Cooperative Agent-Supported Learning with WeLearn

Michael Sonntag, Susanne Loidl-Reisinger

{sonntag, sreisinger}@fim.uni-linz.ac.at
Johannes Kepler University Linz, Altenbergerstr. 69, A-4040 Linz
Telephone: +43(70)2468-{9330, 8437}
Fax: +43(70)2468-8599

Topic areas: Online learning platforms, distance education, cooperative learning, intelligent agents, metadata, agent-oriented software engineering

Cooperative Agent-Supported Learning with WeLearn

Abstract: *Distance Education sometimes suffers from the problem that the environment is powerful, but also complicated to learn and to use. This detracts from the learning process and poses and initial barrier against widespread acceptance. Also, creating teaching material (and holding courses) is more difficult and time-consuming than in conventional form because of new possibilities and additional user expectations. We propose the integration of intelligent agents into learning platforms to ameliorate some of these problems by automating routine tasks and creating added value by themselves (e. g. creating new navigational paths or offering personalised simple coaching). These agents cannot and should not replace coaches, but rather ease their tasks and fulfil additional ones, which were not doable before. The importance of metadata for agent-integration is also touched briefly accompanied by a discussion of the applicability of agent-oriented engineering for online learning platforms.*

Introduction

Currently, Distance Education (DE) is in a kind of intermediate stage: The initial technical problems have been solved, first standards are emerging, but widespread adoption has not yet taken place and also seems not likely for the immediate future. One problem is the high cost of creating good learning material. As e. g. electronic materials allow much more freedom in navigation (hyperlinks) and more types of media are available (video, sounds, animations, interactive elements…), they require expertise in much more areas than when creating a textbook. Another issue is that the starting vision of producing courses to be completed by learners entirely on their own could not achieve the expected results. The need for coaches and tutors resulted in the concept of "blended learning", which again incurs costs weighing against widespread introduction of DE. Also resulting from the last argument the importance of groupwork in DE is emphasised.
Intelligent agents can help in all these areas through automating smaller and/or easier tasks. This would probably not result in direct cost reduction, but rather in improvements of quality, offering additional possibilities and perhaps slightly enlarging the number of learners per coach. As an agent in this context we understand software, which is autonomous, proactive and goal oriented and interacts with its environment. Often agents are also mobile in the sense that they can move during their execution from one computer to another. For the applications envisaged here this is however of relatively little concern. We focus on the aspects of autonomy (fulfilling tasks on their own without continued or detailed guidance) and interaction (separating agents from the actual learning platform, resulting in easier changes of functionality as well as enhanced security and adaptability to personal requirements). Agents are here used on the implementation side providing advanced functionality, leaving beside "avatars" and sticking with a more conventional UI.

In the next chapter the online learning platform (OLP) WeLearn is presented, which is the basis for the implementation of the functionality described in the chapter afterwards. Near the end the role of agent-oriented design (a software development method building up on object-orientation) for groupware is discussed in the context of OLPs and a short summary concludes this paper.

The WeLearn platform

WeLearn (Web Environment for Learning, [8]) is an online learning platform developed at this institute. Actually WeLearn is a framework consisting of 4 major components:

- The WeLearn learning environment or shortly called the WeLearn-system (WS), offering the basic functionality.
- The settings for schools, universities and adult continuing education, providing ready-made frameworks for different types of courses.
- Course material, which was specified and edited especially for DE. It can be used as examples as well as parts for new material.
- The WeLearn Offline Converter that converts courses in CPS [1] format to an offline (D)HTML version. Consequently course material can also be presented on CDs without any change required.

The WeLearn-system is a free and open Distance Teaching/Coaching/Learning environment (GNU-philosophy) [2]. It is universally useable, which means that using it is not limited to specific target groups, didactical models or training areas. Therefore one major design goal was to keep the system easily and intuitively useable, which e. g. would also allow computer novices using it without or with only very little training. The same is true for the teaching side, which is also intended to be intuitively useable and providing results as simple as possible.

