

Ontology-based Support for Designing Inquiry Learning Scenarios

Stefan WEINBRENNER^{a,1}, Jan ENGLER^a, Lars BOLLEN^b and Ulrich HOPPE^a

^a*COLLIDE Research Group, University of Duisburg-Essen
47047 Duisburg, Germany*

^b*University of Twente, PO Box 217,
7500 AE Enschede, The Netherlands*

Abstract. Ontology support has been incorporated into a design environment for inquiry learning scenarios in the context of an ongoing European project. The underlying ontology is being elaborated as an interdisciplinary agreement on basic concepts, their relationships and attributes between designers and developers. The ontology encapsulates concepts such as “pedagogical plan”, “scenario”, “learning activity”, or “tool” together with compatibility relations. This forms the semantic basis of a cooperative graphical editing environment. Similarity measures addressing structural aspects (graph matching) as well as semantic similarities allow for generating social recommendations in the SCY² design community.

Keywords. Learning Design, Pedagogical Scenarios, Ontologies.

1. Introduction

Although often disregarded, there is a common agreement in the TEL community that ontologies can enhance and enrich learning environments, e.g. in the form of competence ontologies used for learner profiling [1] or for building a framework for instructional and learning theories [2]. Our approach uses semantic representations (in the form of an ontology) as an internal resource in a specific learning environment that is currently being developed in the new European research project SCY. In this sense, our approach differs from a general “semantic web” approach (cf. [3]), which would extend to a larger and open collection of web resources. From an educational perspective, an obvious added value of using an ontology is to define a set of terms and their relations and then use these as a shared vocabulary. This shared vocabulary can then serve as a common verbal denominator for a development and design group to synchronise and co-align their wording and conceptual perspectives. By doing that an ontology can help to avoid misunderstandings and to improve the precision of communication in general. In addition to the provision of a shared vocabulary, an

¹ Corresponding author.

² SCY – “Science created by You” is an EU project of the 7th Framework Programme. For more information, see <http://www.scy-net.eu> (last visited in April 2009).

ontology may help inferring semantic relations between objects and thus may indicate recommendations or possible flaws in a certain domain.

In this paper we elaborate on the communication and conceptualisation perspective by introducing specific system support through providing ontology-aware tools for educational designers. In this sense, our target situation is very similar to the one of the SMARTIES scenario editing environment based on the pedagogical ontology OMNIBUS [4]. However, in our approach there is no given equivalent of OMNIBUS as a general basis of pedagogical concepts and patterns, which are used as generic or foundational concepts to underpin specific designs. Instead, the SCY ontology with different sections addressing domain knowledge, pedagogical and task concepts is being built up according to the needs and specific choices (e.g., example domains) of the project, specifically oriented towards inquiry style learning. Also, we do not want to encompass a large variety of pedagogical approaches to be modelled in detail but concentrate on SCY specific methods of teaching and learning. Rather than defining these in very flexible, generic form (as in OMNIBUS), we aim at practically oriented blueprints without capturing “pedagogical causality”.

In general, ontologies can capture semantics (concepts) from domains of interest as well as methodological and task knowledge and support reflection and self-description in the context of the system environment. As for our SCY project, we see the following specific roles of ontologies:

- Represent terminological/semantic basis for human-human interdisciplinary exchange;
- Serve as meta-level description for storage structures (repositories);
- Standardise and interlink metadata vocabulary (e.g., for the learning objects created by the students, domain and student models);
- Provide basic concepts for the definition of agent behaviour on a semantic level;
- Facilitate high level inter-operability between tools and services.

It has turned out turns that building up an ontology for the SCY project both enforced and established an agreement on the basic terminology in terms of concepts, their relationships, and attributes between designers and developers from different academic disciplines. The ontology encapsulates the agreements on basic terms such as “mission”, “pedagogical plan”, “scenario”, “learning activity”, “tool”, or “scaffold” together with lexical enumerations of certain elements (activities, tools, types of learning objects) and their compatibility relations. This forms the semantic basis of the editing environment. Similarity measures addressing structural aspects (graph matching) as well as semantic similarities allow for generating social recommendations in the SCY design community.

