Delivering Relevant Training Objects to Personal Desktop with Modern WBT-Systems

Abstract: In order to take a part in a particular Web-based training session students work with a number of tools reflecting a particular training strategy. By doing so they access different training objects containing relevant information for their current training task. Since modern WBT systems support dozens of different, sometimes rather complex tools and provide access to thousands of training objects students are often confronted with a complex user interface, to say at least. Moreover, users who prepare training sessions for modern WBT systems, that is, authors and tutors experience similar problems of a complex, inscrutable user interface. This paper describes a simple general WBT user interface solution that we developed in order to overcome such problems. Additionally, the paper provides an evaluation of the responses from users that we gathered after we applied that solution in one modern WBT system – WBT-Master. This evaluation showed us that there was much more potential in our solution than we believed at the first, resulting in the evolution of our user interface solution to a simple knowledge delivery tool. By applying this tool in our training sessions we achieved the following. Only the most relevant and up-to-date training objects residing in the system could become a part of our training sessions. Moreover, these training objects were always accompanied with conceptual maps describing their relationships with other training objects in the system.

1. Introduction

Usually, Web-based training (WBT) systems (Gaines, Norrie and Shaw 1996; Shaw and Gaines 1996) support a number of different user roles (Helic, Maurer and Scerbakov, 2000). Standard user roles include authors, tutors, learners and administrators (Helic, Maurer and Scerbakov, 2000). Normally, administrators are considered as a special kind of so-called "power" users that keep a WBT system running in a flawless manner. On the other hand, authors, tutors and learners are viewed as "real" users of the system. These users have the following basic roles with regard to the system (Helic, Maurer and Scerbakov, 2000; Helic, Maurer and Scerbakov, 2001c; Helic, Maurer and Scerbakov, 2001d):

- Students participate in different online training sessions offered by the system;
- Tutors conduct and manage such online training sessions in the system;
- Authors prepare these online training sessions.

As we can see here, from the users' point of view WBT systems deal with online training sessions.

Conceptually, a WBT training session consists of a number of training objects and a particular training strategy (Helic, Lennon, Maurer and Scerbakov, 2001b; Helic, Maurer and Scerbakov, 2001c; Helic, Maurer and Scerbakov, 2001d). Training objects might be simple Web pages, learning courses, discussion forums, etc. On the other hand, a

particular training strategy reflects a particular way of working through the subset of training objects to achieve a particular training goal (Helic, Lennon, Maurer and Scerbakov, 2001b; Helic, Maurer and Scerbakov, 2001c; Helic, Maurer and Scerbakov, 2001d). Such a training strategy might be considered as a certain training methodology.

Technically, a training methodology is implemented as a collection of WBT tools combined to support the desired way of working with the WBT system. For example, consider the basic WBT training strategy known as Web-based learning. This training strategy is a collection of WBT tools, which provide functionality needed to work with simple WBT learning courses (Andrews, Nedoumov and Scerbakov, 1995; Gaines and Shaw, 1997).

Let us now look on how each of the above mentioned user roles maps onto this basic WBT training strategy. Basically, authors implementing Web-based learning strategy need to create a number of simple learning courses. Generally, learning course is just a collection of Web pages interrelated by means of computer-navigable links in a navigational structure. The most used navigational structure is a simple navigational sequence, where Web pages have links to the prior and next member in such navigational sequence. Obviously, tools needed to support the authoring process of such learning courses can be quite simple. The only functionality that is needed is just a simple adding and ordering of Web pages in a navigational sequence. Once when the navigational sequence is created it can be published in the system. In that way the learning course becomes accessible for students who in turn just use simple navigational tools (prior/next links, course map, etc.) to work with that course. Finally, tutors may monitor students' progress by using simple statistics tools or may communicate with students via discussion forum. As we can see, tools provided by such systems to support this basic training strategy are rather primitive offering just a simple basic functionality. Obviously, user interface of these tools is usually simple, easy-to-use, comprehensive and highly usable.

Usually, standard WBT systems support just the basic WBT training strategy. On the other hand, more advanced WBT systems support not only the basic WBT training strategy but also a wide range of other, more advanced training strategies.

Let us look now on the following example of a more advanced WBT training strategy implemented in a modern WBT system called e-Learning Suite (Dietinger and Maurer, 1997; Dietinger and Maurer, 1998). e-Learning Suite supports a training strategy called organizational learning (Dietinger et al., 1999a). Organizational learning is supported by sophisticated WBT tools, which provide the access to the so-called Corporate Memory (Dietinger, Gütl and Pivec, 1999b), made up of information that is found within a company in the form of not only learning courses, but also other documents, discussion forums, meta-data collections, meta-data schemas, etc. Here authors are responsible for managing the entire Corporate Memory and this authoring process may include authoring of text documents, Web pages, meta-data information, discussion forums, etc. As we can see, such authoring process includes not only authoring tools for simple learning courses, but also authoring tools for other training objects (e.g. meta-data collections, meta-data schemas, etc.). Note that these tools can be of quite substantial complexity (e.g. meta-data

schema tools). On the other hand, tutors are also involved in managing the Corporate Memory by trying to add additional information to it. Thus, they usually search the Web to find useful links to add it to the Corporate Memory, they can write their own comments to information that is already existent in the Corporate Memory etc. Obviously, tools needed to support tutors in this WBT training strategy must provide such advanced functionality. Finally, students participating in training sessions involving the usage of the Corporate Memory are confronted with a wide range of different and possible complex tools (viewers for documents, meta-data search engines, complex navigational tools, etc.). Obviously, applying this WBT training strategy has its consequences for all three kinds of users. Now they are all confronted with more numerous tools with more complex functionality, which leads to a more complex, harder-to-use, less comprehensive user interface.