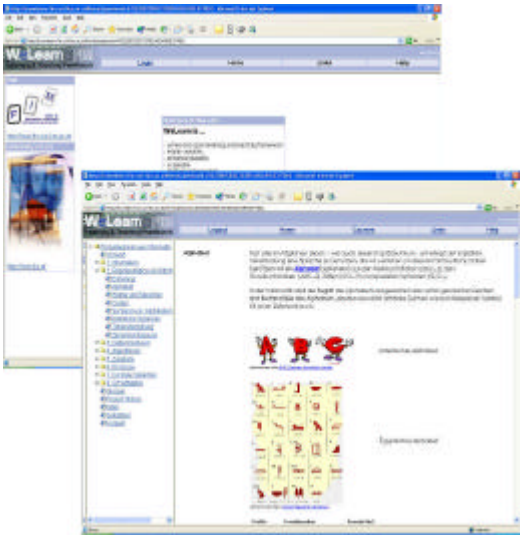

Figure 1: The WeLearn-system

Because of its implementation as a framework it provides possibilities for both adapting and scaling. The underlying construction kit philosophy allows offering not only generally suited areas but also tailored settings. So specific learner and coach preferences can be realised and it is possible to respond quickly to a changing learning situation [3].

From the technical viewpoint WS was developed with the object-oriented paradigm and a view on agent-oriented engineering at the design stage. Therefore everything within is treated as an object: users, folders, documents, forums, etc. As a result a WeLearn-system is the arrangement of selected objects within a highly dynamic system structure. This structure can be continuously modified through the addition and removal of objects, occurring as the result of activities such as setting up users, uploading documents, creating folders or adding CPS material (e. g. courses).

Furthermore, in order to guarantee extensibility and platform independence WS is fully implemented in Java. Independence and extensibility are not the only advantages of this design decision, but also the possibility for integration with e. g. agent frameworks (which are usually also written in Java). Another important fact is that newly created components can be added to existing WeLearn-systems at runtime. This means that running systems can be updated whenever needed. Reasons for that can be various: updating features, enlarging the functionality, new materials, etc. This dynamic adaptability is very similar to (especially mobile) agents and a result of partially using the paradigm of agent-oriented engineering for design. An important issue resulting from this is that system objects (=code) and objects created by users (=data) can be treated independently and so user data is not affected by updating the system itself.

Each configuration of a WeLearn-system can differ in functionality. Furthermore, each course within a WeLearn-system can differ in configuration and functionality from each other. When talking about functionality one can mainly distinguish between administration, presentation of courses, contents (e. g. online material), and support of the learning process (e. g. discussion forums).

Administration in WS consists of the configuration of the system itself, the handling of courses and course materials and administration of users or groups. Courses and content can be personalised so a learner only sees courses he/she attends. This is realised through a strong rights-system managing access to all objects like documents, folders, forums, etc. The system allows treating each object independently. This means that for each object individual rights can be defined for different users and user groups. Furthermore, any data format can be included: All kinds of documents (text, audio, video, etc.) can be uploaded and accessed through the system.

This is also an important issue for the presentation of courses. Course providers can reuse their already existing material without having to adapt it in any way, and start immediately with DE. This is especially important for newcomers to DE desiring to introduce it. Our experiences in schools show that many teachers already possess electronical teaching material. They can easily include this into WeLearn without changes (no need to adopt or transform it) and start teaching with DE-methods with further ado. For realising this, the Content Packaging Specification (CPS of IMS [1]) is supported by the WeLearn-system. This specification defines the structure of courses and couples hierarchical structure and physical resources. Consequently structured as well as unstructured content can be embedded and presented.

In the area of learning process support WS offers several tools and functionality. First of all, each user has his/her own web space. This is a private area where learners can work individually and in addition also have the possibility to share data with other users. This instrument of shared folders can also be utilised within courses to realise collaborative work amongst learners and coaches.

To enable and support communication and interaction, forums can be added to individual courses. These forums can be used as global discussion boards but also for group discussions or chat. If a coach wants to present information, information boards and information texts can be used instead.