2. Educational Design for CSCL Scenarios

Context and requirements for our developments originate from the SCY project, which aims at delivering a flexible, open-ended learning environment for science education and scientific discovery learning. SCY integrates a number of different tools, background material, collaborative activities, etc. to form “missions”, which are then accomplished by groups of learners in an environment called “SCY-Lab”. SCY-Lab provides adaptive support and pedagogical scaffolds to help learners on their missions. One central aspect in SCY are “emerging learning objects” (ELOs) [5], which denote

re-usable, sharable products of learning activities, created by students themselves, rather than traditional learning objects, which are typically used in the sense of pre-fabricated learning material.

At the very beginning of SCY, it became obvious that the expected, complex interplay of tools, activities, scaffolds and ELOs needs to have an external representation for discussion, authoring, exchange, and potentially to be machine-readable and machine-interpretable. A number of languages and tools for the design and storage of learning processes are available, like IMS-LD³, LAMS [6] and COML [7], each with particular strengths and weaknesses. For example, IMS-LD strongly builds on XML representations, which are hardly comprehensible for non-experts. LAMS appears as an combined authoring-player suite that makes it hard to integrate third party tools and COML focuses on the use and integration of hardware like mobile devices or shared displays. However, one strong deficit of all these approaches with respect to the SCY requirements is the lack of support for emerging learning objects.

These shortcomings called for the development of a comprehensible, flexible and pedagogical neutral learning design language. As a result, the concept and graphical representation of so-called Learning Activity Spaces (“LAS”) emerged, which is described in detail in [8]. A LAS is defined as a coherent and intuitive set of activities supported with specific tools and scaffolds. The input and output of a LAS are described in terms of a set of artefacts usually created by students (i.e. the ELOs).

Based on these premises, a graph-based modelling tool has been developed to support the creation and exchange of LASs. This tool, called “SCY Scenario Editor” (or “SCY-SE”), builds on the graph-based modelling tool FreeStyler and enhances it through the use of ontologies. To be flexible and to provide a high degree of representational freedom, but still be able to interpret and intelligently support a designer, the SCY-SE tool relies on an ontology in the backend. The ontology contains a representation of available tools, activities, types of ELOs, scaffolds and the relations between those entities and is able to provide information to identify possible LAS elements, to give recommendations and to discover discrepancies in the LAS design.

In the SCY project, the same ontology is planned to be used for other purposes, too, like query extension when searching for ELOs, identifying applicable tools for an ongoing learning activity or for deciding on an appropriate scaffold for the current situation. However, this paper deals with the usage of ontologies in a graph-based learning design editor. The following chapters will elaborate on technical requirements and use cases of ontology-based models for pedagogical scenarios.

3. Implementation Platforms

3.1. Blackboard Architecture based on Tuple Spaces

Our approach uses a blackboard architecture for the backend, both as a container for the evolving ontologies as well as a communication medium for additional agents. Such a blackboard system comprises a central server with several clients that interact with each other only indirectly by exchanging information through the blackboard [9]. Following a distributed problem-solving paradigm, the blackboard approach is very

³ For the full specification and more information in IMS-LD, see <http://www.imslobal.org/learningdesign> (last visited in April 2000).

appropriate in situations with multiple clients or agents that have quite different functionalities and differing implementation styles. In these situations, in which a problem solving strategy is divided into several steps with each step covered by one agent, there is often an implicit order of processing a query, but without having tightly connected components, i.e. without any explicit, external ordering. This corresponds to a “loose coupling” approach, which essentially means that all participating components were designed using minimal assumptions on the knowledge and awareness of each other. These minimal assumptions lead to a high robustness and failure tolerance and make it easy for designers to flexibly adapt the architecture.

