ISBN: 978-989-8533-51-7 © 2016

A DISTRIBUTED INTELLIGENT E-LEARNING SYSTEM

Terje Kristensen
Bergen University College, Department of Computing, Mathematics and Physics
Inndalsveien 28, 5020 Bergen, Norway

ABSTRACT

An E-learning system based on a multi-agent (MAS) architecture combined with the Dynamic Content Manager (DCM) model of E-learning, is presented. We discuss the benefits of using such a multi-agent architecture. Finally, the MAS architecture is compared with a pure service-oriented architecture (SOA). This MAS architecture may also be used within E-health.

KEYWORDS

E-learning, MAS, JADE, DCM, Intelligent Tutoring, E-health

1. INTRODUCTION

Traditional classroom learning is mostly based on behavioural learning theories where the learner is the object of assessment. The teacher initiates the learning process and the learner responds. Another learning approach, constructivism, focuses on the learner's abilities to develop her own mental models and learning concepts [Kichin, 2000], [Hay, Kichin, 2000], [Kichin, Hay, Adams, 2000], [Novak, Canãs, 2006/2008]. This approach has more and more become accepted to be a more relevant method to promote learning, even at the university level.

E-Learning may be categorized into two different classes, *asynchronous* or *synchronous* learning [Graziadei, et al. 1997]. Classical classroom learning is an example of synchronous learning. The student has to stay in the classroom when the lecture is given, synchronized with the tutor and the class. If the students are not in the classroom at the same time as the lecture is given, they will miss the lecture. We now understand that the phrase "Learn from anywhere at any time" does not include synchronous learning. However, it includes asynchronous learning. The student does not have to be synchronized with the tutor or the class in order to participate or access the learning scenario. In a discussion group, users post their responses when they have available time. The course might require the student to read about a topic, and then do different activities before taking a test.

By use of Internet, people around the world may study on their own without having to worry about the time differences. By using asynchronous E-Learning the student and the teacher do not have to be in the same room or virtual chat room at the same time during the learning process. The student may fit to the study and at the same time being in a job situation. Being able to study at anytime across the Internet, the student may also reduce his travel time expenses. E-Learning is not just about delivering information and knowledge, but also how it is presented to the student.

2. AGENT TECHNOLOGY

Agent technology provides excellent methods for dividing problems into sub-problems and building component based software. Each agent operates independently and each sub-problem can be solved in a structured way. The agent framework infrastructure makes the agents to communicate and cooperate and the solutions of the sub-problems may be conveniently put together to form a global solution. A software agent acts autonomously within some environment. The software runs on a machine and the agent may perceive and act in the environment. For an agent to act it will need to have some actuators. An agent needs to perceive the environment in case of changes so it can act adequately. An agent designed to automatically park a car, needs to look at its surroundings, in order to identify obstacles, other vehicles or pedestrians.

Each agent acts on behalf of one or more clients. Because the agent acts autonomously, the client does not have to tell the agent how to do the task. He just needs to inform the agent of what he wants, to do it. For instance, Bob has delegated the task of retrieving the latest episode of his favorite podcast to an agent. He knows that his agent can act autonomously, so he does not need to explicitly tell his agent how to do it.

Autonomous agents decide what to do at run-time as opposed to having all the decisions hardwired when they are designed. The fact that software agents are autonomous and not hardwired, is one of the reasons why they differ from regular software. Agents also differ from objects by each agent has at least one thread of control, instead of using object methods. The agents are associated with behaviours to identify different models of the agents [Wenger, 1998]. The behaviours are ranging from the simplest model, where the agent only reacts to its environment, to more intelligent models of proactive agents that work towards a common goal or maximizing their utility.

2.1 JADE

The JADE framework provides an agent platform where the agents are situated. Each agent has to be instantiated in an agent container. The platform has a main container with an Agent Management System (AMS) agent providing the naming service for the other agents on the platform and a Directory Facilitator (DF) agent, with a *Yellow Page service*, helping the agents to get access to services provided by other agents. The first JADE instance automatically creates the main container and later instances create normal containers for the agents. JADE is implemented by using the FIPA standards [FIPA, 2013], and makes it possible for JADE agents to interact with any other agent framework implementing these standards.

A platform may have multiple containers, but only have one main container. The other containers can either run on the same machine or on other machines. This makes it possible to create distributed agent platforms. Agents living on the same platform can easily communicate with each other, and the MST (Message Transport Service) uses the Internal Message Transport Protocol (IMTP) for delivering messages. Figure 1 shows a platform with multiple containers, one main container and a set of containers running on other machines.

