Tools for Collaborative Development of Visual Models and Languages

Vassiliy Tchoumatchenko, Tania Vasileva
Department of Electronics
Technical University of Sofia, TU-Sofia
Sofia, Bulgaria
{vpt,tkv}@tu-sofia.bg

Christoph Richter, Heidrun Allert
University of Applied Sciences Upper Austria
Hagenberg, Austria
{christoph.richter, heidrun.allert}@fh-hagenberg.at

Abstract—**The paper describes the design and implementation of a set of visual modeling editors. They aim to provide users with easy to use and customizable but yet semantically powerful tools for collaborative modeling in diverse domains of interest. The tools allow the creation, use and evolution of visual models and their underlying languages. The design, software architecture, technical implementation and graphic user interface of visual modeling language tools are discussed. The results of current field trials with the tools are also briefly outlined.**

Keywords-collaborative semantic modeling; ontology; visual models

I. INTRODUCTION

Collaborative design is a core element of engineering activity. While in some cases artifacts are designed by an individual engineer [1], design is most often a collective endeavor in which a group of individuals with different backgrounds and interests has to envision, explore, discuss, and evaluate proposals and claims towards the artifact at stake [2][3]. In addition to basic design skills, such as the ability to transfer theoretical knowledge to a particular design problem, to tolerate ambiguity, maintain the big picture, handle uncertainty, and making informed design decisions, collaborative design also requires participants to coordinate and orchestrate the joint effort and to communicate and integrate multiple perspectives effectively [4][3].

To prepare students for the practice of design and engineering in the 21st century, collaborative design exercises and projects became an important element of many study programs. While these efforts appear to have quite positive effects on both learning outcomes and motivation (cp. [4] for an overview of recent findings), research also indicates that collaborative design might suffer, among others, from failures to create a common understanding of and to reflect on the design problem at hand. Typical problems towards that end include an insufficient or inappropriate analysis of the design problem [5][6], design fixation and preoccupation with a particular solution [8][7], as well as tendencies to reduce the complexity of the problem in order to arrive at a solution as soon as possible [7]. In addition it has been argued that from an educational point of view the aim of collaborative design projects is not just to learn how to solve practical problems but also to understand and be able to articulate why a particular solution is supposed to work [10].

To address these challenges various authors have advocated the use of collaborative modeling techniques as a means to foster shared meaning making and reflection in collaborative design, e.g. [1][11][10]. While in recent years various tools and modeling languages have been developed to support collaborative work and learning in general, as well as collaborative design in particular, these tools either make little or no use of recent advances in semantic web technology or they are overly complex to use and hardly in line with the pragmatic requirements of students and professionals alike (cp. [12]).

To answer these challenges the paper describes the design and implementation of a set of visual modeling editors. They aim to provide users with easy to use and customizable but yet semantically powerful tools for collaborative modeling in diverse domains of interest. The described tools are part of the web-based collaborative working and learning Knowledge Practices Environment (KPE) currently under development in the Knowledge-Practice Laboratory project (KP-Lab). The tool set consists of two core components: the Visual Model Editor (VME) and the Visual Modeling Language Editor (VMLE). The Visual Model Editor allows users to create and evolve visual models in the form of two-dimensional graph diagrams such as flow-charts, argument-graphs, organigrams, decision trees, program logic models, conceptual maps, etc. both individually as well as collaboratively. The Visual Modeling Language Editor in addition allows users to create, share and edit the visual modeling languages as such, thereby providing users with the possibility to create their own domain specific ontologies.

This paper focuses on modeling as an inherent epistemic activity relevant to various scenarios in design and engineering education. Based on an outline of the role of modeling for collaborative design and a short review of state-of-the-art tools, we describe high-level requirements for computer-supported collaborative modeling. Against this background a visual modeling languages design, software architecture, tools implementation and graphic user interface are discussed. Finally, we provide a short overview of an initial field trial with the prototype as well as future plans.

The reported work is developed within the Knowledge-Practices Laboratory (KP-Lab) project funded by the EU 6th R&D Framework program.

II. VISUAL MODELS AND COLLABORATIVE DESIGN

Visual Models as means to depict and work with complex knowledge structures have been employed in various domains, including design and engineering, for many years. In its broadest sense visual models are any form of diagram in which concepts or statements are represented as nodes and relations between these concepts are represented as arcs or links. The syntactic and semantic constraints of a visual model are defined by the underlying language. Examples of visual modeling languages proposed for use in design and engineering include concept maps [13], causal maps [1], petri nets, but also argumentation based notations such as IBIS [14] or QOC [15].