TupleSpaces, as introduced by [10] together with the Linda language for distributed processing, are a well known means to implement a blackboard architecture. Here, a central server acts as a relay station for exchanging messages. These messages are stored in the form of tuples, i.e. in ordered sequences of typed literals. The most prominent recent implementations are JavaSpaces (Sun Microsystems) and TSpaces by IBM [11]. Our ontology backend, however, is built on top of our own implementation called SQLSpaces [9]. Since the SQLSpaces were developed on top of a relational database, they have an inherent support for persistent and efficient data storage and provide transactions. Additionally to typical convenience features of other TupleSpaces implementations like asynchronous notifications (callbacks), automatic expiration of tuples, blocking and non-blocking commands, bounded queries, a graphical web-interface and a user management, the SQLSpaces have other, new features like multi-language support (currently supported languages are Java, C#, Ruby, Prolog and PHP), wildcard fields, space-based rights management and versioning.

3.2. SWAT

The SQLSpaces are the technical foundation for the actual ontology management facility called SWAT (Semantic Web Application Toolkit). It has been developed by the Collide Research Group in the Ontoverse project [12] that aimed at developing a web-based platform for collaborative ontology creation in the Life Sciences. SWAT is a framework that uses the SQLSpaces as a platform for a multi-agent system that encapsulates all the functionality in several agents coordinated via the SQLSpaces platform. Therefore, it uses several spaces inside the SQLSpaces server as disjoint communication channels.

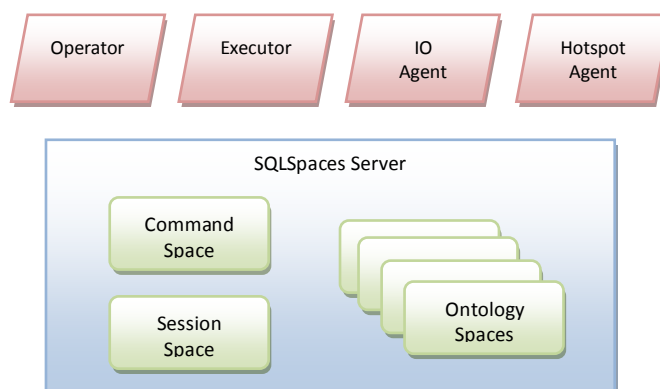


Figure 1. SWAT components

A simplified overview of the SWAT architecture is shown in 1. At the top of the picture some of the agents of SWAT are shown, namely the operator (for supervising other agents), the executor (for starting and restarting other agents), the IO agent (for importing and exporting OWL files) and the Hotspot agent (for calculating modification statistics during the evolution of an ontology). As described in section 5, this paper will explain the design and implementation of another functional agent that is capable of calculating similarity of certain entities on the base of an ontology. At the bottom of Figure 1, the SQLSpaces server is shown that holds all the spaces for SWAT. The command space is the central coordination space for all agents. The session space acts as a kind of logging and coordination space for a user. Finally, for each ontology there is one space in which the whole ontology is stored in the form of RDF triples.

3.3. FreeStyler

FreeStyler [13] is a collaborative modelling environment based on the idea of shared workspaces offering freestyle-sketching as well as different visual languages. FreeStyler can be used in collaborative as well as in single-user scenarios. The support for different representations is very flexible, as it based on a plugin-framework for graph-based visual languages that can be implemented as plugins through a common interface or API. This API was also used to implement the user frontend of the scenario-editor. Figure 2 shows an example scenario design, which consists of several ELOs (green diamonds) and LASs with substructures (blue boxes) thus illustrating the two levels of representation (macro level: scenarios / micro level: LASs).