The IMTP requires full IP connectivity between the machines. This means that the machine where container 1 is running must be able to contact the main container on the server, and vice versa. The main container also holds three important tables, the Global Agent Descriptor Table (GADT), the Local Agent Descriptor Table (LADT) and the Container Table (CT). The CT contains information about all the containers connected to the actual platform, their names and addresses. In Fgure 1 we see three connected containers. Their addresses would be stored in the CT of the main container. Information about all the agents connected to the platform is stored in GADT. It also holds information about each agent’s status and location. The GADT is managed by the main container. This information is used when agents exchange messages with other agents on different containers.

2.2 Agent

The JADE platform takes care of managing and keeping track of the agents, as well as enabling the agents to communicate. However, JADE also contains an API for programming our own agents. Each agent runs in its own separate thread within a container. This means that a multi-agent system is also multi-threaded. Agents act by executing behaviours. To add the behaviour to our agent, we add an instance of this behavior by calling the *addBehaviour()* method with our new behaviour as an argument. There are also more complex behaviours available, and some of the more interesting ones are those that implement the different FIPA interaction protocols. The most used *FIPA Contract Net Interaction Protocol* can easily be implemented as a behaviour by extending a certain class. Sometimes it is difficult when we need to know if some information gained in one behaviour should also be available at a specific time in another one [Bellifemine, Caire, Greenwood, 2007].

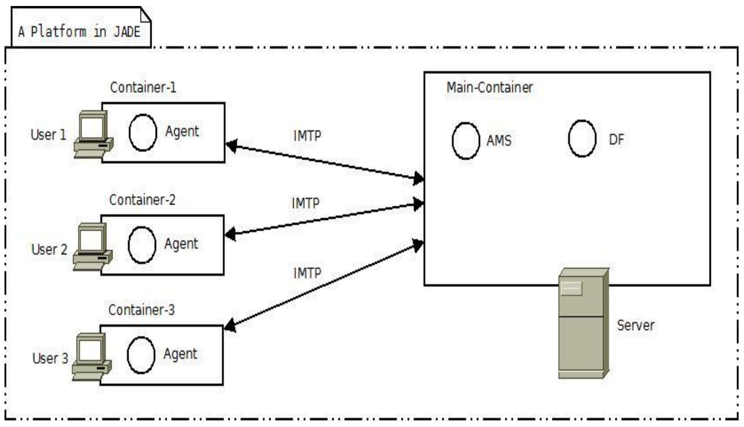

Figure 1. A platform with multiple containers

2.3 Agent Communication

Agents communicate by sending messages. Agents send messages by calling the send() method. A message between agents uses a special Agent Communication Language, ACL [Wooldridge, 2009]. An ACL message has only one mandatory property, a *performative*, identifying the communication type of the message. The use of performatives is closely connected to *speech-act theory* [Searle, 1969], [Austin, 1962] where communication is regarded as actions, and thus having the ability to make physical changes to the environment. A speech-act or performative is a verb describing it. A valid speech-act believing that the hearer has the ability to perform the action. Five classes of performatives are identified: Inform, request, promise, thanks and declaration. The performatives defined by FIPA represent a collection of standards which are intended to promote the interoperation of heterogenous agents and the services they represent. Only two performatives are needed in the system we have designed, INFORM and REQUEST.

2.4 Agent Ontology

For agents to be able to understand each other, they need to agree on the meaning of different concepts. An ontology defines the set of concepts within a specific domain. If an ontology is selected for a message, the conversations between all the participants have to know this ontology. The ontology includes a set of components that describes the concepts and their relations. Different concepts are described by class concepts, and they are ordered in a tree structure with classes and subclasses. Such concepts maybe described by a “is-a” relation as shown in Figure 2.

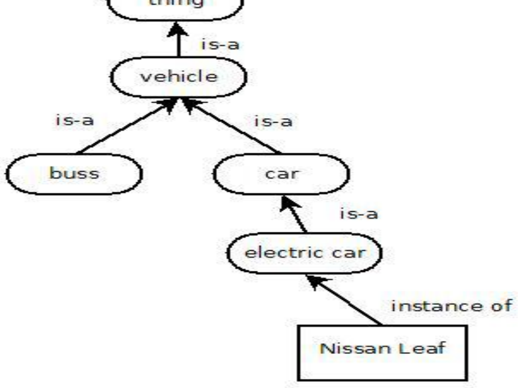

Figure 2. An ontology example

The agents use this structure to get information about the object Nissan Leaf. It finds that it is not a buss, but in fact an instance of an electric car. If the classes have different properties, the object Nissan Leaf would inherit from its parents. To access these properties, the agent traverses up the subclass relation tree to find all the properties of Nissan Leaf. Some classes will also have value restrictions on certain properties, for instance, who owns the car since we know that a car only may have one owner. These are mostly actions the agents may perform and concepts used to reason about their actions.