Despite the adoption of collaborative modeling techniques and tools in various professional and educational settings, current tools often do not exploit recent advances in semantic web technology nor do they sufficiently address the pragmatic requirements that arise when modeling is not seen as an end in itself but as a means to support collaborative design.

A. *Modelling as an Integrated Activity*

While in some cases collaborative modeling exercises are aimed to foster the development of particular skills, such as systems thinking [16], collaborative modeling is often embedded in a more encompassing assignment such as a design project. For example, students are asked to develop conceptual models of an actual problem domain [11], to collaboratively articulate and negotiate different stakeholders' perspectives [1], or to explicate and justify the design decisions they made [10]. In all these cases modeling is not an end in itself but a means to support the collaborative design process.

Visual modeling thereby fulfills a variety of often highly intertwined purposes. Even though models are often seen as tool of representation aimed to depict known or assumed properties of a given target system, they can also be understood as "investigative instruments" in that they require an ongoing conversation of participants with the topic at stake, in order to gain better understanding of what is not known yet [17]. As communicative devices visual models can be used to create shared understanding but also to discuss divergent perspectives [1]. Additionally, visual models allow integrating ideas and reference resources throughout the design process, hence providing an evolving record of the design process [11]. Finally, by forcing participants to explicate and document their ideas, visual models provide insightful means for monitoring and assessment [11].

The added value of visual modeling usually arises from a combination of these purposes. While existing tools such as CMap [1], Compendium [2], FreeStyler [3], or the Visual Understanding Environment[4], support the representational and communicative functions quite well, they provide stand alone applications providing hardly any support for other collaborative design activities. Hence, they force users to switch and exchange resources across applications undermining the potential value of visual models as an integrative record. Furthermore, existing tools neither provide sufficient support to track the evolution of the visual models nor to discuss and comment on the elements within the model, making it difficult to orchestrate or monitor the modeling process.

[1] http://cmap.ihmc.us/
[2] http://compendium.open.ac.uk
[3] http://www.collide.info
[4] http://vue.tufts.edu/

B. *Utilization of Visual Modeling Languages*

In educational contexts visual modeling languages, such as those mentioned above, are primarily seen as a scaffold to guide the modeling process. While it has been acknowledged that visual modeling languages provide quite powerful tools in that they entail ontological commitments and make some aspects of a domain more salient than others [18][19], the users, including teachers, are usually not in control of the semantics (i.e. the modeling languages) themselves but can select from a set of predefined languages at best. The specification of modeling languages is usually seen as an expert task and priority is given to well-defined "standard" languages, which can be used in a variety of contexts. Consequently tools for the specification of modeling languages usually require substantial background knowledge and enforce explicit and complete specifications, often beyond users' interest [12] and are not attuned to the particular task or problem at hand. Alternatively, tools such as CMap provide only minimal semantics leaving it up to the user to informally agree on the semantics implied by color-coding and selection of various shapes within the models.

Based on the assumption that collaborative modeling requires users to have a common understanding of the concepts and categories used but also acknowledging that ontological commitments are at least partly situated in nature and hence not universally valid [20], both of the current solutions appear quite insufficient. Instead it appears desirable that users can adapt visual modeling languages to their particular needs. Furthermore, limitations of existing modeling languages often only become apparent when being engaged in a concrete modeling exercise. Therefore, it would be preferable to be able to extend or modify the modeling languages on the fly.

III. PEDAGOGICAL SCENARIOS AND DESIGN OBJECTIVES

To address the challenges outlined above we are currently designing and implementing a set of visual modeling tools which provide users with easy to use and customizable but yet semantically powerful means for collaborative modeling in diverse domains of interest, including design and engineering education.

Pedagogical scenarios to be supported by these tools can take quite different forms. Nevertheless, strong emphasis is put on scenarios that necessitate intensive collaboration on a given topic. Prototypical scenarios include the following:

- Project-based courses where students are asked to work on actual design problems and have to articulate and keep track of the problem space as well as the design decisions they make.
- Inquiry-oriented settings where students are asked to describe and analyze a particular system or phenomenon (e.g. needs assessments, evaluation studies or empirical studies of user behavior).