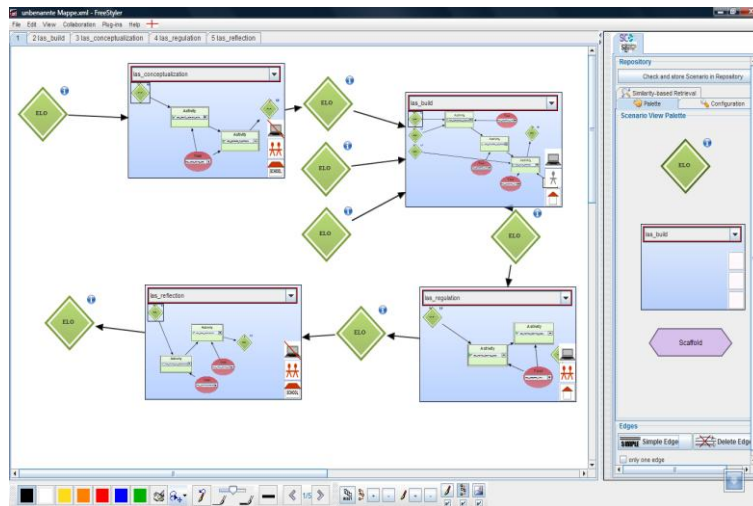


Figure 2. SCY-SE as FreeStyler plugin.

4. Usage Scenarios

The ontology serves as a basis to implement four groups of user support features that all serve different purposes and that all affect different phases of the creation process.

In addition, the way of using the ontology differs from group to group. This will be explained in the following.

4.1. Possible Values

As described in section 2, the graphical modelling language uses the notion of a LAS, which forms the low level structure consisting of atomic building blocks like tools and activities. Furthermore, there are concepts like LAS types, activity types, concrete tools and ELO types. For each of these concepts there is a limited number of possible values to be chosen from a designer. However, it would be quite inflexible to have these lists of possible values hard coded to the application. Therefore, these lists are fetched from the ontology, where they are represented as instances of the corresponding concepts. Using the ontology as a knowledge store for all possible values of certain assignments makes it easy to extend the lists without changing the actual application code and it gives the possibility of changing the values at runtime.

4.2. Recommendations

Another way of using the ontology is to make specific recommendations of values according to the context. In the SCY project it has been clearly defined what activities are useful and adequate in which kinds of LASs. However, these relations between LASs and activities are not mandatory, so it is also possible to create LASs comprising not recommended activities. The recommendations are shown in SCY-SE as green “checks” (meaning “everything is fine”) or yellow warning symbols (meaning “this is not recommended, though might be ok”) in the activities drop-down box and are of course context-sensitive depending on the LAS this activity is located in. Similar recommendations can relate to the formats a tool supports (not yet implemented), so that the application itself would warn a user who tries to consume or produce an ELO with a tool that is not capable of processing this type of data.

4.3. Warnings

SCY-SE can check for several types of constraints and constraint violations in given designs, some of which are ontology based. The simpler, obvious ones apply to the way nodes are connected. Considering the notion of input and output ELOs, it is obvious that these nodes have a natural direction that edges to them should follow. Therefore, an output ELO needs some activity that produces it and an input ELO needs some activity that consumes it. In addition, it does not make any sense for an activity to produce an input ELO, because this ELO is the output ELO of some other LAS, where it has already been produced etc. Moreover, ELOs can have more or less precisely defined types. These types also have a relation to the tools (that support activities), because the tools need to be compatible with this type. Constraints like the directions of connections between ELOs and activities are quite simple and do not need any background domain knowledge in an ontology, whereas the compatibility of formats and tools are obviously well stored in ontological structures.

4.4. Similarity

It is of course possible to save learning designs that were created with SCY-SE as XML files. However, it might also be an advantage to have a central server, where all users can store and exchange their results. Having such a central repository has mainly two purposes: On the one hand, it might support the designer in finding similar scenarios. Access to similar scenarios may help to improve the currently developed design by comparison and reuse. On the other hand, the list of similar scenarios can be seen as a mediating object between several designers. This retrieval mechanism uses some basic mathematical principles such as distance measure and graph matching techniques as well as relations from the ontology (for determining what is more related and similar) in the calculation of similarity.

