specific cooperative task.

This paper describes BAT (Brainstorm Assistance Tool) that has been developed in stage 1 to support distributed cooperative work. BAT relies on a formal representation of Brainstorms, which enables the automated processing of the interactions between the brainstorm participants.

The paper is organized as follows: section 2 presents the brainstorm model, which is used by computational processes, such as the algorithms that learn the models of the brainstorm participants (stage 2), and the simulation system that selects the best teams for the specific task (stage 3). Section 3 describes the developed computational system, BAT. This includes describing the overall system's architecture, its internal components and their interactions. Section 4 is dedicated to the detailed explanation of the output generation process, representing the recording of the users' interactions within the brainstorm activity. In section 5, we analyze the current state-of-the-art regarding collaborative work software and related technologies. In section 6, we conclude and present guidelines for future work.

2 BRAINSTORM MODEL

We have chosen the Brainstorm as the cooperative work task to be addressed by the project. A brainstorm is a cooperative activity, suitable for solving poorly structured problems of a wide variety of classes. The general characterization of the Brainstorm allows us to study many instances of cooperative work.

Brainstorms interactions involve sharing complex and informal oral and written messages whose computational processing is very difficult and lies

outside the scope of the *KnowledgeAndCulture.org* project. Therefore, we have constrained the degrees of freedom associated with those interactions by using a carefully designed graphical user interface through which the interactions between organizational actors involve only simple menu choices.

To this end, we created a domain independent formal model of the brainstorm activity, which captures the whole set of possible types of communicative actions participants may perform in a brainstorm. Brainstorms are distributed cooperative problem solving processes therefore it is only natural to formally represent them as problem-solving acyclic graphs (Figure 1).

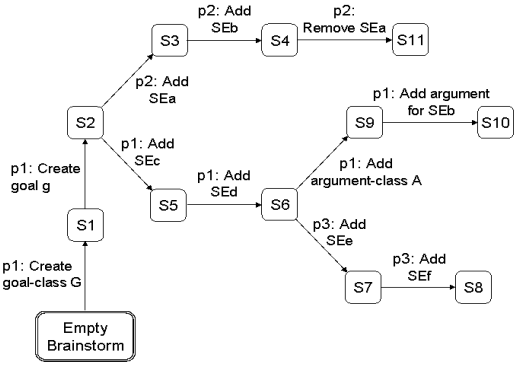

Figure 1 – Brainstorm Graph

Nodes in the brainstorm graph are states representing partial or complete solutions to the discussed problem. Arcs represent actions performed in the origin state that produce the destination state.

The brainstorm model comprises the description of the brainstorm graph and the description of the states represented by the brainstorm graph nodes.

2.1 Brainstorm Graph

All brainstorm graphs have at least one initial node called the Empty Brainstorm, which represents the brainstorm before the discussion of ideas starts. The empty brainstorm node represents state *S0*. Arcs represent actions performed by brainstorm participants. Arc $p{:}a_{i,j}$ from state *i* to state *j* represents the action performed by participant *p* in state *i* leading to state *j*. In addition to the actions they represent, arcs in the brainstorm graph are labeled with the identifier of the participant that performs the corresponding action.

Additionally to actions performed by individual participants, there are certain actions that are performed by the brainstorm as a whole. These actions represent decisions made by the group of participants, according to the decision policy used in the group. The proposed brainstorm model does not address the way decisions are made. They can be made by consensus, or by the majority of participants, or autocratically, or by any other process but this does not concern us here. The decisions of the brainstorm are actions annotated as being done by the special user *Brainstorm*.

The brainstorm graph is an acyclic graph because, once a certain state has been created due to the addition of a new element to the preceding state, the additional element can never be removed therefore it is impossible to return to the original node. Actions used to remove elements form brainstorm states do not actually remove them. Instead of being removed, elements are marked as removed. The initial action in a brainstorm is represented by an arc from the *EmptyBrainstorm* node to the node representing the first non empty brainstorm state. The initial action in the brainstorm graph represented in Figure 1 consists of participant *p1* creating goal class *G*. The second action consists of participant *p1* creating a goal (*g*) of the defined class (*G*). This goal represents the problem to be solved during the discussion.