If we look on another example we see a similar picture. Another advanced WBT system called WBT-Master (Helic, Maurer and Scerbakov, 2001a; Helic, Lennon, Maurer and Scerbakov 2001b; Helic, Maurer and Scerbakov, 2001c; Helic, Maurer and Scerbakov, 2001d; Helic, Maurer and Scerbakov, 2002) supports a wide range of different WBT training strategies, such as Web-based tutoring (tools leading students through a sequence of learning actions to a particular learning goal) (Helic, Lennon, Maurer and Scerbakov, 2001b), Web-based mentoring (tools supporting synchronous online mentoring sessions) (Helic, Maurer and Scerbakov, 2001d), and so on. Again supporting the above mentioned user roles in the context of these training strategies is achieved through a number of WBT-Master tools with a rather complex functionality and user interface. Thus, to create, update and publish a so-called learning goal as a sequence of learning actions (Web-based tutoring training strategy) authors must use a special set of WBT-Master tools, which provide functionality to create learning actions, attach training objects to such learning actions, compose learning actions in a time-dependent sequence, define prerequisite learning actions for certain other learning actions, etc. Tutors must be able to monitor the progress of students, write comments, reassign certain learning actions to students progressing slowly, open other learning action for more successful students, etc. Finally, students must work not only with particular training objects (e.g. learning courses) attached to different learning actions, but also they have to work through the learning actions defined for their particular learning goal. This, of course, includes a completely new set of navigational tools for navigating the orthogonal structure of a learning goal. Again, in order to work with such online training sessions users are confronted with a large number of tools with a rather complex functionality and user interface.

Obviously, implementing advanced training strategies supported by means of modern WBT systems results in a rather complex user interface for all kinds of users of such systems. That is users of advanced WBT systems might be confronted with a serious problem of a complex, inscrutable user interface. This is even truer if we consider that sometimes we do not need just one advanced training strategy to implement our training sessions, but rather we need a kind of heterogeneous training strategy combined out of a number of other advanced training strategies. Such implementations of training sessions can be considered as quite normal in everyday applications of modern WBT systems.

Evidently, if we want to be able to support training session by means of advanced training strategies we need to solve the increasing user interface problem. Moreover, to support heterogeneous training strategies we need to solve this problem on an even larger scale. Hence, not only do we need to simplify the user interface of a particular tool but we need also a more general user interface solution that is able to reflect all peculiarities of any training strategy and provide a single access point to each of tools needed to implement a particular training strategy. Apparently, such solution needs to be highly configurable, customizable user interface solution.

The above-mentioned modern WBT system called WBT-Master provides such general user interface solution in the form of so called Personal Desktop.

2. Personal Desktop

Conceptually, a personal desktop is just a set of folders containing references to designated training objects and WBT-Master system tools.

Training objects from a particular personal desktop folder are always equipped with a set of WBT-Master tools. These tools reflect a particular training strategy, implemented in order to lead students to the desired training goal. For example, clicking on a learning course that is a part of a personal desktop folder pops up a new window including all navigational tools needed to work with that learning course.

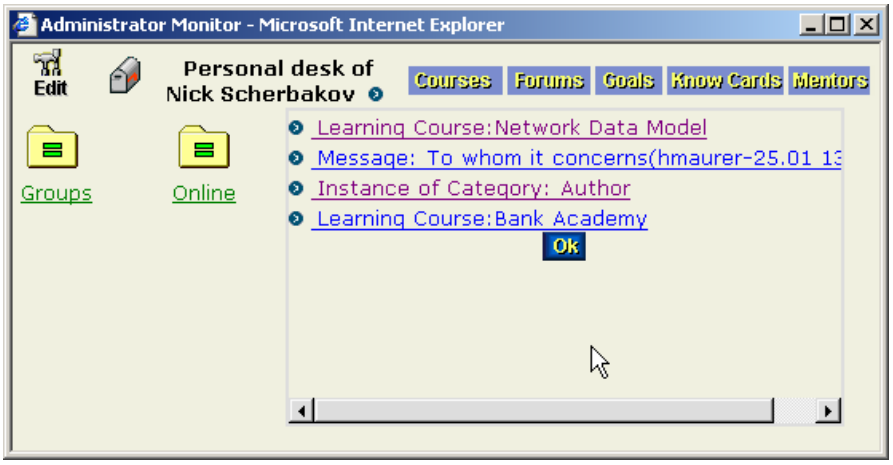

Figure 1: Content of a personal desktop folder

On the other hand, WBT-Master system tools are tools that are needed to create certain training objects and implement training strategies. Normally, authors and tutors in order to prepare and conduct online training sessions use such tools.

Let us now look how Personal Desktop may be used to solve the above mentioned user interface problem in modern WBT systems.

Firstly, let us look on Personal Desktop from the students' perspective. The main idea behind the concept of a personal desktop might be seen as the following. An advanced user, say an author or a tutor manages a particular training session with a group of students. He/she creates a personal desktop folder containing different WBT-Master tools and a number of training objects residing on the server. Once when the personal desktop folder has been created he may share that folder with the students' group, thus allowing them to participate in his/her training session. Obviously, by accessing the created personal desktop folder students have access to designated training objects by means of the implemented strategy. However, students are not any more confronted with arbitrary tools or training objects provided by the system. Rather they are supposed to access only few relevant training objects and that by means of tools reflecting the desired training strategy.