5. Architecture and Implementation

SCY-SE has been implemented as a plugin for FreeStyler and provides the means to create pedagogical scenarios based on a specific graphical language developed within the SCY project (see section 2). To implement the functionalities described in the section 4 this plugin utilises the SWAT framework (see section 3.2). The ensuing overall architecture is outlined in Figure 3.

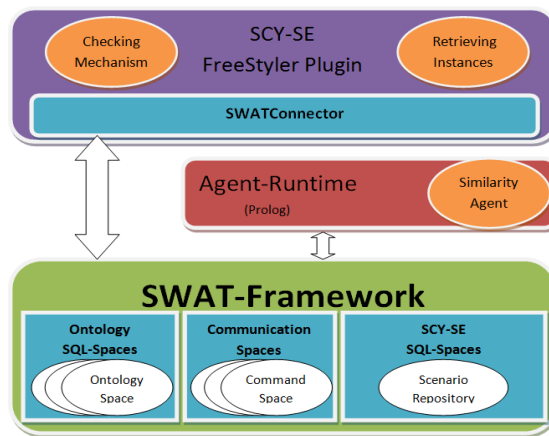


Figure 3. A simplified overview of the SCY-SE architecture.

The FreeStyler plugin for SCY-SE on the top is connected to the SWAT framework through a facility called SWAT Connector. This component encapsulates the communication to the underlying backend and allows easy extension for later work. In the lower part of the figure the SWAT framework with all its spaces is shown. Besides the SWAT-typical spaces already mentioned, there is an additional space for SCY-SE. As the name implies the scenario repository space represents a global storage for all scenario designers to save and share their results. In the center of the figure, the agent runtime component is shown. We used SWI Prolog⁴ to implement this

⁴ For more information, see <http://www.swi-prolog.org>

component whose main purpose is currently to provide the similarity calculation. Following the strategy of loose coupling, the ensuing messages between this component and SCY-SE are all passed via the command space of SWAT.

The exact use cases within the scenario design process, where the ontology access is being used, according to the general ideas described in section 4, are the following:

- *Startup*: fetch instances of LASs, activities, tools, ELO types
- *LAS editing*: compatibility between LAS and activity
- *Save*: checking based on possible relations between the ontology entities
- *Retrieval*: similarity-based search for other scenarios

Technically speaking the first point is implemented as a query to the *instance-of* relation of the corresponding ontology concept, whereas the second and the third point use certain object properties of the concepts “LAS” and “Tool”. Since the fourth point relies on the functionality of the Prolog agent, which is in this use case the component that accesses the ontology store, the communication scheme and the similarity calculation algorithm is described first.

The overall process of calculating the similarity is shown in Figure 4. In the first step SCY-SE constructs lists of the relevant entities of the scenarios to be compared. These relevant entities consist of the three list types ELOs, tools and activities, which are grouped by the corresponding LASs of the two scenarios. These three lists again can be divided into two categories: Activities and tools are atomic values, so that lists of them can be compared directly through the “weighted symmetric distance” of two sets. This distance is based on the cardinality of the symmetric difference, which is then divided by the cardinality of the union of the two sets A and B:

$$SD_{A,B} = \frac{\#((A \cup B) - (A \cap B))}{\#(A \cup B)}$$

However, in our case this distance calculation is not a plain binary comparison, but it has been improved by weights based on the ontological proximity between the elements. The resulting similarity value ranges between 0 and 1.

