A Lightweight Agent System for Collaborative Learning Support

Anatoliy Doroshenko
Institute of Software Systems of NASU and Kiev Division of Moscow Institute of Physics and Technology,
Glushkov prosp. 40, block 5, 03187 Kiev, Ukraine, dor@isofts.kiev.ua

[1]Taras Kushko, [2]Stanislav Vonog, [3]Konstantin Zhereb
Kiev Division of Moscow Institute of Physics and Technology,
03650 Kiev. Glushkov prosp, 40, block 1,
[1]taras.kushko@mail.ru, [2]zadig@gmail.com, [3]cotan@mail.ru

Abstract: Collaboration and interaction features to support real time activities have become an important feature of many practices and education but often can not be used because expensive equipment and sophisticated software are unavailable in standard classroom environments. We propose an approach to achieve collaboration and interaction effects in relatively simple and inexpensive way by development of lightweight agent platform supporting collaborative learning applications called Inspirational Classroom Environment (ICE). The system is written in C# and uses Microsoft .NET platform and ConferenceXP product as the basis for the development of real time collaborative applications and rich multimedia content. A case study is carried out to use the ICE system in learning children by means of collaborative drawing, animations and personal tutor agents.

1. Introduction

Amazing achievements in wireless communications, Internet technologies and mobile devices enabled rapid developments of interactive education facilities in recent years. Research experience show that students can benefit significantly from their involvement in small learning groups [We95], increasing richness of applications features [Gr04], faster learning, greater retention, higher levels of motivation and interest and other findings of cognitive sciences [BM98],[Va97]. Especially, the incorporation of collaboration and interaction to support real time activities has become an important feature of many practices including education [Ha04]. However, these learning features are rarely used today because of lack of expensive wireless equipment and appropriate sophisticated software in standard classroom environments.

In this paper our main goal is to show that collaboration and interaction effects can be achieved in relatively simple and inexpensive way by development of lightweight agent platform supporting collaborative learning processes. The system is written for Microsoft .NET platform in C# and uses Microsoft's ConferenceXP [CO03] as the basis for the development of real time collaborative applications and transmission of rich multimedia content.

ConferenceXP- and Internet2-based architecture provides a flexible and extensible infrastructure for high-bandwidth interpersonal interaction. For example, it is known to be integrated into a lecture-style computer science course at the University of Washington [UW03] and other universities Along with development of ConferenceXP technology the Learning Experience Project [LE03] has been started as an initiative of Microsoft Research with a goal to explore how to make collaborative learning a compelling and rich experience by assuming the availability of emerging and enabling technologies, such as high-bandwidth networks, wireless devices, Tablet PCs, and the advanced features in Microsoft operating systems. ConferenceXP enables universities to build fewer infrastructures and concentrate on researching and developing collaborative applications that enhance the learning experience in and out of the classroom. However, to the best of our knowledge, Conference XP has not been used as an augmentation of agent-based systems.

There are two main contributions of our paper. The first concerns lightweight engineering features of the agent system ICE that is built as an object-oriented library of classes that allows easy usability, extensibility and customizability. Visual agents can be more bound with application than ordinary agents with graphical user interfaces (GUIs). In our platform the agent's graphical user interface can be a part of the interface of the main application.

Another main contribution is a case study in learning children by means of collaborative drawing, animations and personal tutor agents [MC04]. We have developed three sample applications (each one was developed in the form of a separate agent) to illustrate the potential of our platform: 1) Visual Thinking, 2) Exercises with Smart Tutor Agent and 3) Collaborative Drawing. Visual Thinking is an animation tool with a goal to allow children to quickly visualize their own thoughts by creating simple stroke-based animations. Smart Tutor Agent adapts the way the child learns and provides guidance and educational content in such a sequence, quality and quantity that best fit the pupil's personal learning style. Collaborative drawing implies different types of collaboration and allows children to draw simultaneously on a virtual whiteboard creating a new kind of digital artwork.

2. Collaborative Learning and Microsoft's ConferenceXP

Almost every real-world problem can hardly be tackled by a single person. Nevertheless, today's schools mostly teach students how to work on a single particular problem by themselves. This is good since the educational process has to ensure that the students develop certain basic skills. But this is bad since students often appear to be not prepared for real-world situations in which they have to communicate intensively and actively collaborate with their peers in order to do jobs.