After the definition of the brainstorm problem (state *S2*), there are two alternative proposals regarding its solution. Participant *p2* proposes the creation of solution elements SE_a and SE_b. Participant *p1* proposes solution elements SE_c and SE_d. Then (in state *S6*) he or she proposes to create the argument class *A*, and to add an argument of class *A* for solution element SE_d, while participant *p3* proposes yet two more solution elements SE_e and SE_f.

Meanwhile participant *p2* proposes the removal of the previously introduced solution element SE_a.

The proposed brainstorm model includes the specification of the set of all possible actions that can be performed by brainstorm participants. Space limitations preclude the presentation of all possible brainstorm actions therefore only a few action types are exemplified:

- Objects, relations, goals and other kinds of entities are always described by the class to which they belong. This requires the existence of class creation actions such as *create-object-class*, *create-relation-type* and *create-goal-class*. Instance creation actions, such as *create-object* and *create-goal*, associate the created instance to its class. For example:
    - *Create-goal-class* and *create-goal* – define a new goal class and insert a goal to be achieved by the Brainstorm. Brainstorm goals represent problems to be solved.
- *Add solution element* – inserts an element of the final solution. This is used by participants to propose a fragment of the final solution that will achieve one or more of the brainstorm goals. The kinds of elements that can be added to the evolving solution depend on the actual problem being solved. Solution elements for planning problems are actions; solution elements for design problems are design fragments; and so on. Examples of such type of actions are *create-class*, *create-object*,

create-relation-type, *create-relationship*, *create-goal-class* and *create-goal*.

- *Remove-solution-element* – marks an existing solution element as removed.
- *Remove-goal* – Marks an existing goal as removed.
- *Add / remove-argument* – This action is used to insert / mark as removed an argument for / against a given brainstorm element. Arguments are brainstorm elements that do not belong to the solution. Brainstorm elements are goals and goal classes, solution elements, arguments and argument classes, or documents and document classes.
- *Associate / dissociate-documents* – Documents may be associated to / dissociated from brainstorm elements with documenting purposes, for instance to provide legislation regarding some possible action.
- *Acceptance / rejection of brainstorm elements* – Can be used by individual participants and also by the brainstorm as a whole.
- *Associate / dissociate solution elements* with goals are actions used to create a relationship of a previously existing solution element with a previously existing goal.

Most of the brainstorm actions could be reduced to the creation of relationships between brainstorm elements. Those actions have been proposed not because they are strictly necessary but because their meanings are more intuitive than the corresponding create-relationship actions

2.2 Brainstorm State Representation

An action executed in one brainstorm state generates another brainstorm state which is a modified version of the state in which the action is performed. This section describes the representation of the brainstorm states and relates the described actions with the generated representations.

We have used the SNePS (Semantic Network Processing System) [15] representation language to describe brainstorm states. The most important construct in the brainstorm state representation is the relationship.

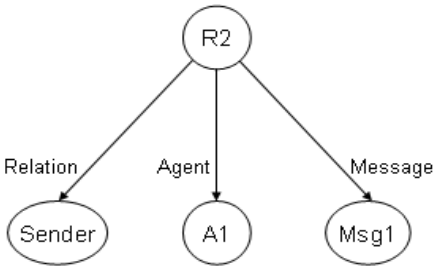

Figure 2 - Relationship

Figure 2 shows a relationship between agent *A1* and message *Msg1*. Informally, the node *R2* means that agent *A1* is the sender of message *Msg1*. Relationships such as the one shown in Figure 2 are the result of the *create-relationship* brainstorm action.

Additionally to the graphical notation as that used in Figure 2, we also use a textual representation language:

R2 # sender(agent : A1, message : Msg1).

Assertions in the proposed textual representation have two parts: assertion label and assertion content. The assertion label is the identifier of the node representing the assertion. The content is a proposition whose predicate is the name of the relation used in the relationship, and the arguments are pairs *<role>:<value>* in which, roles are the names of the arcs used in the relationship, and values are the nodes pointed to by the corresponding arcs.

Relationships are the basic construct used to represent most of the results of the brainstorm actions described in section 2.1. Object-classes, argument-classes, goal-classes, and relation-types, among others are all declared through relationships using specific reserved relations, such as *declared-objectClass*, *declared-argumentClass*, *declared-goalClass*, and *declared-relation*.

