INTERACTIVE WHITEBOARD FOR COLLABORATIVE WORK: A MULTI-AGENT BASED SOLUTION

Franck GECHTER, Stéphane GALLAND
ICAP Team, Laboratoire Systèmes et Transports (SeT),
Université de Technologie de Belfort-Montbéliard (UTBM), France
franck.gechter@utbm.fr, stephane.galland@utbm.fr

Keywords: Interactive White Board, Software Engineering, Multi-Agent System, Collaborative Wrok

Abstract: An Interactive Whiteboard (IWB) is a device that is a giant tactile and projection screen. It can be considered as one of the main elements for pedagogical innovation with information technology. IWB allows to transform the classical classroom environment into a working and learning interactive environment. This pedagogical tool is particularly pertinent in higher school especially in scientific teaching where it allows to better illustrate scientific approach thanks to the ability to record annotations in association with a time line. If the use of IWB seems to be natural in lecture, it can also be used in tutorial and practical class where collaborative work is particularly important. The collaborative ability is closely tied to the softwares used. Few software have functionalities aimed at facilitate collaborative work. This feature is the main issue of this article. To develop this ability, a multi-agent approach has been chosen. Multi-agent systems approach is one of the most interesting thanks to its intrinsic properties and features such as simplicity, flexibility, reliability, self-organization/emergent phenomena, low cost agent design and adaptation skills,... This paper presents the solution developed in order to make IWB able to communicate with each other.

1 INTRODUCTION

An Interactive Whiteboard (IWB) is a device that is both a giant tactile screen and a projection screen. It can be considered as one of the main elements for pedagogical innovation with information technology by many countries. Interactive whiteboards allow to transform the classical classroom environment into a working and learning interactive environment that fits edutainment aims. To that way, IWB must work in association with a standard computer or laptop and a data projector. One single touch allows to modify a document, to launch applications, to open web sites, to write annotations in electronic ink on plans, etc. Then, modifications and pedagogical progression can be saved and replay as many time it is required. This feature is one of the principal key points of IWB as compared with classical blackboard.

Nowadays, IWB are mainly used in primary and secondary schools. For instance, in Great Britain nearly 100 percent of primary schools are equipped by a IWB. By the same, in France, IWB equipment is supported by a national action called PrimTICE[1]. As for higher education, the use of IWB depends on schools and universities policies. For the moment, especially in France the equipment rate is very low. However, this pedagogical tool is particularly pertinent and allows to better illustrate and emphasize scientific approach thanks to the ability to record annotations in association with a time line. If, the use of IWB seems to be natural in lecture, it can also be used in tutorial and practical class where the collaborative work is particularly important. The collaborative ability is closely tied to the softwares used. The main interest of this feature is the possibility to save and annotate every intervention while