3. MOBILE AND TUTORING AGENTS

Our E-learning system has two specific types of agents, *mobile* agents and *tutoring* agents. A mobile agent can be used in different ways. Since mobile agents may move between computers, the agent may move to the data, instead of moving data to the agent. This is very efficient in cases where the data is much bigger in size than the agent. Another example is if the agent needs to call a lot of services on a server, then we can just move the mobile agent to the server to call it locally.

The tutoring agent is an agent created to help the student in the learning process. In learning of programming, a tutoring agent may, for instance, help students learning parameter passing in Java [Naser, 2008]. In our system we would like to connect tutoring agents to different activities. If a student have difficulties completing a task, a tutoring agent could give hints and help the student to solve the problem. An agent may also be used to decrease / increase the difficulty of the task depending on how well the student performs. By using tutoring agents that move to the client, we would have more resources available for the agent. This may sound strange. However, a student may only be running one activity at a time, so there will only be one tutoring agent running as a client at a given time, as opposed to tutoring agents running on the server. GUI may also be available to mobile tutoring agents through an interface. In a multi-agent environment, agents can delegate tasks to other agents. They work together in order to achieve common goals. Agents in such environments usually have different roles depending on their purposes. By defining several agent roles, the agents may be assigned to different skills and permissions [Wooldridge, Jennings, Kinny, 2000]. Agents may not only use their own skills, they may also be able to find an agent with the right skill and request its service.

3.1 Why use Multi-Agent System

Why do we want to design an E-learning system based on MAS? According to Wooldridge [Wooldridge, 2009], there are four factors that determine if an agent-based architecture is suitable to be used to design a system

- Dynamic or complex environments
- Agent as a metaphor
- Distribution of control, data or expertise
- As interface for legacy systems

The environment of our e-learning system is not very complex or dynamic. The system may contain one server and a set of clients connecting to it. In this case we do not need any distributed control since there is only one server. This system will not connect to any legacy systems either. In some environments it would be natural to model the entities as a society of agents and an agent-based architecture should be suitable. A commercial environment is a good example of such a society. The agents buy and sell, and compete with each other, just as in the real world. This is very difficult to model in an object-oriented system.

In an intelligent tutoring system the agent is also a natural metaphor to use. Different types of activities would need different kinds of tutoring agents, each one with a certain ability. The tutors are modelled as mobile agents. When a student accesses an activity, a tutor is transferred to the student in order to help her with the actual problem. A multi-agent system may also be used because it is flexible, extensible and fault-tolerant [Shenghuau, Kungas, Matsin, 2006]. Agents are autonomous and are designed to react to its environment. The more intelligent they are, the more flexible they should be.

By creating an E-learning system as a MAS it should be more scalable. Distributed systems can easily be scaled by adding more nodes. The possibility of creating mobile tutoring agents, makes it possible to transfer an agent to the student to help him to do some activity. This implies that the load of the processing is moved from the server to the individual clients. We believe that an E-learning system designed as a MAS will be more extensible, fault tolerant and scalable than a conventional system. In addition, by using also interface agents we may tailor a special interface to the user.

So why are we using agents? The brain is a highly parallel dynamical system able to solve a wide range of tasks. Neuroimaging has shown us that different brain areas are active during execution of certain tasks. So we may argue that the learning process itself is highly parallel and similar to biological learning [Hodgkin, Huxley, 1952], [Kristensen, McNearney, 2013]. By using a sequential model it is difficult to simulate such independence since the only way for interaction to happen between objects is by letting one object performing an operation on another. Autonomy, reactivity and social abilities are important properties of software agents. These properties are also important properties of the biological neural networks in the brain when we are learning [Kristensen, Johansen, 2006]. Biological neural networks in the brain are autonomous to a certain degree. We may then argue that that such a software model therefore also has a biological foundation.