With mobile, lightweight, wireless Tablet PCs students and teachers have the ability to share ideas in a more creative and convenient fashion. The lessons can be transformed into collective multiplayer team games. For example, pupils can break into several teams and participate in a creative role-based drawing session. The task

for each team is to draw a certain picture (the teacher provides the theme) in a limited amount of time after which the teams' drawing surfaces will become disabled. All participants have different roles. All of them have different tools: a black pen, colored brushes, an eraser, a bucket of ink (a team can run out of ink if they draw too much and too carelessly). The children draw simultaneously creating a new kind of digital artwork. So children have to communicate with each other in order to create a beautiful drawing together with different instruments in a limited period of time. In such way they learn how to apply collaboration techniques in practice.

ConferenceXP provides an extensible foundation for developing collaborative learning and conferencing applications. With it, you can interact and collaborate with others in a virtual collaborative space, called a venue. By enhancing the traditional method of teaching, ConferenceXP technology enriches the learning experience in several aspects [CO03]. What is used in our work is following:

- ConferenceXP is a peer-to-peer application, which means it sends data between ConferenceXP clients, instead of sending data to or receiving data from a server. To support simultaneous users while keeping network traffic to a minimum, ConferenceXP uses multicast to send data. By using multicast, a ConferenceXP client can send data once to all ConferenceXP clients set up to receive the data.
- ConferenceXP supports unicast for point-to-point communication and enables simultaneous users to send large pieces of data keeping minimal network traffic.

ConferenceXP is known not to be recommended in wireless networks where the packet loss can be more than 5% [COXP]. However the experiments have shown that the system based on the agent platform worked quite steadily over wireless network. In our ICE system ConferenceXP was used as a client application in collaborative drawings.

3. Inspirational Classroom Environment (ICE)

Today, many schools have classes equipped with traditional desktop computers. But progress does not stand on one place and in a couple of years there will be schools where wireless TabletPC computers will not be something unusual. This provides the basis for the required student equipment that will enable a collaborative learning experience. So the technology already exists that can make vastly improved learning systems routinely available. Expected improvements in technology have the potential to significantly reduce the cost and complexity of implementing learning-by-doing environments.

General view of our system called Inspirational Classroom Environment can be seen in the following Fig 1.

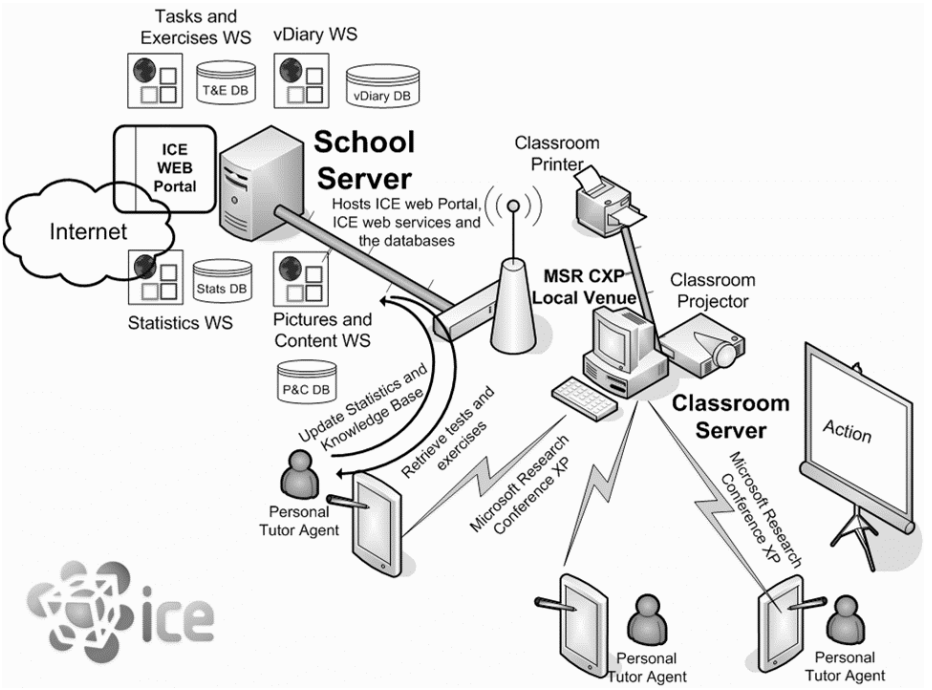

Fig. 1. Inspirational Classroom Environment Physical Design

ICE is a distributed application. There are three distinct client applications: teacher, projector and student. Inking on student's whiteboard appears on participating students' whiteboards. Teacher can use presentation capabilities to broadcast. The teacher's need to coordinate students' activities and see what part in collaboration takes each of the student led to some modification of ConferenceXP Presenter which is augmented with a generic agent system of FIPA-compliant architecture [FIPA] for user personalization and graphical interface. A number of Web-services (WS) to store and retrieve the information needed to support learning-by-doing functionalities are also developed. This allows third-party applications to be developed and used to augment these functionalities.