The whole development was driven by “keeping it simple to use” (providing complexity but hiding it to the utmost degree possible), platform independence and accessibility. WeLearn has reached these goals as feedback from practical use at several institutions (University of Linz, University of Zurich, BRG Wagrain, HBLA Steyr) shows. Users can access their courses within WS from all over the world simply by using a web browser, a feature especially useful for coaches. This and appropriate viewers for the content (if not in HTML; e.g. Acrobat reader) are the only requirements for users.

However, judging from our experience and the feedback we receive from our partners and users (course providers and learner), we are convinced

that further improvement is possible. We are now working on a two-year project funded by the FWF, for realising the vision of a personalisable and adaptive platform supported by agents.

In order to achieve this, WeLearn will be redesigned and enhanced. As part of the new system a docking point for mobile agents to an already existing platform called POND [9], [10] which was also developed at the institute will be specified and implemented. This docking point will allow them to acquire information (topics, available courses, etc.) and products (course materials) for their users to present them (or derived or condense information) in a simple and straightforward way as well as take actions within the system, e. g. assisting coaches in administration. Agents should therefore serve as a kind of middleware to connect users to the new system and hide additional complexity, respectively providing new functions without increasing it. In addition, agents should support the learner within their learning process, as well as the coach within its coaching process. This will be done on a more internal level, trying to hide the complexity of the use of agents, resulting in integrated kinds of "wizards".

Supporting groupwork in DE through agents

Enhancing groupwork in DE can be done in two ways: Improving (either asynchronous or synchronous) communication between learners or enhancing the presentation (awareness, finding other learners to communicate with or to integrate individual results building a base for cooperation). Several ideas for these areas are presented along with the plans for their implementation in (or integration to) the WeLearn system.

Roadmaps and awareness

In the offline version of the WeLearn system (materials to be distributed on CDs) already a basic version of a roadmap is included. Under a roadmap we understand a graphical representation of course material with special properties: Not each small item/file is shown, but rather only larger elements as nodes of a graph. Connections between nodes represent ways of navigation between topical areas. In contrast to the obligatory tree-like structure (both navigational and from the content) of the IMS CPS we soften a bit and allow references to submanifests on different and sibling levels (which by default represent nodes). Through this a network instead of a tree is possible.

For the online version of this roadmap, agents will be introduced to provide additional functionality. They will track other users (by communicating between themselves) and show them on the map (e. g. as coloured dots), providing a kind of visual "buddy list". This helps a learner assess his own progress in relation to others or to find someone currently working on the same topic. This would otherwise need additional communication just for identifying such persons, which share (at least at the moment) a similar interest. This results in more focused communication on the topic.

Another related task for agents could be a kind of "locality chat": Only comments of those persons can be heard which are in the vicinity (working on the same or nearby topics). The special problem of transitivity (the answering person is near enough, but the person asking the question is outside your listening area) is here the task for agents: An agent must decide which parts (of a larger area) should also be visible for the communication to be sensible. This requires identifying questions and associated responses and perhaps additional filtering according to the topical content. As no real problems can arise from mistakes (worst case: ask what it was about), this is a prime area for agents.

Tracks

Related to the roadmap but working from the other way around are "tracks", a well-known metaphor for perhaps one of several common ways through a large but finely partitioned set of data, usually adhering to a general idea. This could be e.g. a track for beginners, skipping technical details and exact specifications, or tracks putting special emphasis on one area but skirting around others.

Instead of creating them manually, the idea is to let agents identify them by observing the users way through the course. This ties in with the problem of requiring extensive (and different) ways of navigating through content. Two main forms can be identified: Providing a complete (or at least very extensive) mesh and "bundling" it together to achieve tracks, or offering only a simple navigation (e.g. linear or tree) and creating the additional tracks through inserting jumps of several users as a new connection (and track). As the first is very confusing, we opt for the second approach offering at least one usable navigation from the start.