[1] http://primtice.education.fr

preserving the logic behind the solving process of the course. This record allows each student to individually analyze the course when back at home. IWBs propose a wide range of pedagogical pertinent properties. Thus, they seem to be a good way of promoting the use of information technology in classrooms and of developing new pedagogical approaches. Thanks to its multimedia abilities, IWB must put in emphasize (distant) collaborative work.
By contrast, IWBs suffer from several drawbacks. First, the cost of such a device is important especially for heavy IWB with huge tactile screen based on resistive or capacitive technologiesn which retail at around $10,000. This implies a sharing of the devices between professors with specifically equipped rooms, reservation, etc. Some lightweight devices can be cheaper (i.e. around "hundreds of dollars"). These are generally based on optical technologies such as Microsoft Touch Wall [2]. At the extreme low cost level, game joypad[3] based solutions can be used but they suffer from an artisanal approach. Second, few IWB softwares are available on alternative operating systems such as Apple MAC OS X and Linux/Unix distributions. Finally, IWB are exclusively used in direct teaching (i.e. with students in the same classroom as the teacher). There are few functionality aimed at facilitate distant teaching and/or collaborative work. This last feature is the one we choose to deal with in this paper. To that end, a reactive multi-agent system has been developed.
Multi-agent systems are an efficient approach for problem solving and decision making applied to a wide range of applications. Among the classical models, the reactive approach is one of the most interesting thanks to their intrinsic properties and features such as simplicity, flexibility, reliability, self-organization/emergent phenomena, low cost agent design and adaptation skills, etc. Such systems rely on reactive agents, which are simple entities that behave based on their perceptions (Ferber, 1999). According to (Muller, 2004), the difference between a multi-agent system (MAS) and a classical problem-solving method, lies in the role and in the significance of the interactions that prevail on the definition of the agents themselves. Moreover, agents environment also plays a preponderant role as shown in (Weyns et al., 2005) and (Simonin and Gechter, 2006), since it is the main place where the system computes, builds and communicates. Indeed, in the reactive MAS framework, one single agent can neither handle a representation of the problem nor compute the global solution. Instead, this solution is obtained from numerous agent-agent and/or agent-environment interactions.
It has been shown that Reactive Multi-Agent System (RMAS) approach is efficient for tackling complex problems such as life-systems simulation (Parunak, 1997), cooperation of situated agents/robots (Drogoul and Ferber, 1993) and problem/game-solving (Drogoul and Dubreuil, 1993). As for the behavioral models, two main trends are generally used. The first is biologically-inspired approach such as (DiMarzo-Serugendo et al., 2004), (Bourjot et al., 2002) or (Brueckner, 2000). The second is the use of physics-inspired behavioral models as in (Gechter et al., 2006)), (Reynolds, 1987), or (Zeghal and Ferber, 1994).
The goal of this paper is to present a reactive agent solution to the collaborative work issue using Interactive Whiteboard. After a general overview of IWB principles, a presentation of the IWB software developed in the UTBM[4] is made. Then, next section focuses on collaborative work feature, describing in detail the reactive multi-agent system designed. Finally, the paper concludes with a description of future work.

2 INTERACTIVE WHITEBOARD PRINCIPLES

2.1 Principles

An interactive Whiteboard is generally associated with a computer, a data-projector and a projection screen. Depending on the solution used, one can use classical whiteboard pens (standard use) and/or electronic pens in order to write with electronic ink. IWB allows to make anything possible with a computer and more. IWB principle is very simple: (cf. figure 1):

1. Commands performed on the surface of the projection screen, are sent from the IWB to the computer. This link can be made thanks to wifi, infra-red or bluetooth, or with a wired solution (USB, serial, etc.).
2. Computer build the new image to be projected taking into account actions performed. This

[2] http://www.officelabs.com/projects/touchwall/Pages/default.aspx

[3] http://johnnylee.net/projects/wii/

[4] Université de Technologie de Belfort-Montbéliard — `http://www.utbm.fr`

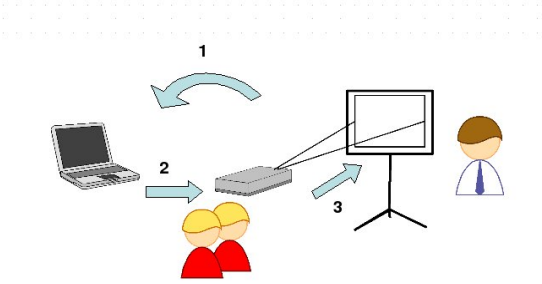

Figure 1: Classical IWB principles

image is then sent to the data projector with a wired connection (VGA cable, HDMI, cable, etc.)

3. The new computed image is sent by the data-projector on the screen. User can then interact with the new image returning on point 1. Some devices are also based on retro-projection or on LCD/Plasma screen. This avoids hot spot[5] and shadow problems.

Distant and/or collaborative work feature is close to the option available in the software used. This feature associated to visio-conference tools will bring the possibility to make distant lessons more attractive, dynamic and interactive. Distant collaborative IWB follows the same principles as the standard IWB with small adaptations (cf. figure 2):

1. Master IWB sends user's commands to its associated computer.
2. The master computer build the new image to be projected taking into account actions performed. This image is then sent to the data projector with a wired connection (VGA cable, HDMI, cable, etc.).
3. The new computed image is sent by the data-projector on the screen. User can then interact with the new image retuning on point 1.
4. The image computed in point 2 mixed up with the speaker image is sent through the network to another IWB's computer.
5. Distant computer sends the image to its associated data-projector.