Capabilities of ConferenceXP Presenter were enhanced by adding new program filters for strokes and new drawing instruments to the program itself. Each stroke in ConferenceXP has a globally unique identifier (GUID) in the system. So if the student draws a stroke on his TabletPC the stroke is transmitted to each TabletPC used in the classroom and its GUID remains unchanged. It enables us to identify the stroke uniquely in the system. If we delete or hide a stroke on a TabletPC an RTStrokeRemove packet is transmitted to each TabletPC and copies of the strokes are deleted according to their unique identifiers. Their GUIDs must be equal to the GUID of the deleted stroke. We enhanced this capability by adding student's identifier to

the stroke. The teacher can filter strokes received from students' TabletPCs and see what kind of contribution each of the participating students made to the drawing.

The system is written for Microsoft .NET platform in Visual C# .NET. ConferenceXP can be used for the development of real time collaborative applications and for transmission of rich multimedia content. In our system ConferenceXP Presenter was implemented in a form of an agent with graphical user interface. The data is stored in SQL Server 2000 databases on a dedicated server. The information is saved and retrieved through specialized Web-services.

4. Agent system of ICE

ICE has been designed as FIPA-compliant lightweight agent platform using Microsoft .NET Framework to be used in various applications. Recently many FIPA-compliant agent platforms were known [LIST] and most of them are Java based. So the first significant feature of the ICE is its initial orientation towards .NET platform. This allows using many applications and libraries developed for this platform, in particular ConferenceXP. Other ICE distinctions are not so obvious and are governed by supposed usage of agent platform.

Many of known agent platforms (such as JACK [JACK] and ZEUS [ZEUS]) provide heavy support for intelligent agent behavior. These platforms are aimed at designing intelligent agents which can exhibit complex behavior both alone and as a part of agent system. Other platforms (such as JADE [JADE], LEAP [LEAP] and also ZEUS) focus on strict implementation of FIPA standards which allows interoperability between different agent platforms. In such platforms focus is shifted towards interactions within agent system rather than operation of single agents and they can be used to design agent communities that should benefit from interaction with each other. There is also the third approach to design agent platforms that considers agents as programming technique that facilitates development of complex distributed applications. Such platforms are better described in terms of recent OMG MASIF specifications [OMG] (an example is Grasshopper [GRAS]) and our agent platform follows this third approach.

The agent system of ICE is designed to facilitate development of collaborative applications such as collaborative learning activities. It takes care of all low-level technical details, such as establishing network connections between agents, finding specific agent and others. From developer's point of view, all agents are situated in single environment (that can be physically located on number of different hosts). They can exchange messages without limitations on their content or format and can use some platform services such as finding another agent or storing and retrieving their state on server. The system does not intend to implement any interaction protocols (e.g. specified in FIPA standards). Instead, developers in ICE should implement application-specific interaction protocols rather than adapt existing protocols for their needs.

Our agent platform lacks some features that are commonly used in other platforms (such as explicit support for tasks, conversations and even mobility) as these features are not required for our applications. Also there is no support for any kind of intelligent behavior. But even without them, we show that smart agents can be implemented independently using agent platform only for networking support (an example of such approach is our Personalized Tutor Agent). Also it becomes possible to keep our platform lightweight and easy-to-understand (class library written in 2000 lines of C# code includes 10 classes of about 50 public methods).

4.1. Agents Architecture

The main entities presented in the ICE are agents and agent managers. Agent manager corresponds to FIPA container and each agent manager can manage a number of agents. Different agent managers can be located on different hosts or on single host (in different application domains). Location of agent managers is not specified (with one exception for the main manager). Furthermore, it can change dynamically during operation of agent platform. Main agent manager has fixed location because it should be accessible from any agent manager in order to connect to platform. General view of agents' architecture can be seen in Fig.2.