- Topic- or theory-related settings where students are asked to (re-)construct scientific arguments and/or to structure a particular domain of interest (e.g. collaborative literature reviews or state-of-the-art analyses).

Against this background the tools we are currently developing are designed to meet the following goals:

- The creation and use of visual models and their underlying languages should be as integrated as possible. Instead of treating modeling as a separate activity, collaborative modeling should be as tightly integrated into the groups' work processes as possible, allowing for easy access and reference to other resources used. Towards that end, tools for collaborative modeling should be directly integrated into the respective learning and working environment.
- Rather than restricting users to a predefined set of modeling languages, they should be able to modify existing or create new languages whenever needed (e.g. when modeling). To allow for an integrated work on visual models and modeling languages users have to be able to easily move between both levels of abstraction without mixing them up.
- Users should be assisted in developing alternative models and to support the triangulation of different perspectives; these models can be based on the same or different modeling languages.
- Supporting long-term and boundary crossing processes of knowledge creation affords the reuse and evolution of the visual models and languages. Towards this end, users have to be aware of existing models and languages, to understand their specific purposes but also to adapt them to their local circumstances and own ideas.
- Allowing users to create and maintain their own modeling languages also requires powerful metaphors and easy to use tools to overcome the formalization barrier imposed by current tools.
- As concepts and their interrelations often become apparent and crystallize only over a series of consecutive refinements and applications, learners should be supported in the systematic refinement and enrichment of models and their underlying languages. In order to trace the rational of their evolution means for comparing successive versions of models and languages have to be in place. Furthermore, whenever feasible, feedback should be provided to learners regarding possible consequences that a suggested change of the language will have for their model.

IV. KNOWLEDGE PRACTICES ENVIRONMENT AND VISUAL MODELING TOOLS

The Knowledge Practices Environment (KPE) [21] is a web-based collaborative working and learning environment offering various facilities for creating and interacting with knowledge artifacts and knowledge process models. The prevailing user interface paradigm in KPE is based on "shared spaces" – a graph-based view on collections of knowledge objects, which provide real-time and history based awareness to facilitate multi-user collaboration (Fig. 1).

KPE offers various tools for evolving the shared knowledge objects and organizing them visually – wiki, note and sketch editors, annotation and commenting tool, one-to-one and group chats, semantic tagging and semantic search.

A. KPE Architecture and Technical Implementation

The Knowledge Practices Environment is a web based distributed software system, which consists of user client, real-time synchronization service, front-end services and data storage (Fig.2).

The client is implemented in Adobe Flex/Flash. When the

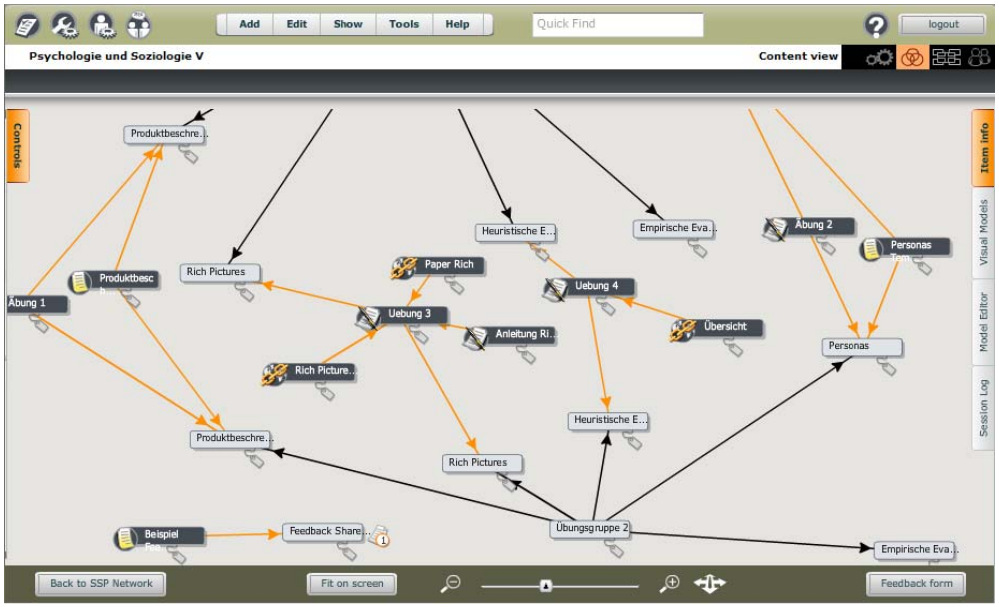

Figure 1. KPE Shared Space.