ELOs, on the other hand, are not suited for a plain comparison as mentioned above. In fact, they are built up from four properties and therefore have an intrinsic complexity that requires a different approach for calculating the similarity. At first glance, it is possible to calculate the similarity of two ELOs A and B by comparing the four properties, so if two ELOs share three of their four properties, they would have a similarity of 0.75. Nevertheless, the comparison of A and C might result in an even higher similarity. So, each ELO of the first list needs to be compared with each of the second one. The outcome of this step is a matrix of similarity values. To condense this matrix to one single similarity value that expresses the similarity of these two lists, we decided to use the so-called “Hungarian method” for graph matching [14] to solve this assignment problem (of which ELO is identified with which other ELO).

As shown in Figure 4, the calculation inside the Prolog agent is done in several steps. After the lists have been written to the SQLSpaces server, a “start comparing” tuple is written to inform the agent about a new query. Then the agent fetches the lists and calculates the weighted symmetric distance for each of the lists. At that point the matrix for the ELOs is already calculated by using the “Hungarian method”. The next step uses the merged matrix of these three matrices. The merging process is simply done by adding and normalising the three single matrices. The resulting matrix represents the similarity of the LASs of the two scenarios. The Hungarian method is

then used again to determine the best mapping between the LASs of the two scenario with respect to the similarity encoded in the matrix. The Hungarian method was chosen here since it is quite standardised can guarantee the optimality of the result. The method may maytake several iterations, but will eventually terminate and return the optimal solution. From the resulting mapping, a single value can be easily calculated by normalising the selected matrix entries. This value is then returned to the already waiting SCY-SE instance that presents the similarity in a user-oriented way.

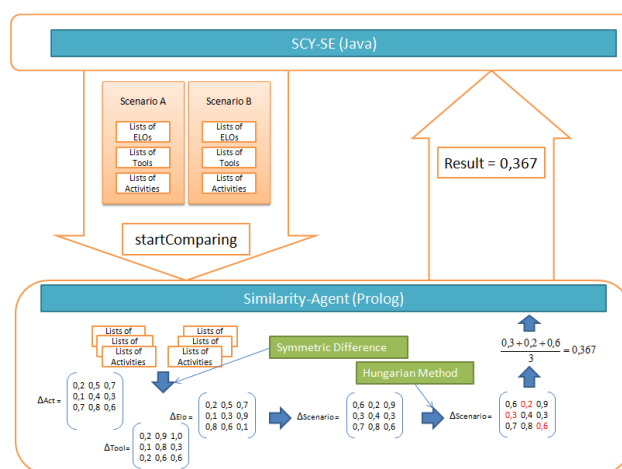


Figure 4. The sequence of actions in the similarity agent.

Using ontologies in pedagogical learning design in the use cases mentioned above, allows us to increase flexibility and correctness of the learning scenario. At the same time, this approach provides a higher precision according to the similarity measurement.

6. Next Steps

In the work presented so far the modelling tool FreeStyler has been extended with a plugin to describe abstract learning scenarios in the context of the SCY project. The possibility to describe these scenarios leads to a next step: Instantiating those abstract “pedagogical plans” towards concrete task-related “missions”. The designed scenarios can be a basis for these missions, which can be enriched by domain specific content. By using the ontology as a common repository for the elements of the scenarios and by the ontology-based checking mechanism, it is guaranteed that the designed scenarios are consistent with the core concepts of the SCY project.

The similarity measurement is currently based on the comparison of activities, tools and ELOs as described above. To improve this functionality, it will be useful to include resources and scaffolds, which are also elements of a scenario (already available in the SCY-SE FreeStyler palette).

Currently, according to the definition of the scenario modelling language, ELOs can only appear as input or output objects to LASs. In addition to this we have considered the provision of “intermediate ELOs”. These special ELOs would be used to connect activities inside a LAS. A potential implementation could make use of the

possibility to create custom edges in FreeStyler. These edges could then be enriched with operational semantics, so that the intermediate ELOs could serve as anchor points for further features. An interesting support for the designers of scenarios may be a repository of predefined LASs. These “standard LASs” would be LASs that contain a typical set of activities and tools, so that designers are able to start designing a scenario from these generic LASs.