Figure 3 shows the declaration of the *sender* relation. Informally, node *R1* states that *sender* is being declared as a relation.

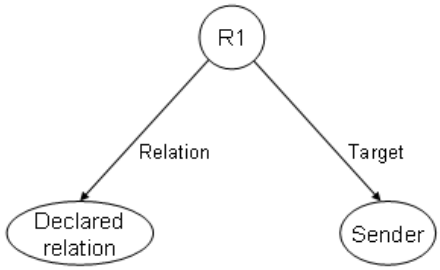

Figure 3 – Relation type declaration

The relationship graphically depicted in Figure 3 can also be textually represented through the following assertion:

R1#declared-relation(target : sender).

The action *create-relation-type* declares a new relation type, resulting in a new graph fragment exactly as the one shown in Figure 3.

More complex assertions are also represented through relationships. For instance, node *R3* in Figure 4 represents the fact that "agent *A1* being the *sender* of message *Msg1* contributes to the satisfaction of goal *G1*"

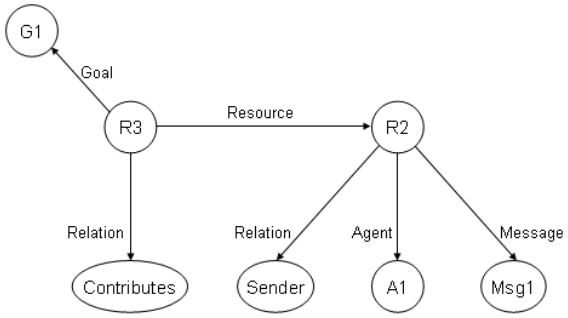

Figure 4 – Solution element contributing to a goal

R3 is also a relationship using the reserved relation *contributes* and the roles *resource* and *goal*. Textually, *R3* can be represented by the following assertion:

R3 # contributes (
resource : R2 # sender(agent : A1, message : Msg1),
goal : G1
)

This time, the *resource* role is played by node *R2*, which represents another complete assertion.

The kind of content in Figure 4 is the result of the brainstorm action *add-relationship-to-goal*.

Many other contents such a removed elements, accepted and rejected elements, goal decompositions, and result derivations are all represented by relationships. However the exhaustive description of the brainstorm state representation language is out of the scope of this paper.

3 BRAINSTORM ASSISTANCE TOOL

In this section we describe BAT - Brainstorm Assistance Tool – system in detail. This system was developed on top of the brainstorm model described in section 2. BAT's main goal is to provide a computational system that enables the interaction between several different participants in a distributed cooperative task. In section 3.1, we describe the overall functioning of the system. Section 3.2 describes BAT's architecture and implementation details.

3.1 Functional Description

BAT is an electronic meeting support system that provides a cooperative communication framework for users to participate in distributed brainstorm activities.

To initiate their interaction, participants must login into the system (Figure 5), which allows them to enter in a specific brainstorm activity. This authentication procedure is based on two parameters: a brainstorm token (which uniquely identifies the brainstorm activity) and a password. Other optional parameters are available in the login, such as username and the user real name. However, users may choose to remain anonymous.

Figure 5 - BAT's Login GUI

After successfully performing the login, the participant is ready to interact with all the other participants in the brainstorm using the application's graphical user interface.

The participants will cooperate in order to find a solution to a proposed problem. This is done through the use of graphical elements that can be inserted in a common area visible to all participants. This area is called the *whiteboard* (Figure 6).