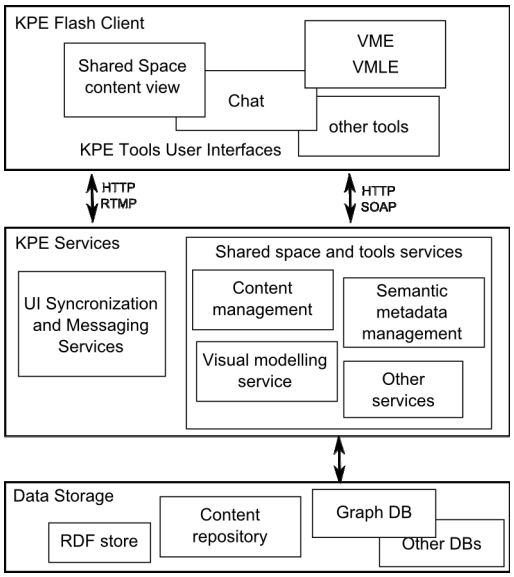

Figure 2. KPE architecture.

user initiates a KPE session, the client is loaded in the web browser and provides a graphical view of the shared space content. The client relies on the front-end services for data manipulation, retrieval and persistence tasks. Most of them are implemented as JAX-WS web services and deployed in Java EE application servers. This architectural decision allows for a truly distributed KPE deployment with different services residing on different servers.

KPE knowledge objects are stored into a heterogeneous database, consisting of RDF store [22] for metadata, Jackrabbit repository [23] for the content (uploaded files, notes, sketches, chat histories), wiki engine and some tool-specific data stores.

A dedicated server, running Adobe Flash Media Server software, is tasked with synchronizing the client's views. All users of a particular shared space can see the changes in the shared space content and layout, inflicted by other users, almost simultaneously.

B. KPE Visual Modeling Tools

The Visual Model Editor (VME) and Visual Modeling Language Editor (VMLE) are KPE tools, whose purpose is to allow users to create visual models and the underlying visual modeling languages as another type of shared artifacts. The direct integration into the KPE allows for an easy transition between modeling and other collaborative activities.

The KPE visual modeling tools support both synchronous and asynchronous collaboration. Users can work on the same model at the same time and changes are propagated in real time to all members of the group or users can work on the same model at different times and when they get back to the system they can retrieve the changes. Moreover visual models can evolve not only based on direct user-inflicted changes but also because the underlying visual modeling languages can also evolve.

Both collaborative developments of visual models as well as the underlying modeling languages are supported. The users

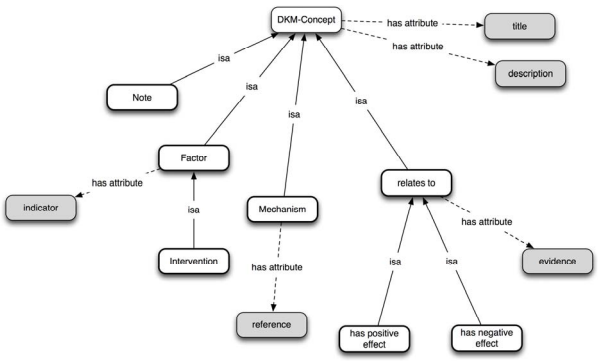

Figure 3. Example of a visual modeling language tree representation.

are free to specify the semantics of the modeling elements used which facilitates the creation of a shared understanding. The tools allow for a controlled evolution of a modeling language preserving consistency with selected existing instances of it.

In KPE the visual modeling languages are modeled as trees, with vertices symbolizing language concepts and attributes and edges representing *is-a* and *has-attribute* relations (Fig. 3).

The KPE users employ the Visual Modeling Language editor to modify the language tree by adding/removing concepts, changing their attributes and defining constrains on the way concepts can be linked together in the visual models.

The visual models are represented as directed multigraphs. Each model is based on a particular modeling language and is constructed from the concepts, defined in this language.