[5]Hot spot occurs when the projection screen is not diffusing enough. In this case, the lamp of the data projector is highly reflected on one specific position on the screen(i.e. on the intersection between screen plan and data projector optical axis). This is generally hardly supported by the lecturer bringing some eye tiredness

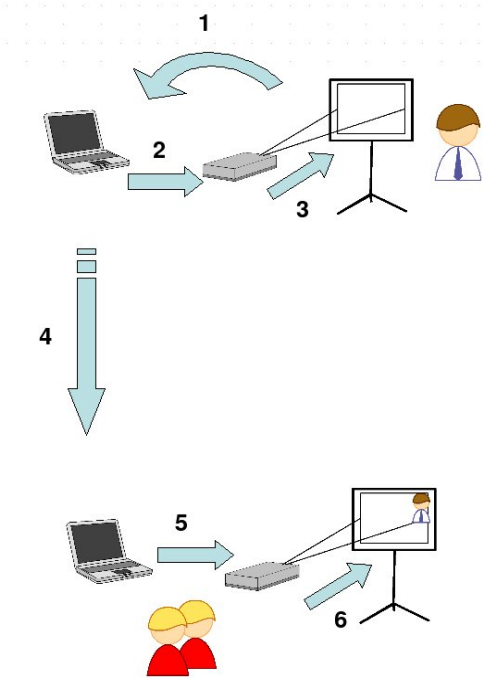

Figure 2: IWB for collaborative work principles

6. Data-projector sends the image to the screen.

This structure is the most widespread. However, we can also imagine that the distant device is also an IWB. In this case, students or another colleague can interact and modify what is sent by the master. In the case of several interacting devices, the master/slave relation is no longer useful. In the network, each IWB has to have the same priority and the same role as other. If one device decides to quit the network, it must be able to do this without any influence on other IWB. This propriety has been taken into account as a priority in the design of the presented proposal.

2.2 User's modes

IWB exhibits several user's mode depending on the type and the nature of the lesson. Here are the main employed modes:

- classical presentation,
- whiteboard/blackboard,
- interactive presentation,
- augmented presentation, and
- distant presentation.

2.2.1 Classical presentation

The classical presentation mode is the minimum requirement for a IWB software. On the software point of view, it is easy to develop and to put into practice except for the use of copyrighted format for save/load functions. Generally, it relies on a simple substitution of the mouse movement by the specific device (infra-red pen, fingers, etc.) adapted to IWB used. In this situation, multi-points detection is not necessary.

2.2.2 Blackboard/Whiteboard

This mode can be considered has the minimum feature for a IWB. In this situation, the user writes as he does with a standard teaching board but the regular ink is replaced by a virtual one projected by the data projector. Besides the classical features with whiteboard, IWB brings some new interesting ones such as : saving documents and pedagogical progressions, recognizing hand written script and scheme, building immediate course documents, etc. As for the programing/software engineering points of view, this mode is a little bit harder to develop. The user interface needs to be structured as a series of transparent layers. Consequently, this mode is really near the user's interface encountered with image processing softwares.

2.2.3 Interactive presentation

This mode corresponds to the classical presentation mode with a possible interaction with objects and components such as rotation, zoom, movie control, etc. In this mode, the multitouch ability can be really useful. Besides classical mode, one needs to be able to design and to include to presentation interactive components. Good examples of reference functionalities of interactive presentation mode can be seen through MIT (Sketch-Interpreting Software) and Microsoft (Touchwall project) shows.

2.2.4 Augmented presentation

Augmented presentation is a mix between classical and/or interactive presentation mode on one side and whiteboard mode on the other. As compared to classical presentation, annotations and modification of components are allowed. Everything can be save in one single format file. In this case, it is interesting to be able to load classical presentation file (such as .ppt or .pdf) as background layer and add on above transparent layers interactive components.

2.2.5 Distant presentation and collaborative work

This mode relies on the synchronization of two or more IWB. The functionalities are the same as a standard chat software with the possibilities to work simultaneously on the same document. This mode is perfect for distant working meeting or distant teaching since classical intervention interactivity is preserved.

3 INTERACTIVE WHITEBOARD APPLICATION

This section describes the overall architecture of the IWB application. Several modules are explained and the networking support is specifically detailled in section 4.

3.1 General Architecture

IWB Application is designed according to the Model-View-Controller (MVC) architecture (Reenskaug, 2003), which is currently considered to be a major architectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing and maintenance of each (separation of concerns).

