Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW

TANGOW: Task-based Adaptive learNer Guidance On the WWW

Rosa María Carro, Estrella Pulido, Pilar Rodríguez
Escuela Técnica Superior de Ingeniería Informática,
Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid, Spain
e-mail: {Rosa.Carro, Estrella.Pulido, Pilar.Rodriguez}@ii.uam.es

Abstract: This paper presents TANGOW, a new tool for developing Internet-based courses, accessible through any standard WWW browser. Courses are structured by means of Teaching Tasks and Rules which are stored in a database and are the basis of TANGOW guidance ability. In TANGOW a Student Process is launched for each student connected to the system. Each Student Process consists of two main modules: a Task Manager that guides the students in their learning process, and a Page Generator that generates the HTML pages presented to the student. The Student Process also maintains information about the actions performed by the student when interacting with the course in the Dynamic Workspace. This information is used by TANGOW to adapt the course contents to the student's learning progress. TANGOW has also information about student profiles, which is used to select, at run-time, the contents of each HTML page presented. TANGOW stands for Task-based Adaptive learNer Guidance On the WWW.

Keywords: Adaptive systems, Web-based training, Intelligent-tutoring systems, Dynamic course generation, Educational multimedia

1. Motivation

The fact of being distributed and available from distant locations all around the world makes the WWW (World Wide Web) the best technology for distributing information. The WWW is extensively used for teaching and learning, and allows students to access almost any kind of information they are looking for [1] [2]. However, the process of learning is more complex than navigating between different pages and reading what is written on them. One of the most important problems in this context is the determination of suitable navigation paths in the hypermedia space that help students to better achieve their learning goals. While it has been argued for some time that this problem can be addressed by a good design of the navigation space as part of the course design process, the need of more sophisticated mechanisms that modify the navigation alternatives by some sort of adaptation procedure is becoming more widely accepted. Not all students have the same skills for learning a concrete subject. Some of them need more explanations than others, and they may have previous knowledge about the subject or not. These are important differences among students, and there are others related to personal features such as age, interests, preferences, etc., which are important too [3].

In this paper, we present TANGOW, a new tool for developing Internet-based courses, which facilitates the construction of adaptive learning environments for the Web, and guides the students during their learning process by registering their behaviour and profile. Curriculum sequencing is generated dynamically, so that the same concepts may be taught in different ways, depending on student's profile and his/her actions while interacting with the course. By changing the learning strategy, even the same student may study the same concept in different ways.

A course is described in terms of Teaching Tasks (TT) and rules [4]. A TT is the basic unit that appears in the learning process, and may be atomic or composed. Knowledge is represented by means of TTs that need to be achieved. TTs may be theoretical, practical or a set of examples. In addition, a TT may have a list of media elements (text, images, videos, applets, sounds, animations, ...) associated. A description language which specifies the relative positions of these media elements is used to construct the HTML pages that will be presented to the students. The specific elements included in these pages will depend on information about student profile and his/her learning progress, and are selected 'on the fly'. Figure 1 shows the description of a task which consists of five different slots: the task type ("T" for a theoretical task), atomicity ("Y" for an atomic task), description (a text describing the task), the name and associated parameters of a method which is used to decide whether the task is finished ("F_TEO" and "pag_visited/tot_pag"), and a list of associated media elements (the contents of the HTML slot).

TYPE=	T
ATOMIC=	Y
DESCRIPTION=	Description of circular signs
END_METH=	F_TEO
PARAMS=	pag_visited tot_pag
HTML=	CIRCULARES STOP C_PROHI E_PROHI EP_VEHI EPV_SIDE