Fig. 4 depicts the user interface of the KPE Visual Model Editor. On the figure, the visual model elements can be identified as white rectangles with arrows between them. The VME allows for references to be created from the model elements to the knowledge objects in the shared space (shown as darker rectangles in the background). The figure also shows a context-chat window, which allows the users, working on the same visual model, to collaborate more effectively.

In order to support the research activities focused on analyzing the evolution of visual models and languages, both VME and VMLE tools keep detailed logs of all changes to the language or model graphs.

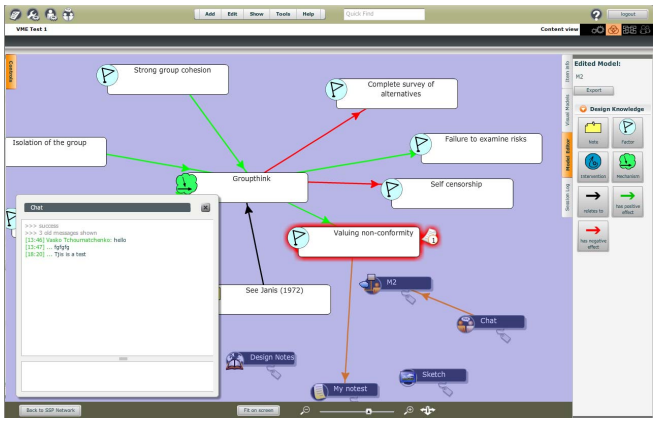

Figure 4. KPE and Visual Model Editor user interface.

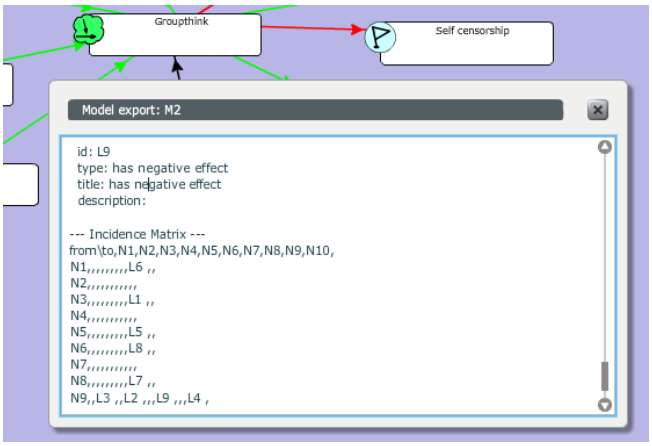

Figure 5. Visual model graph and history log export.

The logs, as well as the models, can be exported in textual format (Fig. 5) for detailed analysis outside the KPE.

The user interface and the presentation logic of the KPE visual modeling tools is implemented as part of the KPE Flash client. A dedicated web service handles the information exchange between the VME/VMLE client components and the models database. The language and model graphs are stored in an embedded graph database, based on Neo4j [24] persistence engine.

V. First Field Trial and Findings

To better understand the appropriation and utilization of the VME in an educational context, the first release of the VME has been tested in two university courses held by the University of Helsinki as well as the University of Applied Sciences Upper Austria in winter-term 2008/09. In the following we provide a brief description of the pedagogical scenario and findings obtained from the field trial at the University of Applied Sciences Upper Austria in the bachelor program "Communication and Knowledge Media".

The field trial was carried out in the context of cornerstone course "eModeration" that is aimed to promote an understanding of design as a form of open-ended inquiry and to engage students in a concrete design activity. Throughout a period of one semester the students are asked to envision, develop, implement and evaluate a solution for a complex design problem in the fields of eCommunication and eModeration. To foster in depth exploration of the design space and to trigger reflection on the proposed solutions, the teams were asked to continuously explicate their evolving understanding of a common design space by means of the VME. Towards this end students were provided with an introduction to principles and methods of needs and problem analysis as well as access to the KPE and the Visual Model Editor. A simple modeling language was introduced to scaffold the needs and problem analysis. Throughout a period of five months 35 students in 10 project teams took part in the field trial. Throughout the semester, the groups met face to face with the instructor alternately every second week.

The main aim of the field trial has been to better understand how students actually make use of the Visual Model Editor and the semantics of the language provided as well as to inform the further development of the VM(L)E. The following observations and findings are based on an exploratory analysis of the models created by the project teams (screenshots had been taken on a weekly basis), the responses to a questionnaire administered to all students at the end of the course, as well as interviews with representatives of four teams and the instructor.