4. THE DCM MODEL

The Dynamic Content Manager (DCM) model is a new pedagogical model for E-Learning that focuses on concept maps and reusable learning objects [Kristensen, et al., 2006-2011]. The model is designed to make it easier for teachers to collaborate when creating learning material, but it also gives the students a more flexible learning experience. Together with the idea of intelligent tutoring agents, it should be possible to create a system that will give the students more flexible learning experience. It may then be natural to design the system as a MAS architecture since we want to include tutoring agents into the learning model. The most important property of the DCM model is the ability to reuse learning content. For content to be reusable, it is important that the content is separated from the presentation. A web application, where the learning content is created as HTML files, is an example where reusing content may be difficult. The reason for this is that an HTML file contains both the content and the code how it is going to be presented to the user. If we want to reuse this learning content in another application that does not use the web pages, we would have to manually strip out the content from the presentation within the HTML file. PowerPoint is also often used as a rapid E-Learning tool to quickly create learning content [Kuhlmann, 2012]. But again, the content is coupled with the presentation in a PowerPoint file. This makes it difficult to reuse the content.

The DCM model solves this by storing the content in a content unit, a collection of resources used to teach a specific theme. The content unit is a reusable element and is not tied to any form of the presentation to be reused in different courses and situations. The business logic of the application decides how to present the content unit. Another property of the DCM model is that the learning process itself is modelled by the content unit. This is done by creating a map that guides the student through the different resources and evaluations. In the DCM model the learning maps represent courses. The possibility of having multiple paths in the learning map, gives the learner the flexibility that the contents of a course may have different approaches. The DCM model consists of four important concepts:

- Content unit
- Learning map
- Knowledge map
- Student map

In our system the agents will have to know different concepts of the DCM model. All the domain specific knowledge needs to be part of the ontology. The agents should know that a learning map contains a set of content units and the way they are organized. However, the agents also need to know about some application specific ontology, described by an ontology language.

4.1 Content Units

The content unit is an atomic unit of knowledge element, often also called a learning object. Each content unit will help the student to acquire knowledge about a specific theme. A content unit contains a combination of resources and evaluations. These resources can be anything the teacher finds useful to represent as knowledge of a theme, and is used to teach the student important aspects of a specific field. This could be explained in a text, an image, a video or a combination of them. However, a resource could also be an activity that the student has to do. Activities can be anything from small exercises to small games that simulates concepts while requiring input from the student. A content unit should also have evaluations. These are used to assess the student, to see how much she has learned of the actual content unit. Each evaluation has a value to describe how much it counts on the final score of the content unit. Content Units consists of:

- *Theme(t,w,v,pv):* theme t, weight w of the content unit. The version v and the version of parents
- *Resources:* a list of resource elements. Resources have a name and a type.
- *Evaluations:* a list of evaluations, where each evaluation has a name, weight and a type. The weight defines the importance of the given evaluation compared to the other evaluations of this content unit.
- *Prerequisites:* a list of prerequisites, describing the paths between resources and evaluations within the content unit.

The prerequisites list acts as edges between the resources and evaluations. This may be visualized as a map. This map of resources and evaluations corresponds to the learning process of the actual content unit. The learning process itself may be modelled in the DCM model. It may then be natural to place the pedagogical philosophy of the teacher in the content unit.

4.2 Learning Map

In the DCM model, courses are represented as learning maps. A learning map contains content units that the students will access to get knowledge. The teacher design the pedagogical philosophy used in the course by adding prerequisites between different content units. This may then create a map structure with a set of paths between the content units. Learning maps may have multiple paths, so the student may select optional ways to learn. The map structure also limits the available new content units of the user. Available content units depend on what content units the user has already completed. By finishing one of the available content units, new paths will be opened in the map, and new content units will be available for the student to access.

To give an example: if theme B requires that the student knows about concepts explained in theme A, it is the teacher’s job to design this into the learning map so that the student does not begin on theme B before theme A. This is done by setting theme A as a prerequisite for the theme B. By using the   combinations of prerequisites and content units we have created a learning map. The pedagogical philosophy the teacher wants to emphasize in the course, is then expressed by the map structure created by prerequisites between the content units. A Learning maps consists of

- *Course(n,w):* the course name n and the weight w of the learning map
- *Content units:* a list of the content units used in this learning map
- *Prerequisites:* a list of prerequisites describing the prerequisites between the content units