In the present implementation, it is not possible to create custom LAS types or other elements and save them to the ontology. Yet, the plugin only reads from the ontology, but one can think of an “ontology modification mode” through which the set of elements stored in the ontology can be extended. To support this the SWAT framework already provides some easy-to-use functionalities to manipulate to underlying ontology.

References

- [1] Kalz, M., J.v. Bruggen, E. Rusman, B. Giesbers, and R. Koper, Positioning of Learners in Learning Networks with Content-Analysis, Metadata and Ontologies. *Interactive Learning Environments*, 15.2007.
- [2] Hayashi, Y., J. Bourdeau, and R. Mizoguchi. Ontological Support for a Theory-Eclectic Approach to Instructional and Learning Design. in *Proceedings of the European Conference on Technology Enhanced Learning (EC-TEL 2006)*. 2006. Crete, Greece.
- [3] Devedžić, V., *Semantic Web and Education (Integrated Series in Information Systems)*. 2006: Springer-Verlag New York, Inc.
- [4] Hayashi, Y., J. Bourdeau, and R. Mizoguchi. Toward Establishing an Ontological Structure for the Accumulation of Learning/Instructional Design Knowledge. in *Proc. of 6th International Workshop on Ontologies and Semantic Web for E-Learning (SWEL '08)*. 2008. Montreal, Canada.
- [5] Hoppe, H.U., N. Pinkwart, M. Oelinger, S. Zeini, F. Verdejo, B. Barros, and J.L. Mayorga. Building Bridges within Learning Communities through Ontologies and Thematic Objects. in *Proc. of the International Conference on Computer Supported Collaborative Learning (CSCL 2005)*. 2005. Taipei, Taiwan.
- [6] Dalziel, J.R. Implementing Learning Design: The Learning Activity Management System (LAMS). in *Interact, Integrate, Impact: Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education*. 2003. Adelaide, Australia.
- [7] Niramitranon, J., M. Sharples, and C. Greenhalgh. COML (Classroom Orchestration Modelling Language) and Scenarios Designer: Toolsets to Facilitate Collaborative Learning in a One-to-One Technology Classroom. in *Proceedings of the International Conference on Computer in Education (ICCE2007)*. 2006. Hiroshima, Japan.
- [8] Lejeune, A., M. Ney, A. Weinberger, M. Pedaste, L. Bollen, T. Hovardas, H.U. Hoppe, and T.d. Jong. Learning Activity Spaces: Towards flexibility in learning design? in *Proc. of the 9th IEEE International Conference on Advanced Learning Technologies (ICALT 2009)*. 2009. Riga, Latvia.
- [9] Weinbrenner, S., A. Giemza, and H.U. Hoppe. Engineering heterogenous distributed learning environments using TupleSpaces as an architectural platform. in *Proc. of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*. 2007. Los Alamitos (USA): IEEE Computer Society.
- [10] Gelernter, D., Generative Communication in Linda. *Acm Transactions on Programming Languages and Systems*, 7(1).1985. p. 80-112.
- [11] Wyckoff, P., S.W. McLaughry, T.J. Lehman, and D.A. Ford, T spaces. *IBM Syst. J.*, 37(3).1998. p. 454-474.
- [12] Malzahn, N., S. Weinbrenner, P. Hüsken, J. Ziegler, and H.U. Hoppe. Collaborative Ontology Development - Distributed Architecture and Visualization. in *Proceedings of the German E-Science Conference*. 2007. Baden-Baden, Germany: Max Planck Digital Library.
- [13] Hoppe, U. and K. Gassner. Integrating collaborative concept mapping tools with group memory and retrieval functions. in *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL2002)*. 2002. Hillsdale (USA): Lawrence Erlbaum.
- [14] Kuhn, H.W., The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2.1955. p. 83-97.