Besides some usability problems, the accompanying research study also shed light on the actual modeling practices. In general the project teams responded quite differently to the modeling assignment, which is reflected for example in the number of additions and modifications made to the models over time as well as their structure. While three groups hardly made any use of the Visual Model Editor or abandoned it after some first tryouts, the other seven groups worked on their models fairly continuously. Among those who used the Visual Model Editor more regularly two groups worked on a single model, while the remaining four groups created 3-7 distinct models.

Even though the "Problem Analysis Language" had been introduced explicitly to the students and scope notes for the different concepts are easily accessible via the Visual Model Editor, the "specified factor", which provides a kind of default concept, was used quite excessively by all teams. Especially those groups who started to work on their models right from the beginning, hardly made any use of the specific concepts provided to depict their design space. This behavior only changed later on after the instructor provided additional suggestions on how the different concepts could be used efficiently. In contrast, two teams that started to work on their models relatively late, made more sophisticated use of the different concepts available right from the beginning. This finding is partly in conflict with the expectation that explicitly defined concepts would scaffold students' elaboration of the design space. It might be that in the early phases of the design process, the effort to explicitly classify ideas according to a predefined scheme does not outweigh the expected benefits or even hinders a brainstorming like collection of ideas. This assessment might change later on when the scope of the project becomes clearer and there is more need to structure and integrate the existing ideas. This interpretation is at least partly supported by the students' reports on how they created and used the models at the different stages of the project.

Some students also mentioned that the predefined modeling languages are too restrictive and that they had difficulties to map their ideas to the concepts provided. Closer examination of the models revealed that several teams had problems to make proper use of the concepts provided e.g. they mixed up resources and actions, and/or to understand the idea of typed nodes e.g. they introduced a concept specified in the Problem Analysis Language as a separate node. On the other hand we also found attempts to proactively introduce additional concepts not provided by the Problem Analysis Language (the Visual Modeling Language Editor has not been available yet). For example on of the teams added a kind of prefix, in this case "problem", to the title of several nodes providing additional categorization. Together with findings from similar courses this underlines the need for a possibility not only to edit the models but also the underlying modeling languages.

The comparison across teams revealed that those who used the Visual Model Editor more intensively often created more than one model in the project's lifetime. While in some cases the creation of multiple models was reportedly due to the fact that the current release of the Visual Model Editor does not allow changing the type of a concept or relationship in a model once it has been created, which make the update of models somewhat tedious, other groups created different models by purpose. One of the interviewee's reported that they had created multiple models in order to be able trace back their understanding at different stages. In another case the participants reported that they produced different models to depict different aspects of their project.

Even though these findings are only preliminary and more in depth analysis of students' modeling practices is needed, it appears that the adoption and utilization of semantically rich visual models might strongly depend on the direct added value for the user. Nevertheless, the assessment of required efforts and expected benefits might change depending on the stage of the project as well as the actual task at hand. Consequently, tools to support collaborative modeling have to be quite flexible in order to accommodate for the changing requirements that arise during the lifecycle of a project. For instance, it requires that users can easily restructure the models they created as well as change the concepts and relations or even introduce new concepts at runtime, which could extend as far as modifying the modeling languages itself. Furthermore, the findings point to the complexity of collaborative modeling as a real world practice that might not only fulfill an epistemic but also a social, pragmatic or even reflective purpose.

VI. CONCLUSIONS & FUTURE WORK

In this paper we briefly discussed the role of collaborative modeling in the context of design and engineering education and outlined our primary design goals. Against this background we introduced the Visual Model Editor and the Visual Modeling Language Editor as a set of tools aimed to provide users with a flexible and easy to use but still semantically powerful tool for the creation of visual models and their underlying modeling languages. The vision behind this tool is to provide users with the possibility to create their own conceptual tools and thereby to advance pre-existing perspectives while being engaged in a meaningful activity, such as a design project. Based on findings from a first field trial with the Visual Model Editor, it appears that the adoption and utilization of semantically rich conceptual models to a large extend depends on the direct added value for the user, while at the same time modeling fulfils not only epistemic but also social and pragmatic purposes for the user.

An updated version of the Visual Model Editor as well as a first release of the language editing component are currently under development and will provide users with full capability to create and update their own modeling languages in the near future. In parallel, in depth analysis of students' modeling practices and the impact of the modeling activities on the project outcomes are ongoing. These will inform both the further development of the tools as well as respective pedagogical scenarios.