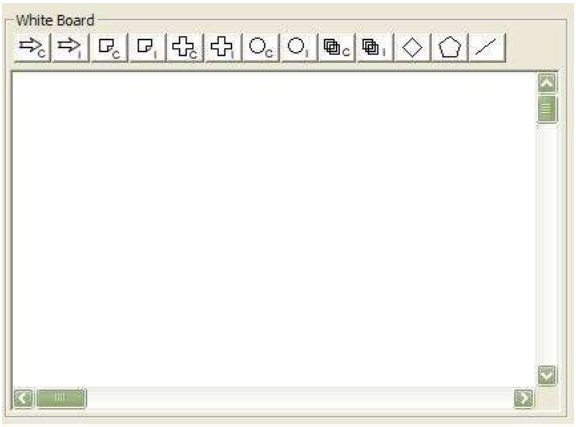

Figure 6 - Whiteboard section of BAT's main GUI

The *whiteboard* area is a common area where the participants can propose solutions to the problem and these will be viewed by every other participant. The graphical elements represent specific concepts, such as classes, objects, relations, goals, actions, arguments and generic data (such as documents) that can contribute, in some way, to the total or partial solution of the proposed problem.

The interactions that take place between the participants are not guided by any specific interaction protocol or set of rules. All participants can make proposals at any time of the brainstorm activity. Synchronization is transparently solved by the application.

When a participant performs a specific action in the brainstorm, the description of the performed action is broadcasted to the remaining participants allowing each of them to automatically update the state of brainstorm. The various states of the brainstorm (steps taken in the brainstorm towards the solution to the proposed problem) are represented in a specific graphical area called the *states graph diagram* (Figure 7).

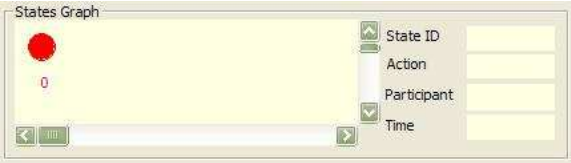

Figure 7 - States Graph Diagram section in BAT's main GUI

BAT keeps a history of the brainstorm activity in this *states graph diagram*, enabling participants to consult previous states of the brainstorm. This allows them not only to be aware of all the steps that were taken in the brainstorm, but also gives them the opportunity to propose alternative solutions.

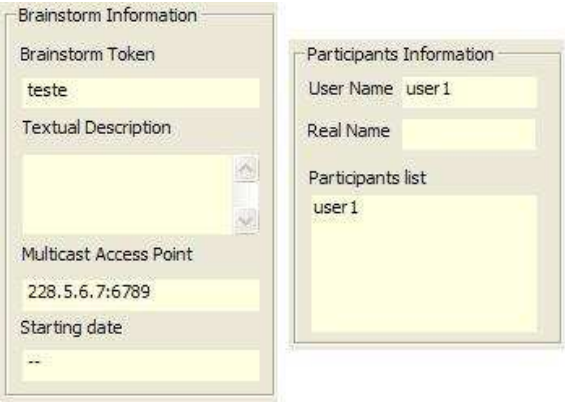

Figure 8 - Brainstorm Description section in BAT's main GUI

The system provides information that is relevant to the participant, such as brainstorm description parameters (Figure 8) (token, participant list, date and time) and individual brainstorm state description parameters (Figure 7) (triggered action, participant that triggered it and time stamp).

BAT has also a chat component that allows participants to talk with the whole group. This is needed whenever a graphical representation of the solution is not enough for all participants to understand it. However, the use of this component is not encouraged because the information exchanged through the chat is not processed in any way by the system.

3.2 Architecture and Implementation

The BAT system is composed of a server application and several client peer applications (one for each participant). The server application is used for authentication and for managing connections to shared communication channels. Each specific brainstorm has a shared communication channel (multicast channel) associated with it. Each participant, using a client application, communicates with all other participants using the multicast channel. In order to support different brainstorms occurring simultaneously, the server application manages all connections and authentications to these multicast channels. The client application is the participant's interface to the BAT system, which allows him or her to participate in a specific brainstorm activity with other participants. The client application also records all the interactions of its participant. This output generation process is described in section 4.

BAT's architecture is a mix between a client-server architecture and a peer-to-peer architecture. The server application is responsible for managing the authentication and the multicast channel connection processes. The client applications, once connected to the multicast channel, interact with each other as peers. This approach has the advantage of distributing the brainstorm computing process through all client applications, instead of having a centralized server doing all the work. The interactions are processed by the client peer applications. The server application is left with only the simple task of authenticating and managing multicast channels.