Figure 3 illustrates the software components of the IWB application. Application controller is the core component. Its roles are to launch the other components of the application and to control the data and control flows between the components. Logic of the IWB application is embedded in the IWB Model, which is detailled in Section 3.2. Basically, this component provides documents, layers and interactive objects of the application. GUI module listen to changes in the IWB Model and give a graphical feedback to the User (see Section 3.4). User is able to interactively control the IWB application through physical devices (see Section 3.4). Corresponding device events are translated into IWB events and forwarded to the GUI and Actor Management components. This last components is composed of a set of runnable functions named actors. Actors should update the IWB model according to

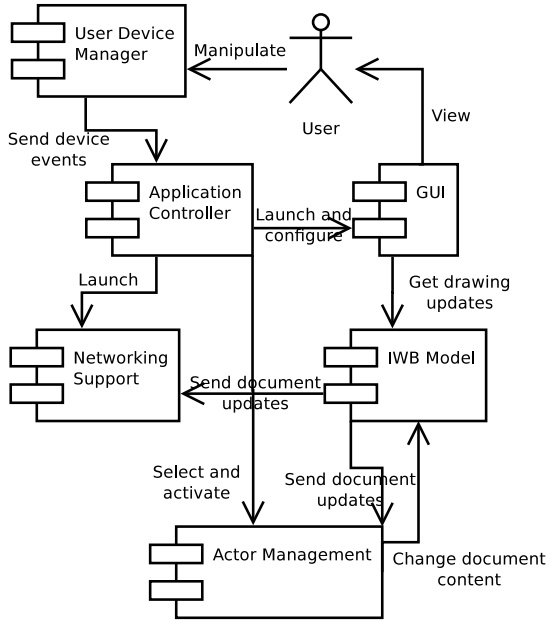

Figure 3: Component Diagram of the IWB Application

their goals and constraints (see Section 3.3). Finally to complete the overall architecture, Networking support is providing through an agent-based approach. This module is detailled in Section 4.

3.2 IWB Model

IWB model includes the "domain logic" of the IWB application. A generic logic model is provided by the IWB model, and it should be extended and specialized for domain-dependent applications (UML modelling, physic simulation, etc.). This core model is based on the concepts of document, layer and interactive component. Figure 4 illustrates the UML class diagram of this core model.

More formally Equation 1 defines the core model as the composition of a document D, a set of layers $\mathcal{P}(L)$, a set of interactive components $\mathcal{P}(C)$, and three functions h_1, h_2, and h_3.

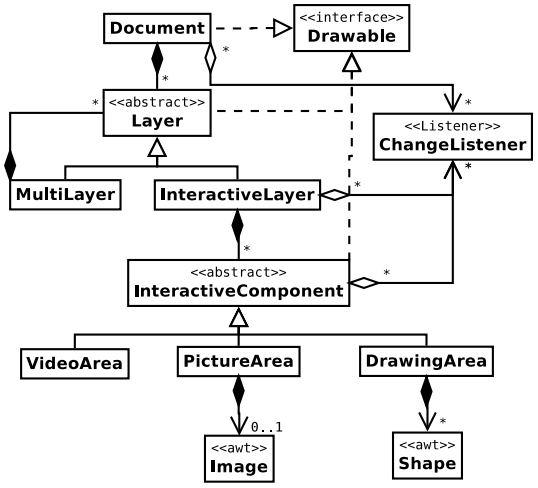

Figure 4: Class Diagram of the IWB Document Model

$$
\begin{array}{ll}
M ::= & D \times \mathcal{P}(L) \times \mathcal{P}(C) \times h_1 \times h_2 \times h_3 \\
D ::= & id \\
L ::= & id \\
C ::= & id \times r \\
h_1 : & L \twoheadrightarrow D \\
h_2 : & L \rightarrow L \\
h_3 : & C \rightarrowtail L
\end{array}
\tag{1}
$$

Document and layers are defined by their identifier id. Each interactive component has an identifier id and an associated resource r (video, picture, collection of shapes, etc.). Class diagram in Figure 4 contains hierarchical relations between the document, layers and interactive components. This relationship is formally defined by the surjective function h_1, the function h_2, and the injective function h_3. They map an object o to the object which is containing o. h_1, h_2, and h_3 denote respectively the layer-to-document, layer-to-layer, and component-to-layer relationships.