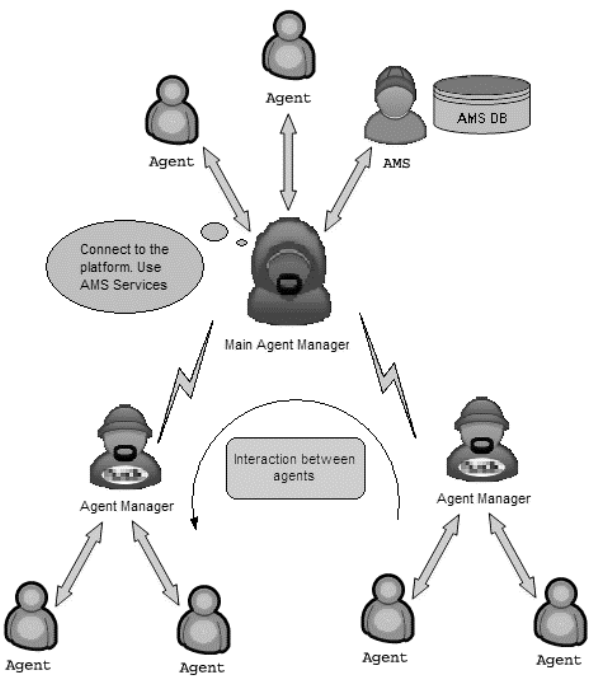

Fig.2 Agents architecture in ICE

The ICE agent platform provides its functionality through: main agent class, agent manager class, Agent Management Service (AMS) agent, configuration system,

graphical user interface (GUI) system. This functionality will be described in detail below.

The ICE agent platform is implemented as a library of classes where any class inherited from Agent base class is considered an agent. So all agents share the same functionality defined in this base class, namely:

- Unique agent identification through the whole platform. Each agent has Agent Identifier (AID) that has three fields: agent name, agent type and agent location (name of agent manager which hosts this agent). Each agent gets AID when it is created and preserves the same AID during its lifetime. Uniqueness of each AID is guaranteed by agent manager that creates it.
- Ability to receive messages. This is achieved by the following method:

  ```
  public virtual void Receive(Message msg).
  ```

  Each implementation of agent typically overrides this method to implement its interaction protocol. Default implementation just logs content of message for debug purposes.
- Ability to react to connection to platform. This is achieved by the following method:

  ```
  public virtual void StartNetwork().
  ```

There are also some helper methods that facilitate common tasks performed by agents.

Agents can exhibit both *reactive and proactive* activity though agent platform provides little support for implementing agent behavior. Reactive behavior is implemented in `Receive` method. There are two general methods of initializing proactive behavior. First is so-called *network-independent* initialization that is performed in constructor of concrete Agent subclass and is commonly used to initialize user's GUI. Second method can be used to implement *network-dependent* initialization which is achieved by overriding `StartNetwork` method and aimed to initialize interaction protocols with other agents and AMS. Network-dependent initialization requires active connection to the platform, while network-independent can be performed without such connection.

4.2 Agent management and configuration

Agent managers host agents and provide main services for them. These include:

- *Creation of agent* − agent name, agent type and name of assembly that contains this type should be specified.
- *Retrieving local reference to agent* − can be used for direct interaction with local agents.
- *Sending message* − this is main service that agent managers provide.
- *Providing AID of AMS*.

All other services are provided by specific agents. One such agent that is vital for functioning of agent platform is AMS (agent management service). AMS is physically an agent. But it can be logically treated as agent manager that holds actual information about the whole system. Our AMS performs the functionality of both AMS and DF (directory facilitator) from FIPA standard.

Agent manager that hosts AMS is considered main agent manager. Its location is fixed and known for all agent managers (this is achieved with the help of configuration system). All other agent managers connect to main agent manager and register themselves in AMS. After that they are connected to the platform. Actually AMS is not used to send messages – when sending message from one agent manager to another they establish direct connection. So if AMS (or main agent manager) is down, agents that already know each other can still communicate. But in order to get information about new agents (required to communicate with them) AMS is needed. In particular, AMS is used to get list of all agents of specific type or of all agents in specific location. An agent can also subscribe to AMS in order to receive information about specific events in agent system, such as agent connected to the platform.

In the ICE it is possible to specify parameters required for agent's operation through configuration system. In this way configuration of agent platform can be changed without any changes in code. If we have different agents with GUIs each contained in a separate dynamic linked library (DLL) then what we need to compose a complex application with all types of functionality of the assumed agents is only to specify which agents should be loaded. This can be done quite easily using configuration utility either in a form of a separate application or as a part of the main application. Thus it improves both customizability and extensibility of the ICE system.