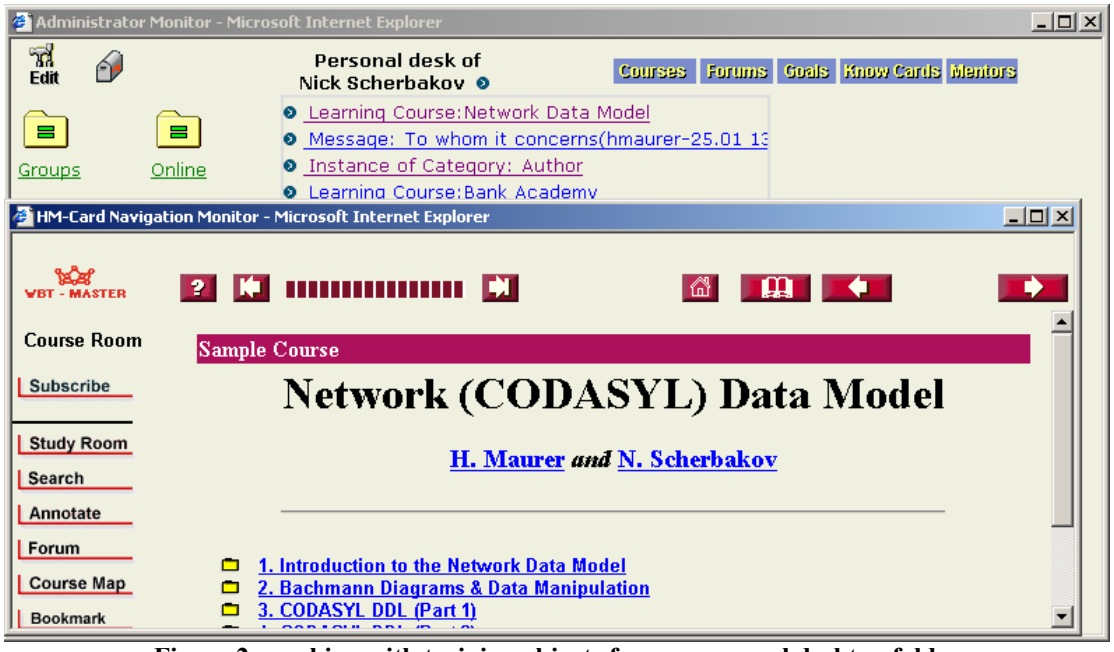

Figure 2: working with training objects from a personal desktop folder

This concept greatly facilitates the fact that a typical student works with just a few tools and training objects offered by the system. In other words, this special customization mechanism is used to adjust the rich system functionality to personal needs of a particular student or a group of students. Hence, a tutor or an author may decide on preferable user interface for such group of students and on training objects, which are needed to accomplish a particular training task.

Another important aspect of the personal desktop concept is collaboration facilities, which are provided by so-called shared folders and internal messaging system. For example, a student group may share a certain folder to put all contributions of the group members into it. In this way, the contributions may be easily accessed by group members and discussed by attaching messages to such contributions.

Now let us look how Personal Desktop solves the user interface problem for authors and tutors. Basically, authors may apply Personal Desktop in two ways.

Firstly, authors create one personal desktop folder for each training session that they want to prepare. As the next step they add training objects for that training session. After the training objects are inserted authors can add all tools that are needed to implement the training strategy that he had in mind. Note, that some of tools are added automatically by the system according to the type of training objects that were selected. For example, whenever authors insert a learning course, some basic navigational tools like course maps, course syllabus, etc. reflecting Web-based learning strategy are automatically added to the personal desktop folder. Of course, authors are free to refine this training strategy by adding other tools. For example, to make their training session more interactive authors may add WBT-Master chat tools to that personal desktop folder. That means that a skeleton for a particular training strategy is actually automatically implemented by the system, and authors are there just to refine it, as they want. Note also, that a heterogonous training strategy is easily implemented by the same mechanism. For example, an author adds a learning course to the current personal desktop folder. The system attaches some tools required for the Web-based learning strategy. As the next step, the author adds a learning goal; again a number of tools needed to support Web-based tutoring strategy are added. Finally, the author adds a number of other WBT-Master tools that make his training strategy complete.

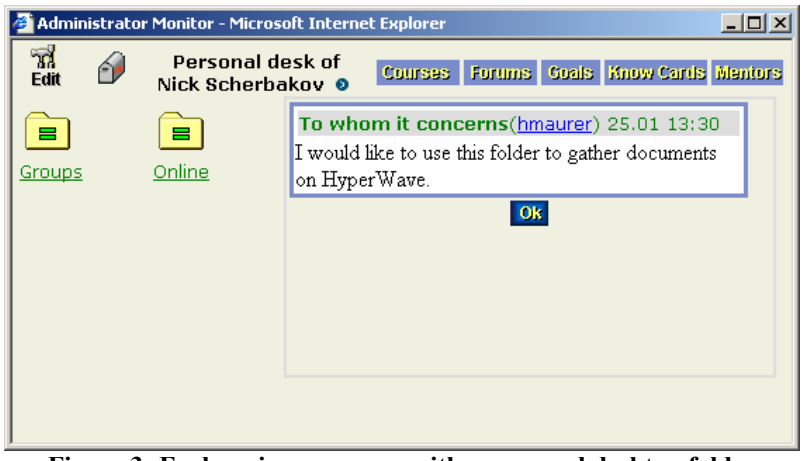

Figure 3: Exchanging messages with a personal desktop folder

The second way of how authors can use Personal Desktop can be described as follows. Suppose, an author needs to prepare a number of learning courses and a number of discussion forums. Discussion forums will be attached to learning course, thus providing boards for students to talk about issues form that learning course. Now, suppose an advanced "power" user (administrator) creates a personal desktop folder where he puts all WBT-Master system tools that are needed to create learning courses and discussion forums. He then shares his personal desktop folder with the author, thus providing the author with just those WBT-Master tools that are really of interest for him. Basically, in this way the administrator does for that author exactly the same what the author does for his students by sharing his personal desktop folder with them. The administrator insures that the author is not any more confronted with arbitrary WBT-Master system tools

totally irrelevant for his current task. Rather the author gets only those tools, which are really needed for his work.