IWB Model is designed to be independent from any user interface library. In this way, it implements several design patterns (Coplien and Schmidt, 1995):

- Composite: layers are composed of layers, which are composed of layers in turn, etc.
- Bridge: IWB model is exhibiting a set of interfaces for high-level features and abstract roles: Drawable, Actionable, Movable, Selectable, etc.
- Facade: IWBEnvironment is representing all the IWB data model. It permits to manage

actors, update the data model, or retreive interactive actions.

- Observer: All the changes in the IWB data model are sent to event listeners, which want to be notified on.
- Strategy: IWB data logic and interaction manipulation logic are respectively embedded in the Document and InteractiveActor classes.

3.3 Actor Management

Actor is a runnable function which is updating the IWB model according to external events and their internal states. Each actor should define a a function according to Equation 2.

$$a : S \times M \times \mathcal{P}(E) \rightarrow S \times M \qquad (2)$$

Function a takes three parameters: (i) the internal state of the actor S, (ii) the state of the IWB model M, and (iii) a powerset of all the possible events E, which may occur in the application. Function a updates the internal state of the actor in one hand; and it replies a new state for the IWB model in the other.

Internal state of an actor depends on its goal. Actors are basically defined by statecharts in which transitions are traversable according to device events such as mouse button release or mouse motion. When a statechart's state of the actor is reached, an action on the GUI or on the IWB model is run. Let take the "Mover" actor. Its goal is to move an interactive component when the user clicked on it and then moved it (see Figure 5).

Actor management is modular and independent from the implementation from the other IWB application modules, except for IWB model. Create new actors for IWB application is simple and fast to tackle. For example, animation of graphical components according to simulation rules is done by a dedicated actor which takes only time progression events as parameters.

Figure 5 illustrates two standard actors in the IWB architecture: the selector and the mover.

Selector actor is an interactive process which is permitting to put an interactive component in a list of selected objects. This actor reacts when the user has pressed a mouse button. Mover actor has the role to move an interactive component at an other location on the IWB screen. It reacts when the user is pressing a mouse button and retreive the component under the mouse location.

```
class Selector extends InteractiveActor {
  void mousePressed(MouseEvent e) {
    InteractiveComponent c = getComponentAt(e.getPoint());
    getSelectionManager().select(c);
  }
}
class Mover extends InteractiveActor {
  Vector v;
  InteractiveComponent catched;
  void mousePressed(MouseEvent e) {
    catched = getComponentAt(e.getPoint());
    if (catched!=null) {
      v.sub( catched.getPosition(), e.getPoint());
    }
  }
  void mouseRelease(MouseEvent e) {
    if (catched!=null) {
      p.add( e.getPoint(), v );
      catched.setPosition( p );
      catched = null;
    }
  }
}
```

Figure 5: Example of Implementation of Selector and Mover actors — using Java-like Syntax

Then, when the user is releasing the mouse button, the actor is moving the component at the new position.

3.4 User Interface and Advanced Interaction Devices

User interface module provides a graphical feedback to the User. As illustrated by Figure 4, elements of the IWB model fire events when their content or their properties have changed. GUI is listening these events and update the graphical context. But even if this approach is very closed to MVC principles, it is not directly applicable to many rendering engine (Java Swing and Java 3D for example). Indeed these engines run an infinite loop on the screen rendering. Each time this loop is run, all the graphical screen must be drawn. To improve graphical performances of the IWB applications, GUI is not listening on IWB model events but invoke painting functions of Drawbable objects, and all the IWB model elements are Drawable (see Figure 4).

Design principles of the user interface permits to use any graphic interface library. For example, Figure 6 illustrates the use of Java3D to render the components inside a 3D virtual whiteboard. Only the menus and the toolbars must be implemented on this 3D library. The IWB model is simply a 2D picture put on a 3D plane. Of course, the choice of the graphical library is also related to its ability to provide a support for advanced devices.