Agents are used in this area for simultaneously identifying patterns in the tracks and annotating them accordingly. If many learners go back from one place to another previously visited, linger there or in the vicinity for a short time, and then return to where they came from and go on, this looks like some kind of lookup. They missed or forgot something and moved back to refresh it. This information can be inserted automatically into the course as an additional link with a matching icon. The certainty of the agent on the correctness can also be visually displayed: Few data and therefore high uncertainty would result in small text (or inconspicuous colour) while lots of exact data result in larger and more prominent display. Moreover, a list of all these additions can be maintained, giving the author feedback on the use of the material by the learners. Another pattern is looking things up (e. g. index or glossary). Here perhaps the information should be added directly to remove indirection. Yet another pattern are "sidetracks": Paths, which are followed for some time, but then the learner returns to the original position and goes on from there (difference to lookup: longer path instead of tightly focused point).

If no previous roadmap exists, it could be built up from the individual tracks through a kind of self-organisation by learners. A drawback of this approach is that it works only slowly. Therefore only relatively stable material can be enhanced in this way. If only small changes are introduced, agents can perhaps adapt tracks accordingly. This is a possibility only because of an explicit description of the course and its structure in the form of a manifest: changes can be detected rather easily and parsing webpages or other documents for identifying connections between them is unnecessary. As this is still rather complicated, it is not planned for initial implementation.

Active training

In special areas where the learner is trained as a service person for non-specialists, agents could be used for practical experience. An example is in E-Government, where clerks are trained to deal with petitioners: Because of the usually rather limited knowledge of them on the topic and only a specific desire, agents can represent them and in this way provide live examples. If an agent does not fully understand the questions or answers presented to him (e. g. because of deficiencies in natural language parsing), this is no problem here: It might also occur in real life and patience and rewording is trained also. A related area, where instead of much freedom very structured communication takes place, is training of callcenter agents. From a set of responses the agent selects the most appropriate one (or sometimes selects another, e. g. when incorporating "moods" by callers).

In this way interactivity and some communication skills can be trained without needing a human counterpart or a coach. The latter is not needed here as the agents can explain how and why they selected their responses: The learner can then itself find out why his responses where suboptimal and what the agent had expected.

As no expertise in natural language processing is available and the intended target groups do not match either possibility, this is currently not planned for implementation.

Asynchronous support

One problem of cooperation is asynchrony: Not all learners are available all the time, introducing communication problems (e. g. the person to ask is not available, and later the asking person must be notified of the now available answer). The same applies to the coach. He cannot be available all the time and not all actions (e. g. postings in forums) can be supervised. The learners/coaches agents (being available all the time) might mitigate this. Standard tasks like arranging appointments for synchronous communication or receiving documents, noting the time and further handling can be done automatically. As these are relatively simple tasks, agents are not really needed but can be of utility because they can be easily exchanged (versioning) and communication with other entities (often involved) is their speciality.

Another type of asynchrony is, that navigation within a course can be difficult because of its network structure. Roadmaps (see above) could help. But the same applies to the whole learning platform: Taking a test, looking at material explaining a wrongly answered questions, reading a response to a previous question, and so on can also cause confusion. Agents can easily remember the path and provide direct links to important points, e. g. the last stage in each main subpart (example above: last question answered, current place within the course, forum visited last), enhancing learner's navigation.

Assembling results

A task planned for implementation is automatically integrating material from several students to a whole. As an example a seminar can serve, where different persons (or small groups) work on individual but related topics. Currently the results (usually websites) must then be integrated manually into the course website. Collecting individual results and placing them into a common place is easy, but rewriting URLs (as might be needed if students do not adhere to guidelines of using e. g. only relative links) or to replace E-Mail links by images (spam protection), is already more complicated. Also, to be integrated into the WeLearn system, a manifest should be created so also a navigational structure within the collected material can be provided. For this all the referenced items (files, images, …) must be derived from the webpages as well as a sensible structure be found. The latter can either also be taken from the webpages (if simple; e. g. already like a tree), or be done in a simple way and left for manual improvement.