Figure 1: Description of a teaching task

In TANGOW, a rule describes how a TT is divided into subtasks. There may be several rules for the same TT, each of them representing a specific way of decomposing the TT into subtasks. It may be necessary to perform all these subtasks following a fixed order (AND sequencing), in any order (ANY sequencing), or it may be enough to perform only some of them (OR/XOR sequencing). In addition, a rule specifies the requirements for it to be applicable, which may depend on information about the tasks already achieved, the student's profile and the learning strategy in use. Figure 2 shows the description of a rule which consists of six different slots: the type of task decomposition ("AND" sequencing in this case), the left-hand-side of the rule ("Priority" task), the tasks included in the right-hand-side of the rule ("Circumstantial Signs Theory and Exercises"), activation condition for the rule and its associated parameters ("c-4" method and "exer_ok from "Vertical_Signs" task), and a description of how parameters of the task in the left-hand-side of the rule are calculated in terms of parameters of the tasks in the right-hand-side of the rule (the contents of the "CALC_PARAMS" slot).

SEQUENCING=	AND
LHS=	Priority
RHS=	Circumstantial_Signs_Theory Circumstantial_Signs_Exercises
ACT_CONDITION=	c-4
PARAM=	exer_ok Vertical_Signs
CALC_PARAMS=	time_in msum2 time_in Circumstantial_Signs_Theory time_in Circumstantial_Signs_Exercises tot_pag mdirect tot_pag Circumstantial_Signs_Theory exer_ok mdirect exer_ok Circumstantial_Signs_Exercises exer_done mdirect exer_done Circumstantial_Signs_Exercises tot_exer mdirect tot_exer Circumstantial_Signs_Exercises pag_visited mdirect pag_visited Circumstantial_Signs_Theory

Figure 2: Description of a rule (full size)

The paper is organised as follows. Section 2 describes the architecture of the TANGOW system. The guiding process performed by TANGOW during a learning session is described in Section 3. Section 4 presents a review of existing teaching systems and analyses how they compare with TANGOW. Finally, in section 5, we summarize the results presented in the paper and mention some further extensions currently being researched.

2. The TANGOW system

The architecture for the TANGOW system is based on the standard Web paradigm, where the server receives requests from students through their WWW browsers. There is a process, the *Task Manager,* for each student connected to the system which takes control of the student learning process during the whole session. If the same student is following more than one course, there will be a *Task Manager* for each of them. All these processes are controlled by a *Process Manager* which receives information about student actions from a CGI program running on the Web server. Communication between processes is done through sockets for distribution purposes.

The *Process Manager* is always running waiting for requests. When one of these requests is received, its parameters are analysed and sent to the corresponding *Task Manager*, which is previously launched if it is not already running. The *Task Manager* stores information about student actions in the *Dynamic Workspace* and sends the relevant information to the *Page Generator* which generates the HTML pages dynamically and sends them back to the student through the CGI program.

All these modules use information stored in the following databases:

Users DB

It contains data about student's profiles and their actions during the learning process. A student profile includes personal information such as his/her age, selected language and preferences with respect to the learning strategy. At the end of a session the corresponding *dynamic tree* (part of the *Dynamic Workspace*) is stored in the *Users DB*.

Course Content DB

It contains all media elements (texts, images, videos, animations, simulations, applets, ...) that will appear in the HTML pages. They are classified according to features defining the student profile (i.e. language, age, ...).

Teaching Tasks Repository

It contains a general description of all the teaching tasks that have been defined by the course designer. This description includes general task attributes such as its name, description, content type (theoretical, practical or example task), composition type (atomic or composed), finalisation requirements (a function which decides, at runtime, whether tasks have been completed), and an optional list of media elements used for page generation.

All the above mentioned components of the system are illustrated in figure 3, where dotted arrows represent information flow and solid ones represent inter-process communication. The white arrow represents a function call.

2.1. The CGI program

The CGI program checks the parameters received from the client browser and sends them to the *Process Manager*. If the request corresponds to the beginning of a session, data related to the client location are sent along with these. Finally, the CGI program waits for a response from the *Page Generator* on an opened socket and sends the generated HTML pages to the student. An example of this process applies to the WWW Browser X in figure 3. Note that requests for static HTML pages do not go through the CGI program (see WWW Browser Y as depicted in figure 3).