Finally, let us look on Personal Desktop from tutors' point of view. The most important advantage of Personal Desktop for tutors can be stated as follows. Obviously, a personal desktop folder provides a very compact context for a student group that is participating in that particular training session. This context provides tutors with all tools that are needed to conduct the training session in a flawless manner. For example, tutors have possibility to use messaging system included in the personal desktop folder to communicate with students. Further, they can easily add their own training objects to the personal desktop folder if they find it suitable. In that way they have a great impact on the training session, since they are immediately responding to the needs of their students by adjusting the training strategy on the run. Similarly to authors, an administrator can share his personal desktop folder containing WBT-Master system tools to allow tutors to have even more influence on the training session. For instance, tutors can use such system tools to create a new discussion forum and add it to the training session if there is a lot of discussion going on. It is important to note here that tutors, similarly to authors and students, are confronted just with those tools that are really relevant for them in the scope of that particular training session.

3. Working with Personal Desktop

Generally, first experiments with personal desktop show a great acceptance of this concept by users.

Authors very much appreciated a possibility to define all requisites of a particular training session, i.e., all tools that are needed to implement a particular training strategy, as well as all relevant training objects as members of a folder. In this way they were not supposed to know all peculiarities of the system in order to combine the system tools into a coherent training strategy. For instance, they were not supposed to know how to attach a discussion forum to a learning course to provide learners with a board to discuss topics from that learning course, but rather they just put that learning course and a discussion forum into a personal desktop folder and the above-mentioned relationship between these objects was automatically established by the system.

On the other hand, tutors liked the messaging functionality of a personal desktop folder the most. This functionality allowed them to keep the communication between members of a particular learner group within the scope of a particular training session. Also, possibility to reuse (for instance, in the form of FAQ) the results of such communication was quite well accepted by tutors.

Finally, learners appreciated the simplicity of user interface as offered by such training sessions very much. Not any more did they need to find their way through dozens of system tools and large number of training objects. Rather they just worked with a few tools and relevant training objects combined into a simple navigable list.

However, we noticed some disadvantages of this concept as well. The most important shortcoming of such an approach can be stated as follows. A particular training strategy, i.e., a particular collection of tools that implements that training strategy might be seen as a rather static entity. That means that a particular training strategy is likely not to change in the course of the time. For instance, consider the above-mentioned Web-based learning training strategy. This strategy consists always from say a number of learning courses and a discussion forum attached to these courses. What really changes in training sessions is training objects. Actually, training objects might be seen as a very dynamic entity. Not only that in different training sessions training objects are completely different but also in one and the same training session training objects are likely to change. For instance, some training objects might become obsolete, newly updated versions of training objects might be created, etc.

These results led us to the conclusion that we need to extend the concept of personal desktop and redesign it in a way that it becomes robust to such changes. Thus, we decided to incorporate a mechanism that would be able to automatically select training objects that are the most relevant and the most up-to-date training objects. Only such training objects should become members of a particular personal desktop folder. Obviously, such mechanism must be able to "reason" about training objects and decide whether a particular training object is relevant to a particular training session, i.e., does it match certain criteria posed by that training session.

Another important disadvantage of the personal desktop can be stated as follows. A personal desktop can hold a number of different training objects. Usually, these training objects are selected for a certain students' group that has certain preferences. For example, students' group taking online training sessions on "Databases" is provided with a number of training objects dealing with relational, network, and object- oriented databases. Although these training objects are different they have certain properties in common, especially on a conceptual level. A standard way to describe such properties is to apply some standard knowledge representation technique. Thus, we usually describe concepts, relationships between such concepts and associate training objects to such concepts and relationships to get a comprehensive semantic overview of these training objects. Such overviews are not only useful for students in their learning process, but for tutors and authors as well. For example, authors contributing with new training objects to the system may first consult the overview to see if a particular subject was covered by other training objects and to what extent. Further, once when they add the new training object to the system they can add it to the semantic overview on a proper position, by assigning it to a particular concept and relating it to other training objects vie semantic relationships.

Such consideration lead us to the conclusion that incorporating such a mechanism into Personal Desktop would tremendously improve the overall quality of the tool. We would not only have the most up-to-date training objects becoming members of a particular personal desktop folder, but also these training object would be enhanced with a global map of training objects, telling us what is a relative semantic position of this training

objects to other training objects in the system in general, and in that training session in particular.

Before we proceed with description of how we incorporated such approaches into Personal Desktop concept we will describe in more details how these two processes are supported by means of WBT-Master tools.

4. Selecting the most relevant training objects – Knowledge Mining

In WBT-Master we captured the dynamics of change in training objects by the so-called Knowledge Mining process (Helic, Lennon, Maurer and Scerbakov 2001b). This process is supported by an alternative way of accessing relevant training objects based on so-called Knowledge Cards (Helic, Maurer and Scerbakov 2001a; Helic, Lennon, Maurer and Scerbakov 2001b). The idea behind this concept is rather simple: Knowledge Cards allow the definition of a conceptual view of the server in the form of a collection of Knowledge Cards. A *Knowledge Card* is a description of particular concept (i.e. *semantic entity*). For example, a semantic entity "Database technology" may be seen as a Knowledge Card. In WBT-Master, Knowledge Cards may be combined into a *semantic network* (Lambiotte et al. 1984; Nosek and Roth 1990) using just one type of relationship: "is a part of". Inverse relationships may be called "consists of". For example, the Knowledge Card "Relational Data Model" may be related as "is a part of" to the Knowledge Card "Database Technology".