To enhance this material for the WeLearn platform (and generally) metadata should be provided for them. If this is not explicitly included, agents can derive at least some from the webpages itself (e. g. identifying abstracts and keywords from the text, using the title tag as subject, look for "last changed" information for the date, …). As an example, agents for deriving Dublin Core–Metadata [5] from webpages already exist. They will be extended for identifying further information (and differently presented one) and integrating it into the WeLearn system (by being able to insert this information also into manifests). If such metadata exists for each course (regardless whether created by agents or the authors), it should then be integrated into a description of the whole course (or at least prepared for the coach to review). Here not only simple copying together but also reducing material (e. g. omitting date/time) and ordering (e. g. keywords) are tasks agents are required for.

As most of these tasks can be done automatically, continuous improvement is possible in the sense that every time new material gets available, it can be immediately included into the whole instead of having to wait for all of them to arrive when done manually. This partly reduces the work for the coach and partly changes it to check and correct the proposed results from the agent.

Simple personal coach

As an extension to the human coach, agents can provide additional guidance. Agents can here be seen as personal helpers, facilitating the learning process. They can easier remember the exact learning history of "their" learner: What chapters/parts of the course he/she had already visited and which test have been completed (together with the results on the different topic areas), and what the personal interests are (either explicitly set or derived from visits to other courses). Based on this information, recommendations for further actions can be given. Examples are chapters to repeat because of low results on tests in this area or the next part to visit. In this way an individual path through a course can be recommended. This is eased through the existence of tracks (reducing the complexity for the agents) and metadata (giving information on the topics of a part). However, this only works for routine parts. Special recommendations for newcomers (which course to take or which path to follow) are probably outside the scope of agents and should remain task of the coach, providing more reliable guidance.

Based on the same information, agents can also create individual tests, taking note of previous results and the parts viewed. This again is based on metadata information, both from courses (and parts of them) and the list of questions. As this information is available for personalised examinations, it can also be used simultaneously for giving immediate feedback on wrong answers, including direct links to the part of the course material this is explained in.

Community Building

Metadata and personal profiles can be used to find learners with similar interests for discussion, learning communities, etc. Another scenario would be to find learners facing the same or similar problems (through their queries; to be seen apart from the area they are currently looking at) or one can even find learners that have already solved such problems (who could take the role of a coach). First, using the metadata helps to find problem-related communities. Second, learners see that others also face similar problems and that they are not "lonesome riders". Last but not least coaches get the possibility to concentrate on severe problems when they do not have to cope with all problems.

Task automation

Many tasks within a DE learning environment are repetitive or long drawn-out. These tasks are often very simple to perform but time-consuming. Therefore automation through agents would be welcome. When talking about administration, agents can help arranging the platform (set up and archiving of documents and courses, setting user rights, creating the local/preferred "standard" environment for new courses, etc.).

Agents can also be used as notification systems. Every time a new message is posted in a forum, a file is added or modified, tests or test results are available, agents can automatically create notification messages or summaries of the elements involved and notify users by E-Mail, SMS, etc., also including direct links for faster access.

Partially also the correctness of exercises (multiple choice, gap-fill-in, etc.) can be checked by the agents (see also above). Agents can also here aid administration by sending the answers given directly to the coach or the tutors for detailed checking. Also, keeping track of lists (what parts are completed, questions answered correctly, etc.) is a task to be automated.

Metadata in cooperative learning

Metadata has already been mentioned as a leading actor and an enabler of cooperative learning. But what is metadata and why is it so important?

Metadata is data about data. So when using metadata, data is added to enable precise identification, retrieval and distribution of the actual content data. Data enriched with metadata evolves into structured information. The general idea of metadata is not new. Actually we already use it in everyday life. When we write papers, articles, etc. we sometimes add annotations. Annotations represent additional information directly related to the text at a certain point and can be seen as metadata (although this term is usually only applied to electronic additions).

In the area of DE several metadata specifications have been developed so far, which are referring to the additional information needed. LOM (Learning Object Metadata) developed by IEEE LTSC [4] is one of these specifications. It includes tags for administrative metadata (e.g. author, revision date, rights, etc.), technical metadata (duration, digital format, platform requirements, etc.) or subject classification metadata (e.g. catalogue system, subject heading, keywords, etc.).