In addition to the standard keyboard and

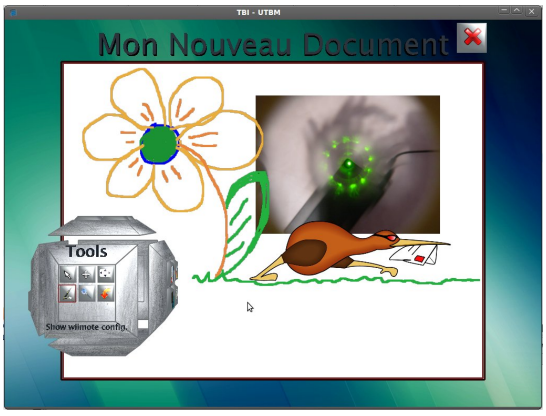

Figure 6: IWB Application Screenshot with Java3D User Interface

mouse, it is possible to use advanced devices to interact with the IWB application. These advanced devices are mainly used in lecture context: Wiimote®, LED pen, Dtrack®, etc.


Figure 7: LED Pen and Dtrack® Devices

Wiimote® uses of infrared light to locate the position of the device in space. LED pen, illustrated by Figure 7, is used with a webcam at the top of the screen to locate the position of the user. Dtrack®[6] is virtual reality tracking system successfully used by the IWB application.

[6]http://www.ar-tracking.com

$$tr_d : D \rightarrow \mathcal{P}(E) \quad (3)$$

All the device drivers are defined by a function tr_d in Equation 3. Function tr_d takes as parameter the state of the physical device. This state depends on the type of device but should contains at least a position (on the plane or in the space), and several indicators about the button's states. Function tr_d transforms the device's state d into a set of events $\mathcal{P}(E)$, which are comprehensible by the IWB actors. These events are directly forwarded to the actor under execution. In case of 3D devices, Function tr_d should project the 3D position on the plane of the IWB screen.

4 USING REACTIVE AGENTS TO MAKE WHITEBOARDS COLLABORATE

4.1 Problem overview

The common idea that first come out when it is decided to make several processes collaborate through a network is the use of a classical client/server architecture. In this education paradigm, one server, somewhere on the network, plays teacher's role. Clients connected to this server play student's role. Server/Teacher brings new document on the board, share them, draw comments. If they want to modify the whiteboard, clients/students must ask to the server/teacher to take the control of the board. This architecture works like the reality in teaching room, but with more flexibility and modularity thanks to the use of information technology concepts. Another approach is the classical blackboard collaborative structure. Most of the classical collaborative proposals found in literature are based on one of these methods (Tewissen et al., ; Komis et al., 2002; Aiken et al., 2005). By contrast to this, we decided to design a totally different architecture for collaborative work. This is based on a decentralized solution that stems from classical multi-agent system applications. In this case, each IWB is considered to be an agent able to modify the work of any other linked entity. There is no longer a leader/server and followers/clients, since every whiteboard play the same role on the network. Each agent has a copy of the collective work. This copy is updated thanks to its personal modifications and to other connected IWB modifications. Indeed, each

time one IWB changes something locally, modifications, i.e. events, are transmitted to its nearest neighbors [7], and then from one agent to the other all the connected IWB received the modifications. Besides, this architecture allows agent to connect/disconnect the IWB network in run time without involving perturbations in the system. When, one connects to the IWB network, it receives all the past modifications and update its own copy of the board. When this agent disconnects, it keeps its own copy that can be locally saved and stop all interaction with other IWB agents.

4.2 Agents' behaviors

Previous section presents a global overview of the problem. This paper will now focus on the details of this proposal. As already said, let now consider one IWB to be an agent. Each agent has its own board on which it is working. When connected, the behaviors of this agent are the following:

- Manage the list of nearest neighbors.
- Transmit each modification (i.e. action on interactive component) of its own board copy to its nearest neighbors.
- When one remote modification is received from one of its neighbor, update its own copy if necessary.

In addition to these behaviors, we can also add two transitional behaviors aimed at connecting the agent to the IWB network and quitting it. All these behavior are summarized in figure 8. We will now detail them.

4.2.1 Connecting the network

First of all, each IWB must be configured to be able to share information. The first IWB starts its server thread which is waiting for external connection. When a new IWB is coming, it has to know one IP address of one member of the IWB network. When only one IWB is already on the network its address must be known, but when there are more than one IWB, any IP address is sufficient. When connected, the newcomer receives all the modification event required to obtain the same working document as any other IWB on the network. This includes the background, interactive components state, relations between components, etc. Not all the information since the beginning of the network are sent.