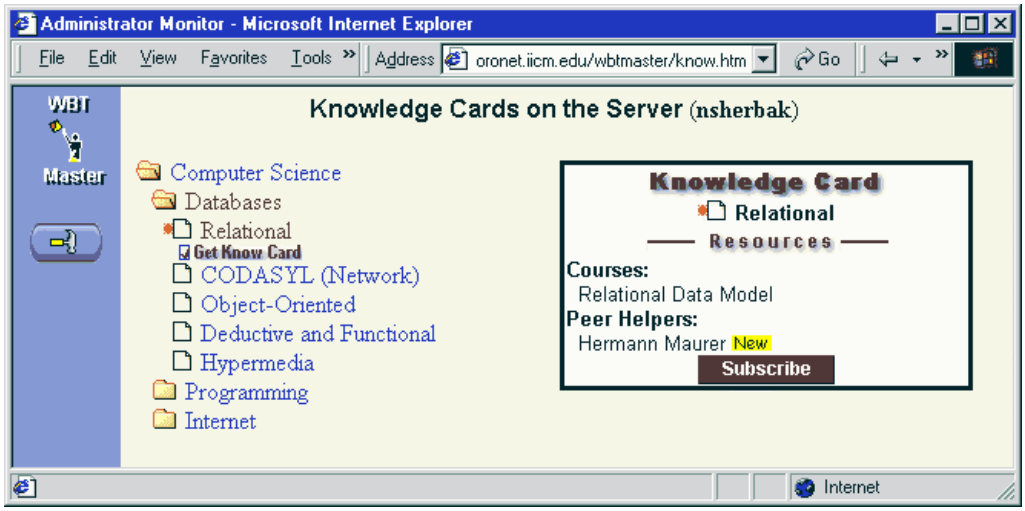

Figure 4: Resources attached to a Knowledge Card

Thus, a Knowledge Card associated with some particular concept (e.g. "Database Technology") contains information on training objects for this concept and can be related to other Knowledge Cards using the semantic relationship "is-a-part-of". The semantic relationships essentially define a graph structure (as opposed to just a hierarchical one). For example, the same Knowledge Card "Relational Data Model" may be defined as a part of "Introduction to Oracle", "Information Systems", etc. To be more concrete, each Knowledge Card may provide access to a number of associated training objects. For

example, a learning course "Relational Data Model" may be associated with the Knowledge Card "Relational Data Model". In addition, some other learning courses, learning goals, discussion forums, documents, etc. may be associated with the same Knowledge Card.

Whenever a content provider contributes to the server with new material, he/she is supposed to associate it with one or more Knowledge Cards or create a new Knowledge card and place it into a proper position within the semantic network. Of course, a specially designated member of the server administration team (Knowledge engineer) may also do it.

Instead of browsing through countless training objects new users are supposed, in the simplest case, to browse the semantic net consisting of previously defined Knowledge Cards. We should especially mention the most important property of the semantic network - the possibility of inferring training objects using semantic relationships (Sowa, 1984; Sowa, 1991). Whenever a user accesses a Knowledge Card, the system infers all training objects that are associated with this particular Knowledge Card and with the Knowledge Cards related to this one. Thus, for example, suppose that there were no training objects associated with the Knowledge Card "Computer Science", but a number of other cards (say, "Databases", "Programming", etc.) were defined as "is a part of" "Computer Science". Accessing the "Computer Science" knowledge card will result in the training objects inferred from other related cards.

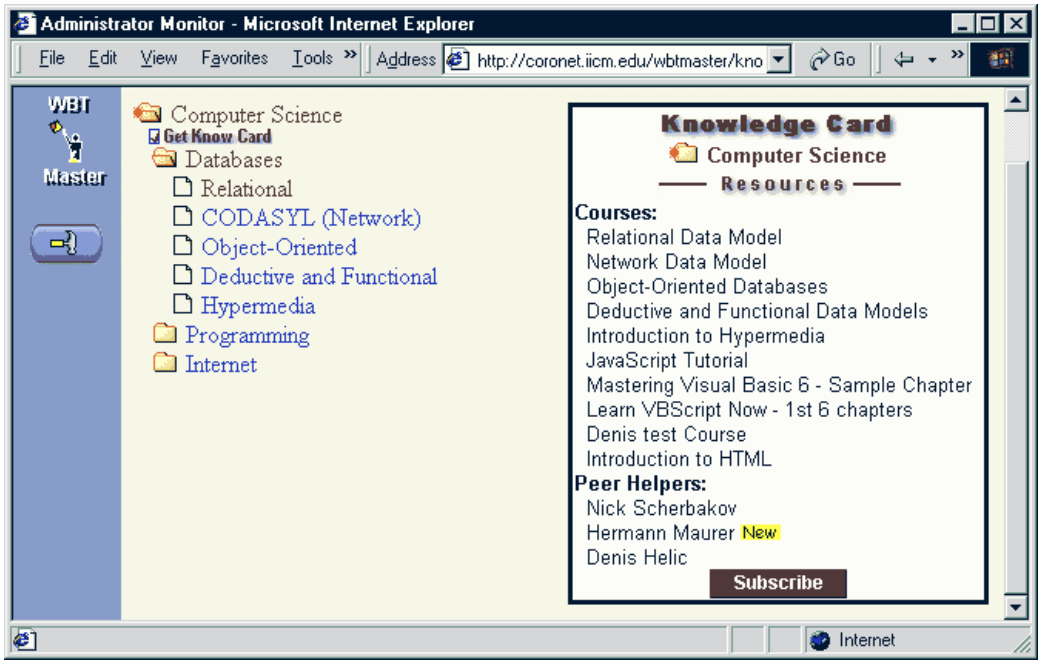

Figure 5: Training objects automatically inferred for a Knowledge Card

The situation discussed above, leads us to a very important conclusion. Students do not need to browse the whole semantic net. They might be interested in a Knowledge Card related to the most important previously defined concepts of interest. In this case,

whenever a student accesses such a Knowledge Card all relevant and up-to-date training objects will be inferred automatically.