3.2.1 BAT's client application

Internally, each client application has two mandatory components: the brainstorm engine and the user interface. Optional components, such as plug-ins for domain-specific brainstorms, can be added to the client applications. These plug-ins provide participants with user friendly interfaces especially tailored for specific problems, such as strategic games. Each plug-in offers a full duplex translation service between the problem specific inputs and outputs, and the general purpose brainstorm representations.

The user interface is composed of the following elements:

- *Whiteboard* – common drawing area, where all participants can insert graphical elements cooperatively in order to construct a representation of the solution for a specific problem (Figure 6)
- *Actions Toolbar* – toolbar that contains the various actions that participants can perform within the brainstorm activity (Figure 6)
- *States Graph Diagram* – specific area that contains a graph representing the history of the brainstorm activity. Each node represents a brainstorm state, and each arc represents an action performed by one of the participants in the brainstorm (Figure 7)
- *Chat Section* – area where the participants can write small text messages, which will be read by the remaining participants
- *Participant List* – list of the participants in the brainstorm activity (Figure 8)

- *Brainstorm description parameters* – set of parameters that describe the brainstorm (Figure 8)
- *Current State description parameters* – set of parameters that describe the brainstorm state selected by the participant (Figure 8).

The brainstorm engine is the kernel of the client application. It receives actions' descriptions from the client application's interface, processes it and performs the appropriate actions (such as creating new brainstorm states and sending information to other client applications). The brainstorm engine stores all the information produced/gathered during the brainstorm activity, such as the brainstorm history and description parameters. This component is also responsible for listening the multicast channel for incoming messages. When it receives data from any of the remaining client applications, it processes it and, if appropriate, sends the information to the interface component.

The brainstorm engine component will also be used in the simulation stage as the agents' main component, which will drive their activities within the brainstorm simulation process (see section 6 for details).

3.2.2 Implementation details

The BAT system was built using the JAVA™ programming language and the XSP software [13], a component-based software tool providing both development and execution support for component based applications.

The XSP software, developed by the *"We, the Body and the Mind"* Research Group of ADETTI/ISCTE, enables the creation of components that communicate using a transparent communication framework based on XML data exchange.

This communication framework relies on the simple registering of services in a yellow-pages server, which can be consulted by any component seeking for specific service providers. This search mechanism is hidden from the components, since it is automatically activated when invoking services.

For designing the GUIs in the BAT system, we used Eclipse's SWT plug-in [14]. SWT (Standard Widget Toolkit) is a graphics library and a widget toolkit integrated with the native window system (Linux and Solaris are supported as well). Despite the tight integration with the native target platform, SWT is an OS-independent API. SWT can be seen as a thin wrapper over the native code GUI of the host operating system.

4 OUTPUT GENERATION PROCESS

BAT stores all the information related to the interactions that occur within a brainstorm activity in a well-structured generic brainstorm representation. This will allow the use of this information by a great number of applications, including learning algorithms and case bases.

The brainstorm state representations that are generated and maintained by BAT are used in agent based simulations as partial models of organizational actors. Each pair of a state representation and the associated actions is stored in case bases (one for each actor) and used in case-based reasoning during the simulation.

The brainstorm state representations (e.g., Figure 2, Figure 3, and Figure 4) are graph structures consisting of nodes and arcs connecting them. In those representations, there are two kinds of nodes: atomic nodes and molecular nodes.

Atomic nodes are those from which there are no emanating arcs. The atomic nodes in Figure 4 are *G1*, *contributes*, *sender*, *A1* and *MSG1*.

Molecular nodes are those with arcs to other nodes. The molecular nodes in Figure 4 are *R2* and *R3*. Molecular nodes are said to dominate other nodes. *R3* in Figure 4 dominates *G1*, *contributes* and *R2*; and *R2* dominates *sender*, *A1* and *MSG1*. Atomic nodes do not dominate other nodes.

BAT internally encodes the brainstorm state representations as symbolic structures of two different types: *molecularNode/2* and *atomicNode/1*.

molecularNode/2 structure has two components: the node identifier and the node parameters. Node parameters represent the arcs emanating from the node.

Each node parameter is encoded as a *parameter/2* structure with two components: arc and node, which may be an atomic or a molecular node.

atomicNode/1 structure has only the node identifier.