Configuration data is stored in special XML file. It can be easily edited with configuration utility. Agents have access to this data through specific class. Configuration data is the same for agent manager and all agents it hosts. Configuration system stores information required for agent manager to run agents, namely: list of assemblies containing agent classes; list of agents to be loaded (agent type and preferred name are specified; if this name is already in use, name of agent is generated automatically); location of main agent manager (host, port); and preferred port to use (if it is already in use, free port with greater number will be used). Agents may also store their information using this configuration system.

Agents may have a rich user interface associated with them. It can be represented as independent forms or controls loaded into parent application. In ICE, parent application provides set of containers that can host various controls. Some containers host only one type of controls (e.g. image buttons), others can host any type of controls. The latter containers often use the whole screen space and controls loaded into them represent different applications. Only one control is active at any moment, and switching between them is performed with image buttons. Different agents load their controls into main interface independently, so they do not know about other agents' controls.

5. ICE Applications

We have developed three sample applications (each one was developed in a form of a separate agent) to illustrate the potential of our platform:

1. Visual Thinking
2. Exercises with Smart Tutor Agent

3. Collaborative Drawing

Visual Thinking

Visual Thinking is an animation tool based around Microsoft's Ink architecture for Tablet PC [MSDN]. The goal was to create an animation tool which could be easy and intuitive to use, and could allowed the user to quickly visualize his own thoughts by creating animations that can be in turn used as a prototype for moving GUIs, storyboarding movies, creating his own cartoons etc. It is developed as Windows control that could be easily integrated into any application using the .NET Framework. As Visual Thinking is designed for the Tablet PC and contains certain features only available on the Tablet PC it runs with little noticeable difference on almost all computers with the .NET Framework. Specifically designed for the Tablet PC Visual Thinking's user interface makes bringing your Ink drawings to life as easy as cut and paste. This means you can copy and paste frames to and from other documents allowing you to make animations *as easily as you can think them*.

As a component for other .NET applications, Visual Thinking can be used like any other control, and uses a few simple functions to manage it (like Play, Pause and the Time and UseEditor properties, which make creating your own animation editor or player easy). Also, the Visual Thinking control gives you access to an Ink object which represents the current frame being viewed, this allows you to perform any normal Inking operation just as you would on any other Ink object. Using the Visual Thinking control is recommended because it simplifies almost all operations to one or two simple function calls; however the architecture also allows making user's own animation controls with a series of structures similar to the Ink class diagram. For instance, you have StrokePath (to represent a Stroke through time), and InkPath (which represents a set of Strokes though time).

Visual Thinking was integrated into ICE platform in a form of a separate agent, so this capability can be easily incorporated by student's application. In order to draw an animation you must do drawings in a specific order (which most people do anyway) and then easily update a sketch to a new time and place. This animation can be easily transferred not only to any other TabletPC but also to the Classroom Projector where animations (visualized thoughts) can be viewed and discussed by every student. An example can be seen below in Fig. 3.

Figure 3 Initial and final drawings in Visual Thinking agent

Exercises with Smart Tutor Agent
The next application developed in a form of an agent is a Smart Tutor Agent that actively collaborates both with student, teacher and server that is used for storing tasks, educational material, statistics etc. The need for creating such an application is based on the fact that teachers are not able to devote much attention to every single student (since they cannot work directly with many pupils simultaneously); rarely can they afford to prepare individual tasks for every student; it is hard to track what is s happening in classroom and assess all pupils' actions. Checking tests and home assignments takes too much teacher's time; pupils can not receive immediate feedback on the activities and tests at the very moment they are completed.

Every child has a personal tutor agent − represented by an animated character − inside their Tablet PC. The agent adapts to the way the child learns and provides guidance and educational content in such a sequence, quality and quantity that best fit the pupil's personal learning style. As an additional stimulation, the creature changes its mood, behavior and appearance depending on how well the pupil performs. The agent helps the teacher motivate the pupil and performs a number of routine tasks for him. These tasks include the compilation of personalized tests, learning materials and home assignments for each pupil, gathering full statistics of pupil's results, and assisting and encouraging pupils and while they are doing exercises.

Collaborative Drawing
By using virtual collaborative space and multicast sending abilities of ConferenceXP we have created collaborative drawing application. Three types of application are used in collaborative drawing: teacher, projector and student.