2.2. The Process Manager

The *Process Manager* runs as a server, waiting for requests from the CGI program on an opened socket. If the request parameters correspond to the beginning of a session, the *Process Manager* launches a new *Task Manager* and establishes communication with it through a new socket. In addition, it assigns a new identification number to the opened session and stores it, along with the port in which the new process is running, in the *open sockets* table. If the request identifies a session already started, the *Process Manager* obtains the communication port identifier with the corresponding *Task Manager* from the *open sockets* table. Finally, the *Process Manager* sends the request parameters to the *Task Manager*.

2.3. The Task Manager

As already mentioned, the *Task Manager* provides a guidance to students in their learning process by deciding the next set of achievable tasks that will be offered to the student. The elements in this set depend on the active learning strategy, the student personal data and student actions previously performed.This information is transferred to the *Task Manager* as parameter values in the submitted requests. Furthermore, the *Task Manager* stores information about the actions performed by the student and their results (the number of pages visited, the number of exercises done, the number of exercises successfully solved, etc) in a *dynamic tree.* In this tree, nodes correspond to tasks achieved by the student while edges represent how tasks have been decomposed during the learning process. Finally, the *Task Manager* provides the *Page Generator* with the parameters that will be needed during dynamic page generation. This will be explained in detail in Section 3.

2.4. The Page Generator

The *Page Generator* receives useful parameters associated to the active task for generating the HTML pages dynamically. These parameters are related to the student profile and the student actions. Based on this information the *Page Generator* decides which type of media elements (i.e. texts, images, videos, animations, simulations, applets, ...) will appear in the HTML document and how they will be laid out. Information about the student profile will be used to select specific media elements according to features such as its content difficulty, language in which they are written, etc. Once the HTML page is generated, it is sent back to the student through the CGI program.