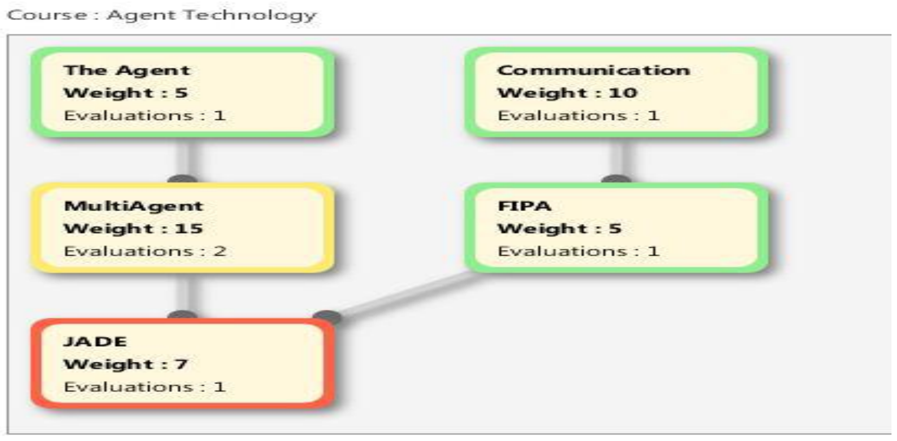

Figure 3. A Learning map example

We have decided that only other content units can be made prerequisite to a given content unit. The reason is that content units are atomic units and should be used as a whole. If we only want to use parts of a content unit, we should create a new content unit. By storing the content unit as a new version unit the original one is left intact.

A learning map is basically a graph consisting of nodes and edges where the content units are the nodes and the prerequisites of the content units are the edges. The edges make the different paths a student may select. A content unit without any prerequisites is a source node in the graph. A learning map needs at least one source node. Otherwise the student cannot access any content units. In addition the graph cannot contain any cycles. The maps have to be directed acyclic graphs (DAG) to make sense. In Fgure 3 we see that there are two source nodes. They are both colored green, meaning that they have both been completed by the student. In this example the content unit “FIPA” has also been completed. The only available content unit is MultiAgent with weight 15. The student has to complete two evaluations before making JADE available. In addition, showing completed, available and unavailable content units, the system also shows the prerequisites in the map. In this learning map, the content unit “Communication” is the prerequisite of “FIPA”.

4.3 Student Map

The knowledge map contains all the available knowledge of the system. The map contains information about how the content units are structured in the courses. A key point of this map is to make it easier for teachers to reuse content units. When creating a new learning map the teacher can add previously created content units from the knowledge map to create the new learning map. If there is a content unit related to a specific theme a teacher wants to use, she may find it in the knowledge map instead of creating a new content unit from scratch. As more content units are created, the knowledge map will grow bigger. In a large knowledge map it may be difficult to find the most appropriate content units that we are looking for. To solve this problem, it should be possible to filter the map by searching using different keywords. The knowledge map could also be visualized as a map to make the structure clearer to the teacher. The content units could be displayed in the same way as in a learning map. However, this may also become a problem when content units are used in a lot of different learning maps. The map would then have a lot of edges, and it will be difficult to get any useful information from it. It is therefore highly important that there is a way to visualizing the importance on the edges between different content units. We could solve this by adding a weight to the edges. Such a weight could for instance depend on the relative frequency the edge occurs in the different learning maps of the actual content unit. Edges with low weights could then not be included in the learning map, and by making the edges thicker one could illustrate their importance. When a student is taking a course we may want to know his progression, the score on evaluation tests, which questions she failed on and the path of the student through the learning map. The progression of the student is important for both the teacher and the student. The student wants to see what courses she has completed, and also what content units are available next time. The teacher might want to look at the overall progress in a course, to identify if some students are falling behind. The scores on the evaluations could for instance count as the student’s grade of a course. This depends on how well the student has been doing (scored), the weight of the evaluation and the type of the content unit. However, student maps may also be used to identify possible problems in the course. If all the students get a very low score on some evaluation, the teacher should make the resources of the content unit more clear and informative.