For instance, molecular node *R2* in Figure 4 is represented by the *molecularNode/2* structure presented in Figure 9.

```
molecularNode(
  name : 'R2',
  parameters : [
    parameter(
   arc : relation,
   node: atomicNode(name : sender)),
    parameter(
   arc : agent,
   node : atomicNode(name : 'A1')),
    parameter(
   arc : message,
   node : atomicNode(name : 'Msg1'))])
```

Figure 9 – Molecular node internal encoding

R3 is encoded as a *molecularNode/2* structure with three parameters, two of them represent the arcs to atomic nodes *G1* and *contributes*, and the third one represents the arc *resource* to the molecular node *R2*.

A brainstorm state is represented by the set of nodes that are not dominated by any other node. In the case of Figure 4, the brainstorm state is the singular set

containing the *molecularNode/2* structure representing node *R3*.

The internal encoding of brainstorm state fragments is not the same as their textual representation as defined in section 2.2. Therefore, before being used by the simulation agent case-based reasoning mechanism, they must be translated to the defined textual format.

The translation process converts atomic node identifiers into literal constants, and it converts molecular nodes into labeled assertions.

The internal encoding shown in Figure 9 corresponds to the following textual representation:

R2 # sender(agent : A1, message : Msg1)

The node identifier used in the internal encoding (*R2*) plays the role of the assertion label in the textual format. The main predicate of the textual format of a molecular node is the atomic node introduced by one of the arcs *relation*, *action* or *function* in its list of parameters. In the case of Figure 9, the node introduced by the *relation* arc is *sender* therefore the predicate of the textual representation is *sender*. The arguments of the textual assertion are the result of the translation of the molecular node list of parameters.

The translation of a *parameter/2* structure

parameter(arc : <arc name>, node : <node>)

is

<arc name> : translation(<node>)

in which, translation(<node>) is the recursive translation of the atomic or the molecular node as just described.

The arcs of the parameters list in Figure 9 are *relation*, *agent* and *message*. The relation arc has already been used for the main predicate name. The *agent* arc introduces the atomic node '*A1*', and the *message* arc introduces the atomic node '*Msg1*'.

5 RELATED WORK

There are several methodologies, applications and systems available for electronic cooperative systems. In this section, we present some of these methodologies and systems and their relation to our work.

Nominal Group Technique [3] is a brainstorm organization methodology, which is based on the following steps: individual generation of ideas; round-table discussion for clarifying each idea; idea voting process; and final discussion of results. In [3] a system based on this methodology is also presented, the NGTool. This is not a very good example of a cooperative system but it is a good way of managing distributed meetings with a lot of different participants. BAT's methodology is more open and its main goal is to promote cooperation between the participants.

A GUI designer for decision support systems is presented in [5], which presents a well structured description of the principles of this kind of systems. Some of the ideas presented in that paper were used when developing BAT, such as, the broadcast communication mechanism (used in multicast communication channels) and the synchronization mechanism (to prevent communication conflicts).

DOLPHIN [9] is an electronic meeting support system, but it wasn't conceived to work as a distributed system. Its main goal is to enhance the capabilities of the traditional meetings. BAT's *whiteboard* concept – as the main common area in the electronic meeting where all participants can place their contributions – was taken from DOLPHIN, since it is a good mechanism to enable the cooperation between the participants.

GroupSystems [10] is one of the most referenced [4], [7], [8] group support meeting systems. Although, well organized and user-friendly, this system has a very open free text based interface, which can not be used within the *KnowledgeAndCulture.org* project, since it would involve the resource to robust natural language processing systems in order to convert free text inputs into a machine understandable format.

TeamEC [11] is a group decision support system very similar to GroupSystems therefore it exhibits the same disadvantages.

GroupExplorer [12] is a flexible group decision support system that enables facilitated work teams to carry out a variety of strategic management and decision making processes effectively. It produces cognitive map-based outputs, which is more congruent with the main goals of the BAT system. However, this system could not be used since it is not free and only commercial licenses were available.

6 CONCLUSIONS AND FUTURE WORK

We have presented the BAT system (and the generic purpose brainstorm model) as an enabling technology for users to participate in distributed brainstorms to perform cooperative tasks. We have also presented the output generation process, which will allow the use of the brainstorm activities' data in a multi-agent simulation system. The simulation process will reveal the best team(s) for performing the specified cooperative task.