REFERENCES

[1] W. Visser, „Both generic design and different forms of designing," DRS (Design Research Society) International Conference, November 1-5, 2006 Lisbon, Portugal.

[2] L.L. Bucciarelli, "Between thought and object in engineering design," Design Studies, vol. 23, pp. 219-231, 2002.

[3] F. Détienne, "Collaborative design: managing task interdependencies and multiple perspectives," Interacting with Computers, vol. 18, no. 1, pp. 1-20, 2006.

[4] C.L. Dym, A.M. Agogino, O. Eris, D.D. Frey, L.J. Leifer, "Engineering Design Thinking, Teaching, and Learning," Journal of Engineering Education, vo. 34, pp. 103-120, 2005.

[5] J.R. Mathias, „A study of the problem solving strategies used by expert and novice designers", PhD Thesis, University of Aston, Birmingham, UK, 1993.

[6] C. Ho, „Some phenomena of problem decomposition strategy for design thinking: differences between novices and experts," Design Studies, vol. 22, pp. 27-45, 2000.

[7] A.T. Purcell, J.S. Gero, „Design and other types of fixation," Design Studies, vol. 17, pp. 363-383, 1996.

[8] M. Mähring, M. Keil, "Information technology project escalation: a process model," *Decision Sciences, vol. 39*, no. 2, pp. 239-272, 2008.

[9] J. Stempfle, P. Badke-Schaub, „Thinking in design teams – an analysis of team communication," Design Studies, vol. 23, pp. 473–496, 2002.

[10] A.H., Dutoit, T. Wolf, B. Paech, L., Borner, J. Ruckert "Using rationale for software engineering education," CSEET, 18th Conference on Software Engineering Education & Training, April 18 - 20, 2005, Washington, DC, pp. 129-136.

[11] V. Kokotovich, "Problem analysis and thinking tools: an empirical study of non-hierarchical mind mapping," Design Studies, vol. 29, pp. 49-69, 2008.

[12] M. Van Kleek, M. Bernstein, M., P. André, M. Perttunen, D. Karger, M. Schraefel, "Simplifying knowledge creation and access for end-users on the SW". CHI 2008 Workshop on Semantic Web User Interfaces, 2008.

[13] J.D., Novak, D.B. Gowin, "Learning How To Learn," New York, Cambridge University Press, 1984.

[14] H.W.J. Rittel, „Second Generation Design Methods," in Developments in Design Methodology, N. Cross, Ed. Chichester: J. Wiley & Sons, 1984, pp. 317-327.

[15] A. MacLean, R.M. Young, V. Bellotti, T. Moran, „Questions, Options, and Criteria: Elements of Design Space Analysis," Human-Computer Interaction, vol. 6, pp. 201-250, 1991.

[16] L., Vanasupaa, E. Rogers, K. Chen, "Work in Progress: How Do We Teach and Measure Systems Thinking?" Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference, October 22 – 25, 2008.

[17] T. Knuuttila, "Models as Epistemic Artefacts: Toward a Non-Represenationalist Account of Scientific Representation," PhD-Thesis. University of Helsinki, Helsinki, Finland, 2005.

[18] D.D. Suthers, "Collaborative Representations: Supporting Face to Face and Online Knowledge Building Discourse," Proceedings of the 34th Hawaii International Conference on the System Sciences (HICSS-34), January 3-6, 2001, Maui, Hawaii (CD-ROM).

[19] G. Guizzardi, L. Ferreira Pires, M. van Sinderen, M. "Ontology-Based Evaluation and Design of Domain-Specific Visual Modeling Languages," Proceedings of the 14th International Conference on Information Systems Development, Karlstad, Sweden, 2005.

[20] C. Floyd, S. Ukena, "On Designing Situated Ontologies for Knowledge Sharing in Communities of Practice," 1st Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'05). June 13th, 2005, Porto, Portugal.

[21] The KP-Lab website. [Online], ICT-27490: Knowledge Practices Laboratory (KP-Lab), Available:www.kp-lab.org/

[22] The KP-Lab website - RDF Suite. [Online], http://www.kp-lab.org/tools/rdfsuit , accessed on January 2010

[23] The Apache Jackrabbit website, [Online], http://jackrabbit.apache.org

[24] Neo4J web site [Online],http://neo4j.org, accessed January 2010