5. Implementing global maps of training objects – Knowledge Profiling

Providing conceptual overviews (Sowa, 1984; Borgida, 1987; Borgida, 1991; Brodie, Mylopoulos and Schmidt, 1984; Minsky, 1968) of training objects in WBT-Master is supported by means of Knowledge Profiling process (Helic, Maurer and Scerbakov 2001b). The process of Knowledge Profiling may be explained through the following typical situation.

Suppose a faculty needs to install knowledge profiles for each of its courses. The process may be seen as defining a course and providing descriptions of the knowledge components required for that course. Moreover, the knowledge components may be further associated with training objects which provide such knowledge. On joining course students first take part in a number of Web-based training sessions. So, they must be able see what knowledge is needed (i.e. they should be able to browse the course profile) and automatically access the training objects.

WBT-Master supports such knowledge profiling of training objects in the form of Knowledge Domains (Helic, Lennon, Maurer and Scerbakov 2001b; Helic, Maurer and Scerbakov 2001c). Each Knowledge Domain is a set of training objects and/or other resources (e.g. Web pages, text documents, etc.) belonging to a number of predefined semantic categories. For the previously discussed example, we could speak about three semantic categories: “Course”, “Training Object” and “Know Component”. For instance, we can also say that a document “Course: Relational Database Management Systems” (describing the content of that course) is an instance of the category “Course”, a learning course “Relational Data Model” is an instance of the category “Training Object” and a document “Relational Calculus” is an instance of the category “Know Component”. Speaking in general terms, we can say that each semantic category is linked to a set of training objects that are called instances of the category. Thus, Knowledge Domains may be seen as a special knowledge representation technique (Brodie, Mylopoulus and Schmidt, 1984; Calvanese et al., 1998), similar to other frame-based knowledge representation schemas (Calvanese et al., 1998), adjusted specifically to the needs of a WBT environment (Burke, Hammond and Young, 1996).

A *Knowledge Domain Schema* may be seen as a definition of all categories and all possible semantic relationships between them.

The definition of a semantic category includes the definition of a number of attributes, which are properties of instances of the semantic category. An attribute is a standard key-value pair. The value of an attribute is defined to be of a specified type, i.e., a value may be a string, a number or a selection from a list of possible values. For example, the category “Course” may have two associated attributes: title (string) and lecturer (string).

Similarly, the category "Know Component" may have just one associated attribute – title of the component.

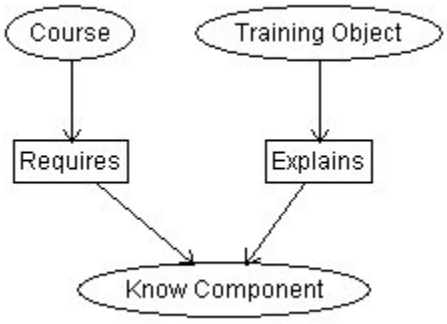

Figure 6: Schema of a Knowledge Domain

The Knowledge Domain Schema defines common properties of all the category instances. Any resource may be inserted (stored) into a particular Knowledge Domain as an instance of a predefined category. Thus, a responsible author simply selects an existing Knowledge Domain and a predefined category for a new resource and the system guides the author through the process of defining attributes and necessary relationships.

For example, if a new instance of the category "Know Component" is created, the system automatically requests the selection of a title (an attribute predefined for the category). It also provides references to the resource where that "Know Component" is described and a certain course for which this component is required. This, of course, facilitates the creation of well-structured repositories.