Teacher divides students into several teams (two in the case shown in Figures 4, 5 and 6). The task for each team is to draw a certain picture (the teacher provides the theme) in a limited amount of time after which the teams' drawing surfaces will become disabled. All participants have different tools: a black pen, colored brushes, an eraser, a bucket of ink (a team can run out of ink if they draw too much and too carelessly). All tools are provided by teacher beforehand. Children draw simultaneously creating a new kind of digital artwork. So they have to communicate with each other in order to create a beautiful drawing together in a limited period of time. In such way they learn how to apply collaboration techniques in practice.

Figure 4 Teacher's interface

Figure 5 Projector's interface

Figure 6 Student's interface

6. Conclusion

In this paper, we have presented a new model for interactive learning experience based on Microsoft .NET platform and ConferenceXP technology. Based on these models, we designed a generic software framework for educational purposes called Inspirational Classroom Environment (ICE). Our results show that this framework can enable more students to be exposed to a comprehensive learning experience and increases the involvement of the teaching staff in the process of education. As mentioned earlier, this framework provides a technological basis for Real Time Interactive Collaborative Education. Technologically, we will measure the performance of the ICE framework under the current workload. Our next efforts will focus on the development of a generic software framework that supports a number of ICE-based classes. Moreover, new features will also account for the network condition as well as device profiles for different users.

By encouraging students to learn by doing, ConferenceXP technology augmented with ICE is a step towards new quality of education. ICE project is dedicated to enhancing the learning experience with extensible, customizable technology that supports learning in the classroom. We are looking forward to enhance ICE technology, including adding support for simulation and visualization applications, virtual environments, and games.

Acknowledgement. The authors are grateful for the help and support from joint-stock "Microsoft" (Russia) and Agent R&D Group at Moscow Institute of Physics and Technology

References

[BM98] M. Barak, T. Maymon, Aspects of teamwork observed in a technological task in junior high schools, Journal of Technology Education, 9(2):1-27, 1998.

[CO03] ConferenceXP**,** http://www.conferencexp.net

[COXP] http://www.conferencexp.net/community/Default.aspx?tabindex=12 &tabid=15#Does_ConferenceXP_work_over_a_wireless_network

[FIPA] FIPA Abstract Architecture Specification http://www.fipa.org/specs/fipa00001/

[GRAS] Grasshopper http://www.grasshopper.de/

[Gr04] W. Griswold et al. ActiveCampus: Experiments in Community-oriented Ubiquitios Computing. Computer. 37(10):73-81, 2004.

[Ha04] V. Hassler, Online Collaboration Products, Computer. 37(11):106-109, 2004.

[JACK] A. Hodgson, R. Roenquist, P. Busetta, Specification of Coordinated Agents Behavior (The Simple Team Approach), Agent-Oriented Software Pty., Ltd, Melbourne, Australia, http://www.agent-software.com

[JADE] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with a FIPA-compliant agent framework, Software - Practice And Experience. 31(2): 103-128, 2001

[LE03] The Learning Experience Project http://research.microsoft.com/research/pubs/view.aspx?tr_id=746

[LEAP] LEAP http://leap.crm-paris.com/

[LIST] http://fipa.org/resources/livesystems.html

[MC04] Microsoft Crowns Imagine Cup 2004 Champions http://www.microsoft.com/presspass/press/2004/jul04/07-06Champions04PR.asp

[MSDN] Microsoft Ink http://msdn.microsoft.com/library/en-us/tpcsdk10/lonestar/Microsoft.Ink/Microsoft.Ink.asp

[OMG] OMG Mobile Agent Facility Specification http://www.omg.org/technology/documents/formal/mobile_agent_facility.htm

[UW03] University of Washington Department of Computer Science & Engineering http://www.cs.washington.edu/masters/dl_tech/

[Va97] C. Van Boxtel, J. Van Der Linden, G. Kanselaar, Collaborative construction of conceptual understanding: interaction processes and learning outcomes emerging from a concept mapping and a poster task, Journal of Interactive Learning Research, 8(3/4):341-361, 1997.

[We95] N.M. Webb, J.D. Troper, and R. Fall, Constructive activity and learning in collaborative small groups, Journal of Educational Psychology, 87(3): 406-423, 1995.

[ZEUS] J. Collis, and D. Ndumu, Zeus Technical Manual. Intelligent Systems Research Group, BT Labs. British Telecommunications, 1999