[7] This notion will be detail in the next paragraph

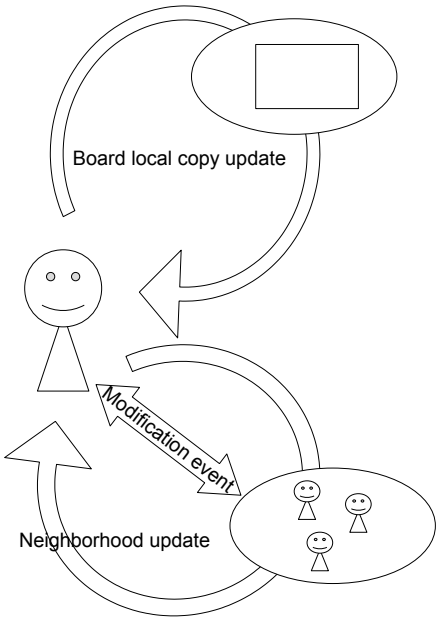

Figure 8: IWB Agent behaviors

Indeed, only the last informative modification are memorized. For instance, when one component is moved once and then moved again at another place, only the last place is memorized and then sent to newcomers. From now on, the connected IWB is able to collaborate.

4.2.2 Neighborhood management

When connected, each IWB agent starts to search for its nearest neighbors. For this, after a successful connection to the agents network, the newcomer ask the neighbor list of its access point (i.e. the agent, the IP of which has served to the connection). One neighbor is considered to be near one IWB when its time response is lower than others'. Thus, each agent will build incrementally a list of their nearest neighbors. This list is managed by a thread that updates neighborhood taking into account the time stamp of each received modification event. Using this strategy, each IWB is able to maintain up to date a list of the more efficient neighbors. This list changes in run time depending on the arrival/departure of IWB and on physical network traffic. In some cases, one agent can decide to remove one of its

neighbor, especially because this has a response time below all other agents in the neighborhood. It this case, a message is sent to the slower agent to retrieve its acknowledge. If this is positive, the agent is removed from the list, if negative it is kept. This is required to maintain links to every agent, especially when one of them is isolated. If the only link an agent had with the network is cut without asking, we risk to loose some agents in run time.

4.2.3 Modification transmission

Each time one IWB agent modify one interactive component, one event is generated. This event is composed by a series of values which corresponds to the modification performed on the considered component. Three different types of main remove event has been defined:

- *New Layer Event* : Creation of a new layer in the document
- *New Component Event:* Creation of a new interactive component such as hand writing area, picture, etc.
- *Component Modification Event*: Modification of the shape, size, position, etc. of one specific component.

Moreover, each event includes also a time stamp corresponding on the time/date of the modification. If the modification is the creation of a new component a special *new component event* is generated including all the necessary characteristics. For instance, if this new component is a picture, the picture and its characteristics (size, position, color depth, etc.) are integrated in a *new component event*. Each picture, independent to it initial file format is transformed into jpeg image when loaded on one IWB. Then each so build event is propagated, after serialization, to the agent neighbors. This transmission is made through classical TCP/IP sockets in order to ensure that all messages are received.

4.2.4 Updating with remove modifications

When one event is received, each agent rebuild the component and/or the list of the modifications performed by deserialization. Then, each modification/creation is performed on the local copy of the shared board. In parallel, the received event is also propagated to the receiver agent neighbors in order to ensure that all connected IWB received each event. If one event has already been received, it is directly trashed without propagation to neighbors. Indeed, there is no need to build a safety mechanism in the IWB network protocol since the data transmission is performed by the TCP protocol. We only have to ensure that each agent is able to receive each message wherever they are connected to the IWB network.

4.2.5 Quitting the network

When one agent decides to quit the network, it sends to its neighbors a specific message which includes the list of all of its neighbors. Then, when this message is received, each receiver updates its neighbors list by removing the quitting agent and considering new neighbors to fill the gap remained.