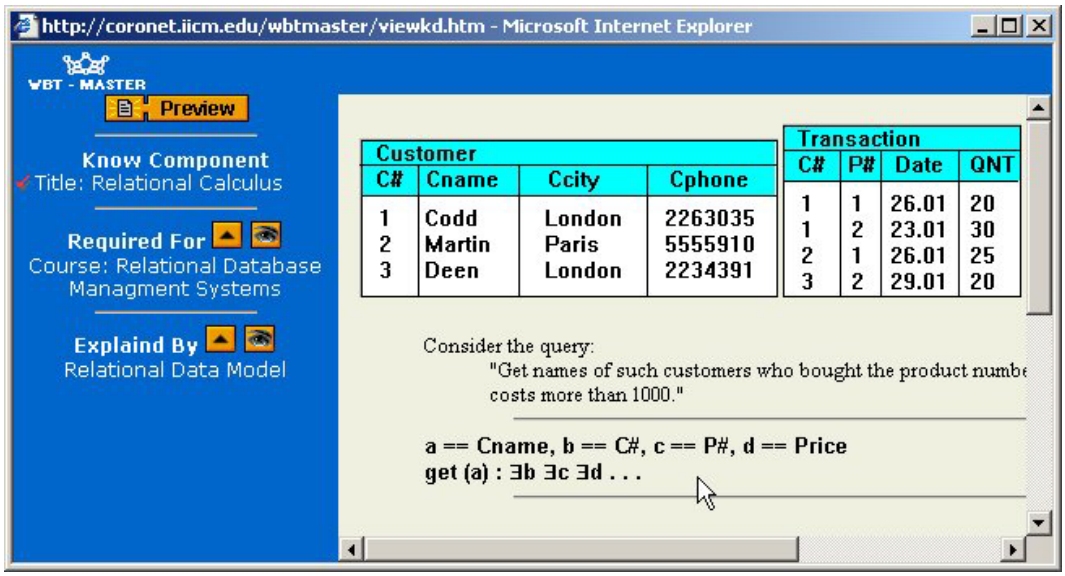

Figure 7: Browsing a Knowledge Domain

The concept of well-structured Knowledge Domains facilitates also browsing and searching of the training objects reused as instances of semantic categories. Thus, for example, whenever a user access the document "Relational Calculus", the system

automatically provides information on attributes attached to this document, references to instances of other knowledge categories which are related to this one, next/prior navigational tools, etc.

6. Extending Personal Desktop – Knowledge Delivery

The extension of the Personal Desktop concept treats Knowledge Cards and Knowledge Domains as training objects that might be added to a personal desktop folder.

Figure 8: Adding a knowledge card to a personal desktop folder

By adding a Knowledge Card as a training object in a personal desktop folder we achieve the following. Whenever users access a Knowledge Card from a personal desktop folder the system automatically infers all related and to that particular concept relevant training objects.

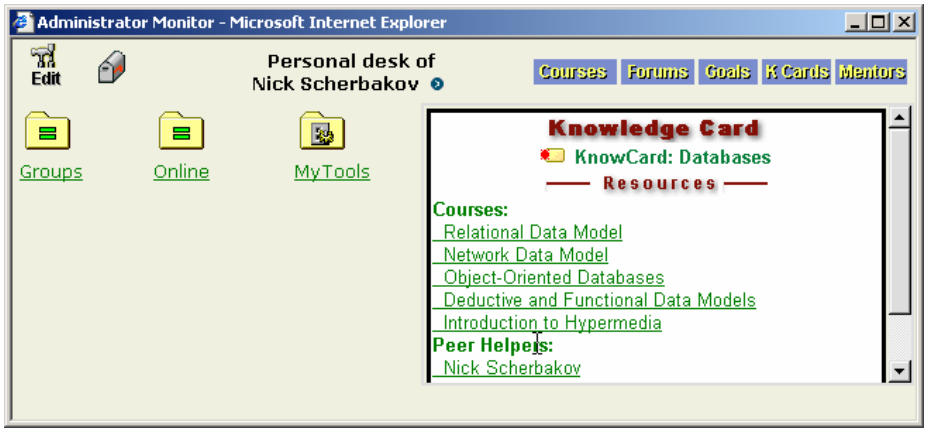

Figure 9: Accessing a knowledge card from a personal desktop folder

If a new training object is associated with this Knowledge Card, the system automatically captures that change by inferring the newly added training object the next time we access the personal desktop folder. Thus, this mechanism is a rather robust one to changes in the underlying repository of training objects.

On the other hand, by adding a knowledge domain to a personal desktop folder users are provided with possibility of browsing semantic overviews of training objects residing in the system. Again, if a new training object is added to such an overview, the system automatically updates the overview the next time when we access it.

Hence, by incorporating these two simple knowledge-processing mechanisms into the concept of personal desktop we were able to enhance Personal Desktop from a general user interface solution to a simple Knowledge Delivery tool.

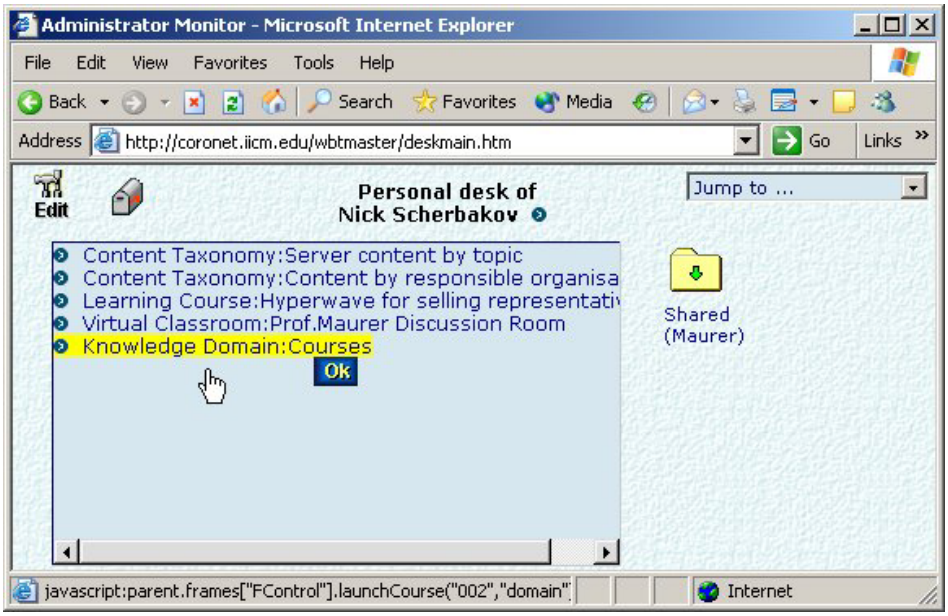

Figure 10: Accessing a knowledge domain from a personal desktop folder

7. Conclusion

Generally, first experiments with the Personal Desktop system demonstrate a rather good functionality and acceptance by users.

Students like the situation where they are not confronted with too much system tools or irrelevant training objects but rather with just few important tools and training objects.