Agents can use this valuable information in various ways. Content can be found easier using metadata. Furthermore, when considering the metadata, agents can search for and assemble content especially suited for the individual learner. So within one course each learner gets his/her personalised learning material adapted to his/her skills and previous knowledge. Additionally, agents can also handle the presentation of the content according to personal preferences. If a learner is more text oriented, agents can select material in text format (if available, or convert it). If a learner desires to read less or is visually impaired, agents will choose audio files and larger graphical presentations.

But it is not only the content that can be looked after by agents. Agents knowing the learning profile of their user can build teams or learning groups. Nowadays so-called "buddy lists" are used to form learning groups. Learners can subscribe the list in order to find learning partners, but most often different skills and their current knowledge are not considered holistically.

Besides working with and using metadata, agents can be enabled to create metadata as well. Documents and forums can be scanned by agents and

tagged with metadata according to their content. Agents can also track learners within the learning environment. Through this they build a model of the users interests for personalisation and provide input for other tasks (e. g. automatically creating tracks; see above).
Finally, metadata is often expressed in several languages. If the needed one is missing, agents can provide a first approximation using web-based translation services. These can be either used directly or later be approved by the author or coach, or remain as such (annotated with its unofficial status as an automatic translation; a kind of meta-metadata).

Agent-oriented design for OLP

The basic idea behind agent-oriented software engineering (AOSE) [6], [7], [11], [12] is grouping on an even higher level than object-orientation (OO). OO is for the most part implementation centred and static: a class in its function is rarely changed after the design and modifications happen through adding subclasses. AOSE on the other hand focuses on the function of "things" (in this case objects) and ignores implementation (which is then usually done using OO techniques). The main emphasis of AOSE is putting focus on the coordination of several dynamically interacting components. Each component works in a separate thread, therefore inherently incorporating synchronisation problems because of communication between them. In a learning platform, each user should possess an own path of execution (e. g. several processes or threads), as tasks will vary widely. The idea behind AOSE is seeing software design less as a decomposition of tasks (which is fine for single-thread or individual complex problems) than as designing protocols for communication. We can therefore state that agents (usually, but not necessarily) consist of objects, but are fundamentally different. While objects can control their internal state (e. g. via declaring attributes as "private"), they cannot control their behaviour: A method is just called and the object has no say in this. Agents however can additionally control their behaviour in the sense that they are asked to do something and they autonomously decide whether to do so or not. This difference is only of importance if entities might follow different (and potentially conflicting) goals. This is e. g. true for cooperation in DE. Each learner possesses his/her own topical focus, speed and way of learning. Interaction between them must therefore be based on mutual consent and always incorporate the possibility of many (perhaps even wildly) differing environments (position within the course, on-/offline status, role, …).
An advantage of using AOSE for implementing learning platforms is, that usually all goals are more or less decoupled: Cooperation improves results for one or both parties, but usually does not reduce the quality of either participant. Therefore a "fight" between learners (and between their agents) for resources does not happen, reducing the complexity of interactions and the intelligence needed. Also, the environment of a learning platform is much too complex for an individual piece of software to completely monitor or even understand (in the sense of an internal description of it; a world model). Agents always have to cope with a limited view on their surroundings (and sometimes not only incomplete but even wrong information) and are therefore better suited for implementing advanced tasks of such a platform like the ones described above.
Deducing tracks from usage by learners is a good example: personal agents record the individual navigational paths (doing so on the server requires lots of additional work, introduces delays, and data must be sorted together afterwards for use). Afterwards, a separate agent retrieves these paths from all the agents (communication) and compares them, identifying and classifying new tracks. These must then be integrated into the material (inserting new links, changing the navigational structure), while notifying agents of the changes. The latter is necessary for those agents whose owners are currently active, as their navigation must be redrawn to incorporate the changes. This central agent should also be implemented as an agent as otherwise changes (e. g. new types of paths or different strategies) are difficult to incorporate and would require stopping the whole system.
The last aspect is another advantage of using AOSE for implementing advanced parts: OLP are almost always asynchronous and provide only some synchronous tools. This means it should never be taken offline to be available round the clock. Updating software is therefore a problem, as the whole system must be stopped in case of monolithic design. Exchanging dynamic libraries is also not possible as they might be currently in use. Agents do not possess this limitation. They can be rather easily exchanged. If this is supported, new agents can just copy the state of the old version, restart the last activity, and remove the original. If restarting is impossible or the agent is currently in use by its owner, this is no problem for agents: The new agent just waits for a better time and takes over later. Each agent is a unique entity and therefore the same (code) agent can exist for the same owner (state) twice without problems. Also, extending the functionality can be done very easily: Just a new agent needs to be introduced and configured (and connected to existing ones, if this is not done mostly automatically as in our systems).
An optional quality of agents is mobility. In connection with an OLP (where learners are by definition distributed) this can be an additional advantage. While most agents will be located at or near (on a dedicated host) the server, some agents might be configured by their owners locally and then transferred to an agent server for their work. In the case of remote agents they can in this case also be easily updated, by creating them on the server and afterwards sending them to the user upon his next login. This is however not a necessity.