To identify what problems of a content unit, the teacher may observe which questions the students failed on. Courses may have multiple paths. If for instance, the content units have been presented in the wrong order we may have a didactical problem. By combining information about the paths of the student and their scores in a learning map, the teacher may be able to identify the problem. This means that the student map is also an important tool for the teacher, in order to improve the teaching of a course. A student map keeps track of the progression of a student in a course. There is one student map for each course a student is taking. A student map contains information about the resources and completed evaluations with all the answers on a test taken by the student. Each student map consists of:

- *Student:* the map of an actual student
- *Course:* the course to which map it is related
- *Completed content units:* an ordered list of the content units the student has completed
- *Completed resources and evaluations:* an ordered list of completed resources and the answers of the on the evaluations of the content unit

5. THE DISTRIBUTED MODEL

The DCM platform may be distributed, with multiple containers running on other locally connected computers. This enables us to scale the system horizontally by adding more computers, each containing a set of agents. These agents are registered with the main container and communicate with the AMS and DF agent through an intra-platform communication using the IMTP (Internal Message Transport Protocol) [Bellifemine, Caire, Greenwood, 2007]. When a user requests a service, the DF will provide a list of agents capable of carry it out. Some of the agents will be running in added computers and may therefore reduce the load of the agents on the main container. An important requirement of the system is that it should be easy to access the user services. The functionality of the system is defined by the different services provided by different server-agents. The user may request one of these services. The server-agents are designed using the Gaia methodology [Jennings, Wooldridge, Kinney, 2000]. We analyzed and identified some of the important roles in the system by using the roles model given in Gaia. For the system to be able to handle create, read, update and delete (CRUD) operations on the different concepts of the database we defined database handler roles. By using GUI a user may interact with the DCM system and may be able to create and take courses by using this client. Users may access the client in different ways. We believe that the system is more user-friendly when the users access the system through a web page. To start the client application, the user first click on the “Start DCM Client” link where the application is available provided by the Java Web Start framework. The DCM system does not have its own a web server, but colleges usually have their own, from which the client is available. Another option is to download the client and use it as a standard desktop application. The client consists of two parts, the GUI and the agent container. The username is also used to identify the agent platform.

5.1 The Client Agents

The client should be easy to use. This quality has been expressed in multiple use case scenarios. A navigation tree has also been developed, so the user may be able to quickly navigate between learning maps, content units and resources. The navigation tree may be used to select the different courses as well as navigate directly to a specific resource or an evaluation of a course. A learning map may be expanded to show all the content units in the navigation tree. The user can then click on the desired content unit in order to view it. In Fgure 4 we see that the user has expanded the course Agent Technology given by the five themes of the course. The theme Communication is also expanded, and two resources are here available. The user may now easily select one of the content units or one of the resources in the Communication theme.

The GUI displays the learning maps, content units and resources and handles input from the user. We have used the MVC (Model, View, Controller) pattern to create the GUI. Figure 4 shows an overview of the client. The GUI consists of views and controllers. As an ontology language we have used Protégé [Wooldridge, 2009]. The models consist of different ontology classes generated from Protégé such as LearningMap, ContentUnit, Evaluation, etc. The views are responsible for showing content, text and images, one view for each of the different models. The GUI also consists of buttons and other controls so the user may interact with the system in a nice way. The clientAgent is also responsible for updating the GUI when it receives information messages from the server. If the system is extending to handle mobile tutoring agents by using an IPMS (Inter-Platform Mobilitty Service) plug-in, these agents are transferred to the client. When a user is starting an activity, the client receives a corresponding tutoring agent. The agent will then run inside the client's agent platform where it will support the user doing some activity. The mobile tutoring agent will interact with the GUI using an interface specifically designed by the tutoring agents.

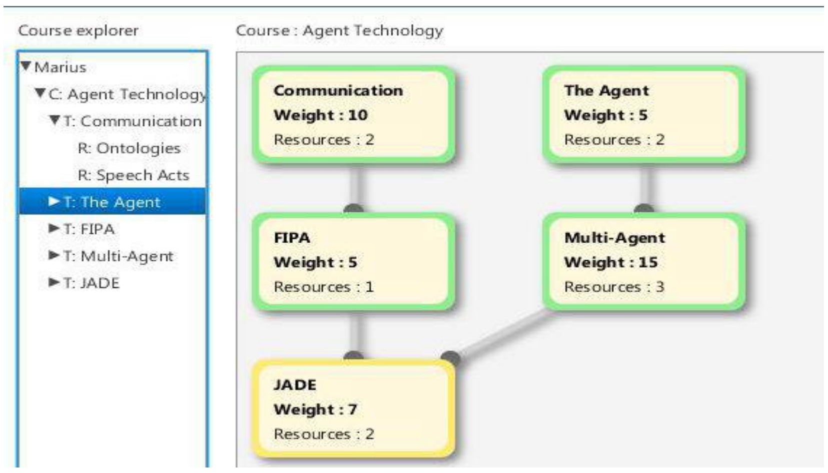

Figure 4. GUI screenshot - viewing a course / learning map

In Fgure 4 the user has selected the Communication content unit, and is now viewing the map made of the two resources. An interactive graphical view of the maps makes it easier to understand the structure of a course and the learning process of a theme. Both learning maps and content units are displayed by the GraphView tool of the client. The colours show the difference between completed, available and unavailable content units/resources. The user may also navigate by clicking on items in the map. When clicking on a vertex it will open the view to display the content of that model.