5 CONCLUSIONS

The Interactive whiteboard software exposed in this paper is currently passing tests in real classroom teaching conditions. It works on most existing computer platform such as Linux-PC, Windows-PC, Mac-OSX computer. It has also already been tested in collaborative research meeting with distant collaborators. The multi-agent system developed seems to be robust and to be able to manage several IWB without perturbation. We now wait for the return of the first experiments to improve the system, correct bugs and modify the ergonomic interface taking into account users advices. Despite its successful test, the application is still work in progress. We will soon change the interface to use the graphical libraries and 3D engine from the SeT[8] Laboratory. Besides, we decided to translate the application on mobile platforms including android based tactile devices and/or IPad. This will allow distant work between collaborators directly with their mobile devices (PDA, phones, tablet, computer etc.) even if they are in train or in other public transportation network. Finally, we also study the use of the Microsoft Kinect interface to improve the IWB user interface, in particular to avoid to use active infrared pen.

[8]Laboratoire Systèmes et Transports, Multiagent Team — `http://www.multiagent.fr`

ACKNOWLEDGEMENTS

The IWB platform is supported by the Voxelia SAS[9] company, France. The authors would like to thank Nicolas Gaud, Mikael Goncalves, Olivier Lamotte and Renan Zeo for their support and contributions.

REFERENCES

Aiken, R. M., Bessagnet, M.-N., and Israel, J. (2005). Interaction and Collaboration Using an Intelligent Collaborative Learning Environment. *Education and Information Technologies*, (10):67–82.

Bourjot, C., Chevrier, V., and Thomas, V. (2002). How social spiders inspired an approach to region detection. In *proceedings of AAMAS 2002*, pages 426–433.

Brueckner, S. (2000). Return from the ant : Synthetic eco-systems for manufacturing control. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

Coplien, J. O. and Schmidt, D. C. (1995). *Pattern Languages of Program Design*. Addison-Wesley Educational Publishers Inc.

DiMarzo-Serugendo, G., Karageorgos, A., Rana, O., and Zambonelli, F. (2004). *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*. Lecture notes in Atificial intelligence, n 2977 ISBN-3540212019.

Drogoul, A. and Dubreuil, C. (1993). A distributed approach to n-puzzle solving. *proceedings of the Distributed Artificial Intelligence Workshop, Seattle (United-States)*.

Drogoul, A. and Ferber, J. (1993). From tom-thumb to the dockers: Some experiments with foraging robots. *proceedings of From Animals to Animats II*, pages 451–459.

Ferber, J. (1999). *Multi-Agent Systems, an introduction to Distributed Artificial Intelligence*. Addison-Wesley.

Gechter, F., Chevrier, V., and Charpillet, F. (2006). A reactive agent-based problem-solving model: Application to localization and tracking. *TAAS*, 1(2):189–222.

Komis, V., Avouris, N., and Fidas, C. (2002). Computer-Supported Collaborative Concept Mapping: Study of Synchronous Peer Interaction. *Education and Information Technologies*, (7):169–188.

Muller, J.-P. (2004). Emergence of collective behavior and problem solving. In *Proceedings of Engineering Societies in the Agents World, ESAW 03, LNAI 3071.*, pages 1–21. Springer Verlag.

Parunak, H. (1997). Go to the ant: Engineering principles from natural agent systems. In *Annals of Operations Research*.

Reenskaug, T. (2003). The Model-View-Controller - Its Past and Present. In *Int. Software Development Conference (JAOO/GOTO)*, Aarhus, Denmark.

Reynolds, C. (1987). Flocks, herds, and schools: A distributed behavioral model, in computer graphics. *SIGGRAPH Conference Proceedings*, pages 25–34.

Simonin, O. and Gechter, F. (2006). An environment-based principle to design reactive multi-agent systems for problem solving. In *Environments for Multiagent Systems II, extended and selected paper from E4MAS'05 workshop.LNAI 3830*, pages 32–49. Springer Verlag.

Tewissen, F., Baloian, N. A., Hoppe, H. U., and Reimberg, E. "MatchMaker": Synchronising Objects in Replicated Software-Architectures. In *Proceedings of the 6th International Workshop on Groupware (CRIWG '00)*, pages 60–67, Washington, DC, USA. IEEE Computer Society.

Weyns, D., Parunak, V., Michel, F., Holvoet, T., and Ferber, J. (2005). Environments for multiagent systems, state of the art and research challenges. In *Post-proceedings of the first International Workshop on Environments for Multiagent Systems, LNAI vol 3374*. Springer Verlag.

Zeghal, K. and Ferber, J. (1994). A reactive approach for distributed air traffic control. *proceedings of Avignon94*, pages 381–390.

[9] `http://www.voxelia.com`