On the other hand, tutors and authors very much appreciate the fact that they were able to completely control their training sessions and especially corresponding training strategy. They can be sure that their students are confronted with only those tools that are the essential part of the implemented training strategy.

Moreover, the extension of the concept of a simple user interface tool to a Knowledge Delivery tool leads to the conclusion that students are not only confronted with the relevant system tools but also with the most relevant and up-to-date training objects that are needed for a successful training sessions. Of course, through the concept of

Knowledge Domains students' work is supported by means of conceptual overviews of training objects required for that particular training session.

References

Andrews K., Nedoumov A., and Scherbakov N. (1995): *Embedding Courseware Into Internet: Problems and Solutions*, Proceedings of ED-MEDIA'95, pp.69-74. 1995

Borgida A. (1987): *Conceptual Modeling of Information Systems*, On Knowledge Base Management Systems, Springer-Verlag, M. L. Brodie, J. Mylopoulos (Ed.), 1987.

Borgida A. (1991): *Knowledge Representation, Semantic Modeling: Similarities and Differences*, E-R Approach '91, Elsevier Publishing, 1991.

Brodie M., Mylopoulos J., Schmidt J. (Ed.) (1984): *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer Verlag, New York, NY, 1984.

Burke R., Hammond K., and Young B. (1996): *Knowledge-based navigation of complex information spaces*, Proceedings of the Thirteenth National Conference on Artificial Intelligence (pp. 462--468). Cambridge, MA: AAAI Press/MIT Press. 1996

Calvanese D., De Giacomo G., Lenzerini M., Nardi D., Rosati R. (1998): *Knowledge representation approach to information integration*, Proceedings of AAAI Workshop on AI and Information Integration, 58-65, AAAI Press/The MIT Press, 1998.

Dietinger T., Maurer H., (1997).: *How Modern WWW Systems Support Teaching and Learning*, ICCE 97 (Ed. Z. Halim, T. Ottmann, Z. Razak), Kuching, Sarawak Malaysia, December 2-6, 1997, pp 37–51

Dietinger T., Maurer H. (1998): *GENTLE - (GEneral Networked Training and Learning Environment);* Proceedings of ED-MEDIA'98, AACE, Charlottesville, VA. 1998

Dietinger T., Gütl C., Pivec M., Maurer H., (1999a) *: Targeted Information Retrieval*, Proc. ICCE'99, IOS Press, Amsterdam, vol. II, 355-358, 1999.

Dietinger T., Gütl C., Pivec, M. (1999b): *Meeting the needs of the collaborative information society through targeted information retrieval*, Proc. of the Intern. Multi-conference Information Society IS'99, Ljubljana, Slovenia (ISBN 961-6303-18-X), 1999.

Gaines B.R., Norrie D.H. and Shaw M. (1996): *Foundations for the Learning Web*. ED-MEDIA'96: World Conference on Educational Multimedia and Hypermedia. 1996

Gaines B.R. and Shaw M.: *Institutional Transformations to a Learning We*b (1997), in: Proceedings of ED-Media/ED-Telecom, the World Conference on Educational Multimedia/Hypermedia, T. Müldner and T.C. Reeves (Eds.), 1997.

Helic D., Maurer H. and Scerbakov N. (2000)*: Web Based Training: What Do We Expect from the System.* Proceedings of ICCE 2000, Taiwan 2000, 1689-1694.

Helic D., Maurer, H. and Scerbakov, N. (2001a): *Accessing Best-Match Learning Resources in WBT Environment*, Proceedings of ED-MEDIA 2001, June 2001

Helic D., Lennon J., Maurer H. and Scerbakov, N. (2001b): *Aspects of a Modern WBT System*, Proc. Int. Conf. On Advances in Infrastructure for Electronic Business, Education, Science and Medicine on the Internet, SSGRR 2001, CD-Rom Publication (ISBN: 88-85280-61-7), Paper 38 (August 2001)

Helic D., Maurer H., Scerbakov N. (2001c): *Knowledge Domains: A Global Structuring Mechanism for Learning Resources in WBT Systems*, Proceedings WebNet 2001, AACE, Charlottesville, USA 2001, 509-514.

Helic D., Maurer H., Scerbakov N. (2001d): *Mentoring Sessions: Increasing the Influence of Tutors on the Learning Process in WBT Systems*, Proceedings WebNet 2001, AACE, Charlottesville, USA 2001, 515-519.

Helic D., Maurer H., Scerbakov N. (2002): *Aspects of Collaborative Authoring in WBT Systems*, Proc. Int. Conf. on Advances in Infrastructure for Electronic Business, Education, Science and Medicine on the Internet, SSGRR 2002, CD-Rom Publication (ISBN: 88-85280-62-5), Paper 37 (January 2002)

Lambiotte J.G., Dansereau D.F., Cross D.R. and Reynolds S.B. (1984). *Multirelational Semantic Maps.* Educational Psychology Review 1(4): 331-367. 1984

Minsky M. (1968): *Semantic Information Processing*, Cambridge, MA: MIT Press, 1968.

Nosek J.T. and Roth I. (1990): *A Comparison of Formal Knowledge Representation Schemes as Communication Tools: Predicate Logic vs. Semantic Network*. International Journal of Man-Machine Studies 33: 227-239. 1990

Shaw M. and Gaines B.R. (1996): *Experience with the Learning Web*. Proceedings of ED-MEDIA'96: World Conference on Educational Multimedia and Hypermedia. 1996

Sowa J.F. (1984): *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.

Sowa J.F. (Ed.) (1991): *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo, USA, Morgan-Kaufman, 1991.