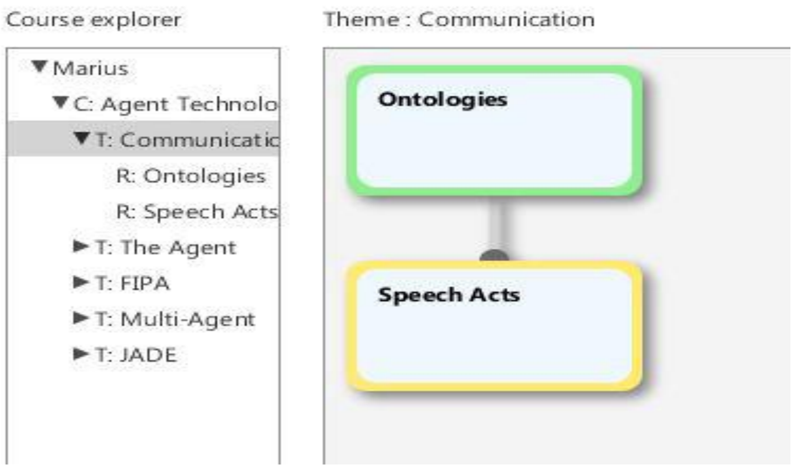

Figure 5. GUI screenshot - viewing a theme/content unit

6. CONCLUSION AND FUTHER WORK

In this paper we have focused on creating a new architecture of an E-learning system based on the DCM model and MAS. It is shown how the Knowledge Map may be used to structure the content of a course. The approach is flexible in respect to organising the content of a course. The content includes learning resources, practical tasks and evaluations. The Learning Map is created by the professor to design a course. It describes a selected scenario as a path through the content. The actual Learning Map may vary between different professors, even in courses with the same content and syllabus. This is due to the individual didactical understanding of each professor. The Student Map is created by the system, based on the results and weights of the Evaluations. It represents a model of the learning progress of individual students taking the course. The Student Map may be used by the educator to monitor the learning progress. A platform is created for e-learning as a distributed system by combining such a model with MAS.

In a Service-Oriented Architecture (SOA) [Pautasso, Zimmermann, Leymannn, 2008] we add more services to create new functionality. Systems using such an architecture are easy to extend. A SOA application usually creates web services that the client may access via a HTTP connection. Since all interactions are initiated by the client of the service, the result of the service is sent in response to a HTTP message. This means that we can easily access web services from behind a gateway. However, by using of MAS architecture we have to initiate a conversation between a pair of agents. In our system we have used the *FIPA SL Content Language* [FIPA, 2013]. We create messages that conform to the FIPA SL Content Language. As far as the agents have access to the same ontology, the agents connected to the server understand the messages, no matter from which platform they were sent. A SOA system may also have the property of loose coupling. There are different ways for a system to be loosely coupled. Web services make the system loosely coupled with respect to time/availability, location and service evolution. A Multi-agent system also has the same properties. We may easily modify and develop services that agents provide without directly affecting the clients that request the services..

6.1 Further Work

More time is needed to implement optimal functional requirements of the system. The functionality we have developed has been related to create learning maps and content units. The prototype lacks functionality with respect to the scores of the students, the courses taken and their evaluations. We then need to focus more on the implementation part of Student maps and add agents that provide services, related to the statistics of the courses the students have taken. An important aspect of this E-learning system is the creation of mobile tutoring agents related to the individual learning process of the student. This is a very interesting and important aspect and is domain specific. The students also need to create their own maps to describe their conceptual understanding of the subject and to create their own teaching scenarios. In this way the students gain experience with (Meta) modelling. By using a system as DCM, based on Knowledge map, Learning map and Student map, one adapts the e-learning system to the learning process itself, in contrast to traditional learning management systems where the learning process must be fitted to the management system. Such an architecture may also be used to design an intelligent distributed E-health system of autonomous interacting components (agents). In this case, the knowledge repository may consist of atomic knowledge elements taken from the health sector. The learning map may then indicate different ways a patient may be treated. The tutoring agent in this case may be a patient agent following the individual patient.