In the near future, our goal is to use the BAT system in the *KnowledgeAndCulture.org* project by creating a checkers brainstorm plug-in, which will allow collecting brainstorm activities' information of this domain, for use in the next stages of the project.

Furthermore, the main component of the BAT's client application (the brainstorm engine) will be incorporated in the agents, in the multi-agent simulation system.

In the end of the project, we will have an analysis on how the Brainstorm Model supports all types of interactions in real brainstorms, e.g., the stage in which the problem is clearly defined. At this time, we will identify the strengths and weaknesses of the Brainstorm Model in order to improve it and make it usable in every domain.

We also intend to create a number of post-process applications to assist the extraction, storage and management of knowledge from meetings – like meeting minute's generation, intelligent knowledge retrieval from meeting minutes – integrated in a larger system for organization knowledge management.

7 REFERENCES

[1] *KnowledgeAndCulture.org*-POSI/SRI/46582/20027 http://www.knowledgeandculture.org

[2] Louçã, J.: *Cartographie Cognitive, Réflexion Stratégique et Interaction Distribuée: une Approche Multi-Agent. PhD Thesis*, Paris IX Dauphine University and Sciences Institute of Lisbon University (2000)

[3] P. Antunes and N. Guimarães, "*NGTool–Exploring Mechanisms of Support to Interactivity in the Group Process.*" CYTED-RITOS International Workshop on Groupware, CRIWG '95. Lisboa, Portugal, September, 1995.

[4] C. Costa, M. Duque, P. Antunes, and J. Dias, "*Sistemas de Apoio a Reuniões: Limitações Actuais e Oportunidades de Disseminação Da Tecnologia*" Information Systems, vol. 12, July, 2000. (Special Issue with selected papers from the first workshop on Cooperative and Distributed Multimedia Systems, CoopMedia 2000. ISSN: 0872-7031).

[5] P. Antunes and N. Guimarães, "*User-Interface Support to Group Interaction.*" Second International Workshop on Groupware, CRIWG '96. Puerto Varas, Chile, September, 1996.

[6] N. Gonçalves and P. Antunes, "*'Decision Can': Uma Biblioteca de Casos para Tomada de Decisão*" Information Systems, vol. 12, July, 2000. (Special Issue with selected papers from the first workshop on Cooperative and Distributed Multimedia Systems, CoopMedia 2000. ISSN: 0872-7031).

[7] Nunamaker, Jr., J.F.; Dennis, A.R.; Valacich, J.S.; Vogel, D.R.; and George, J.F, "*Electronic meeting systems to support group work.*" Communications of the ACM, 34(7):40-61, July 1991.

[8] Bonner, Richard F. and Basavaraj, Devendra K. L., "*Electronic Group Support Systems: Lessons from Business for Education*", The 12th Annual Conference of the Australian Society for Computers in Learning in Tertiary Education, December – 1995

[9] Norbert A. Streitz and Jörg M. Haake, "*DOLPHIN - Hypermedia-based Support for Group Meetings*", in ERCIM News No.21 - April 1995 (Online Edition - http://www.ercim.org/publication/Ercim_News/enw21/dolphin.html)

[10] GroupSystems Application, "*Visit a GroupSystems Meeting*", On-line tutorial at www.groupsystems.com (http://www.groupsystems.com/products/tour1.htm)

[11] TeamEC for windows, "*Team Expert Choice – Group decision support software*", On-line presentation on www.telemanworld.com. (http://www.telemanworld.com/razaofinal/pt/ec/e_choice.html)

[12] Group Explorer, "*Group Explorer*", On-line presentation on phrontis,Lda website. (http://www.phrontis.com/ge.htm)

[13] *eXtended Service Platform* - "We, the Body and the Mind" Research Group of ADETTI/ISCTE (http://we-b-mind.org/EN/software.html#XSIP)

[14] IBM - *Eclipse Platform* (http://www.eclipse.org)

[15] Shapiro, S.C.; and Rapaport, W.J. 1987. "*SNePS considered as a fully intentional propositional semantic network*". In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 263-315. Springer-Verlag, New York