Conclusions

In this paper we presented reasons for and some ideas for tasks for integrating agents into online learning platforms. The advantages are a reduction of simple work and enhancing the experience for the learner by providing personalised services and addition support in the area of awareness and navigation within course materials and the Distance Education platforms itself.

For agents to provide suitable results without requiring too much time and effort for parsing the actual content, metadata is needed. Some parts can be derived automatically by agents (e. g. simple document information based on formats within the content, or automatic translations), although the most useful ones are probably those manually created by the author.

The agent paradigm can also be useful in the area of design, as online learning platforms are inherently multithreaded and consist of many separate entities communicating with each other. Design methods focusing on the interaction between elements (like AOSE) are therefore better suited for a high-level design.

We are confident, that agents can improve DE, leading to a more widespread acceptance, if their complexity is hidden from the user and they fulfil their tasks in the background, than offer rich functionality, but at the cost of increased initial training and the need for extensive configuration.

Acknowledgement

This paper is a result of the project “Integrating Agents into Teleteaching-Webportals“ sponsored by the FWF of Austria (Fund for the support of scientific research; Project number P15947-N04).

Literature

1. IMS Content Packaging Specification, Version 1.1.2 Final Specification:
http://www.imsproject.org

2. Divotkey, R., Mühlbacher, J.R., -, Remplbauer, D.: The WeLearn Distance Teaching Framework. Granada: EDEN (European Distance Education Network), 2002 Eden Annual Conference, Open and Distance Learning in Europe and Beyond Rethinking International Co-operation, Conference Proceedings, 2002

3. Mühlbacher, J.R., Mühlbacher, S.C, -: Learning Arrangements and Settings for Distance Teaching/Coaching/Learning: Best practice report. In: Hofer Christian, Chroust Gerhard (Ed.): IDIMT-2002. 10th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner 2002

4. IEEE P1484.12 Learning Object Metadata (LOM) Standard, http://ltsc.ieee.org/wg12/

5. Dublin Core Metadata Initiative:
http://www.dublincore.org/

6. Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In: Ciancarini, P., Wooldridge, M. (Eds.): Agent-Oriented Software Engineering. Berlin: Springer 2001. LNAI Vol. 1957

7. Jennings, N.: An agent-based approach for building complex software systems. Communications of the ACM April 2001. New York: acm Press 2001. 35-41 (Volume 44, Number 4)

8. WeLearn: Web Environment for Learning: http://www.fim.uni-linz.ac.at/research/welearn/

9. POND:
http://www.fim.uni-linz.ac.at/research/Agenten/

10. -: POND - Ein Agentensystem mit Fokus auf Sicherheit und Bezahlung für Leistungen unter Berücksichtigung rechtlicher Aspekte (POND - An agentsystem with focus on security and payment for services with consideration of legal aspects). Dissertation. Johannes Kepler University Linz: 2002

11. Jennings, N. R.: On Agent-Based Software Engineering. Artificial Intelligence, 117(2), 277-296, 2000

12. Wooldridge, M.: Agent-based Software Engineering. IEEE Proceedings on Software Engineering, 144(1), 1997