REFERENCES

Book

Austin, J.L. 1962. How to Do Things with Words. Oxford University.

Bellifemine, F., Caire, G., Greenwood, D. 2007. Developing multi-agent systems with JADE. Wiley.

Graziadei, W.D. et al. 1997. Building Asynchronous and Synchronous Teaching-Learning Environments: Exploring a Course/Classroom Management System Solution. State University of New York, NY USA.

Novak, J., Canãs, A. 2006/2008. The theory underlying Concept Maps and how to construct them, Technical report, IHMC CMaps Tools, Florida Institute for Human and Machine Cognition, USA.

Searle.J. 1969. An Essay in the Philosophy of Language. Cambridge University Press. 203 pp.

Wenger, E. 1998. Communities of Practice. Learning,, Meaning and Identity. Learning in Doing: Social, Cognitive and Computation Perspectives. Cambridge University Press, UK.

Wooldridge M. 2009. An Introduction to Multi-Agent Systems, Wiley.

Journal

Hodgkin, A.and Huxley, A. 1952. "A quantitative description of membrane current and its application to construction and excitation nerve". In J. Physiol., 117, pp. 500-544.

Hay, D., Kinchin, I.M. 2000. Using Concept Mapping to Measure Learning Quality. Education & Training, pp 167- 182.

Kinchin, I.M, 2000. Using Concept Maps to Reveal Understanding: A Two-Tier Analysis, In School Science Review. 81:296, pp 41-46.

Kinchin, I.M, Hay, D., Adams, A. 2000. "How a qualitative approach to concept map analysis can be used to aid learning by illustrating patterns of conceptual development, in Educational Research.

Kristensen, T., Dyngeland, M., Bech, Ø. 2013. Towards a Multi-Agent E-learning Platform. Journal of Computer Engineering and Informatics. Volum 1.(2) pp. 64-81, Hong Kong, China

Kristensen,T., McNearney,D. 2013. Simulating Spiking Neurons by Hodgkin Huxley Model. International Conference on

Bioinformatics & Computational Biology, BIOCOMP'13, July, Las Vegas, CSCREA Press, 2013, USA.

Wooldridge, M., Jennings, N.R, Kinny, D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems.

Conference paper or contributed volume

Kinney, D., Georgeff, M., Rao, A. 1996. A methology and modeling techniques for systems of BDI agents. In Van de Velde, W. and Perram, J.W. Editors, Agents Breaking Away. Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in Multi-Agent World, pp.56-71. Springer verlag, Berlin Germany.

Kristensen. T. 2011. The Dynamic Content Management System. IEEE Explorer Digital library, pp.1-8, 2011.

Kristensen, T, Johansen, J. 2006. Simulation of Attentional Networks in the Brain – an Agent based Approach. Proceedings of European Simulation and Modelling Conference, ESM'2006. Toulouse, France.

Kristensen, T., et al. 2009. Dynamic Content Manger – a New Conceptual Model for e-Learning. In Proceedings of International Conference on Web information System and Mining,WISM'09, Shanghai China, Springer Lecture Notes in Computer Science.

Kristensen, T., Hinna, K., Hole, F.O, Lamo, Y. 2007. Different E-learning paradigms - a Survey. In Proceeding of MIT-LINC (MIT Learning International Network Consortium 4.th International Conference: Technology-Enabled Education: a Catalyst for positive Change), Amman, Jordan.

Kristensen, T., Lamo, Y., Mughal, K., Tekle, K..M., Bottu, A.K. 2007. Towards a dynamic, content based e-learning platform. In V. Uskov, editor, Computers and Advanced Technology in Education. ACTA Press, pp 107–114, Beijing, China.

Kristensen, T., Lamo,Y., Mughal. K. 2006. E-learning Systems in the Bergen Region, Norway - an Overview, Proceedings of the IRMA International Conference, Washington DC, USA.

Naser, S.A. 2008. An Agent Based Intelligent Tutoring System for Parameter Passing in Java Programming. Journal of Theoretical and Applied Information Technology.

Shenghuau, L., Kungas, P., Matskin, M. 2006. Agent-Based Web Service Composition with JADE and JXTA. International Conference on Semantic Web and Web Services.

FIPA ACL. 2013, (http://fipa.org).