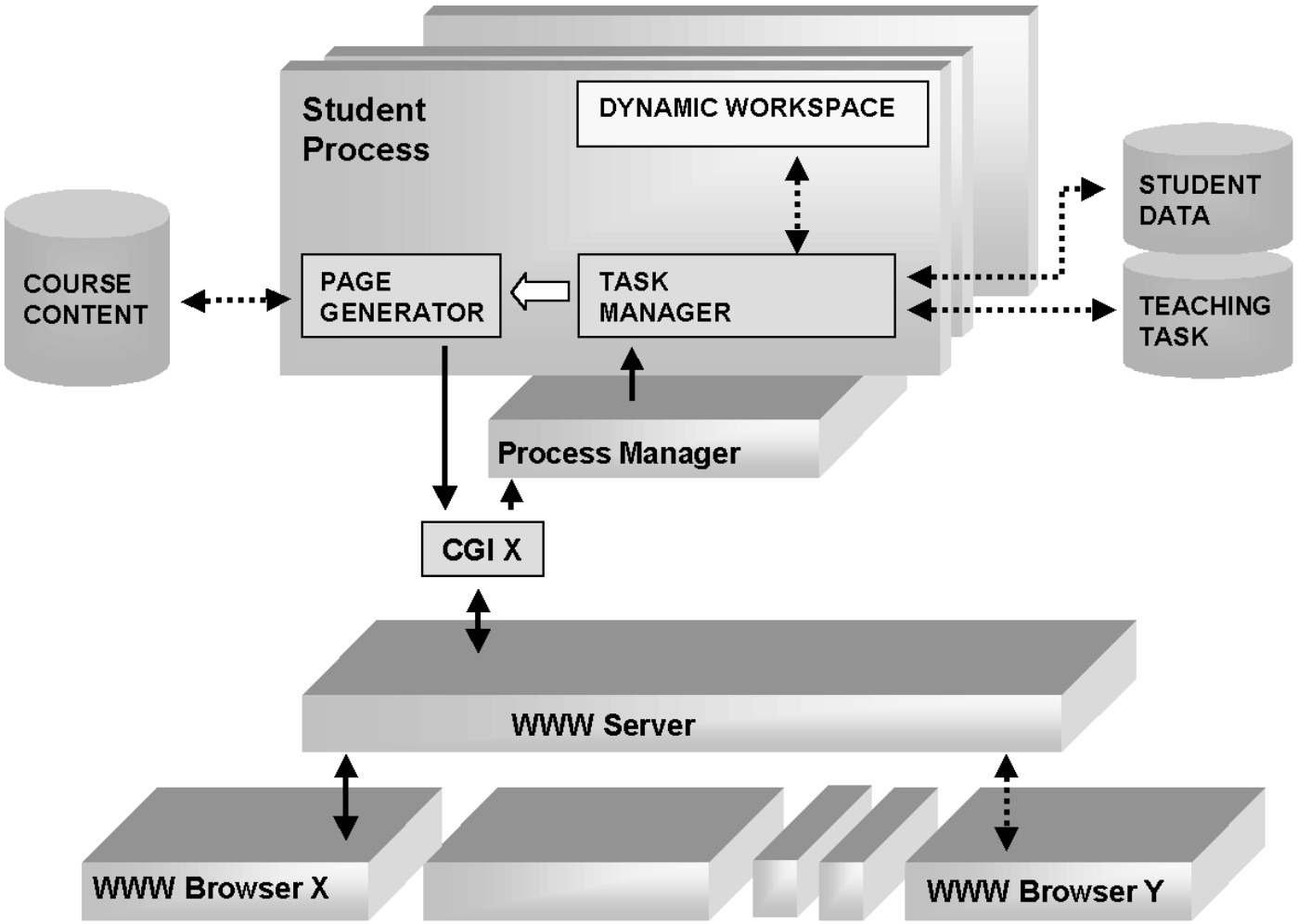

Figure 3: The architecture of the TANGOW system.

3. Adaptive guidance in TANGOW

When a student enters TANGOW, (s)he is asked to select the course (s)he wants to take, and to identify him/herself by writing his/her name and password. Moreover, if it is the first time that the student enters the system, the student is asked for personal data such as his/her age, and preferred language and learning strategy. This information is stored in the *Users DB*.

If the student has accessed the system before, the *Task Manager* uses the information stored about the student's previous actions related to the selected course (if any) to ask the *Page Generator* to provide the student with the suitable page of the course. If there is no such information available, the initial page of the course will be presented.

At every step of the learning session, the *Task Manager* constructs a list of *achievable tasks* that can be presented to the student by selecting subtasks that (1) belong to the set of subtasks in which the current

task is decomposed in the *dynamic tree*, (2) have not been achieved yet, and (3) are *actionable*. A task is *actionable* if there exists at least a rule defining it whose activation condition is satisfied. This list will be presented to the student as a menu page. If the list of achievable tasks contains a single element, this will be directly presented. Each link in the menu page includes information about the task to be tackled and the rule which describes it. When the student clicks on a link, the *Task Manager* checks this information, and looks for the description of the task in the *Teaching Tasks Repository*. It builds the *dynamic tree* node associated to the task and sends relevant information to the *Page Generator*, which will create the next page to be presented to the student.

Figure 4 shows an example of two different lists of achievable tasks: the one on the left has the "Vertical signs" task activated. This task has no prerrequisites, but "Circumstantial signs" has a precondition which makes the task actionable only if a minimum number of exercises about "Vertical signs" have been well done (see figure 2). As the student has not completed the "Vertical signs" task yet, the link corresponding to "Circumstantial signs" task is not activated. The list on the right includes the "Circumstantial signs" task link because the rule describing this task has its prequisites satisfied. In other words, the student has done the minimum required number of exercises related to "Vertical signs". Note that the "Vertical Signs" link now is not active. That is because the associated task has already been achieved. These pages correspond to a course on driving which can be found at http://helena.ii.uam.es/html/courses.html.

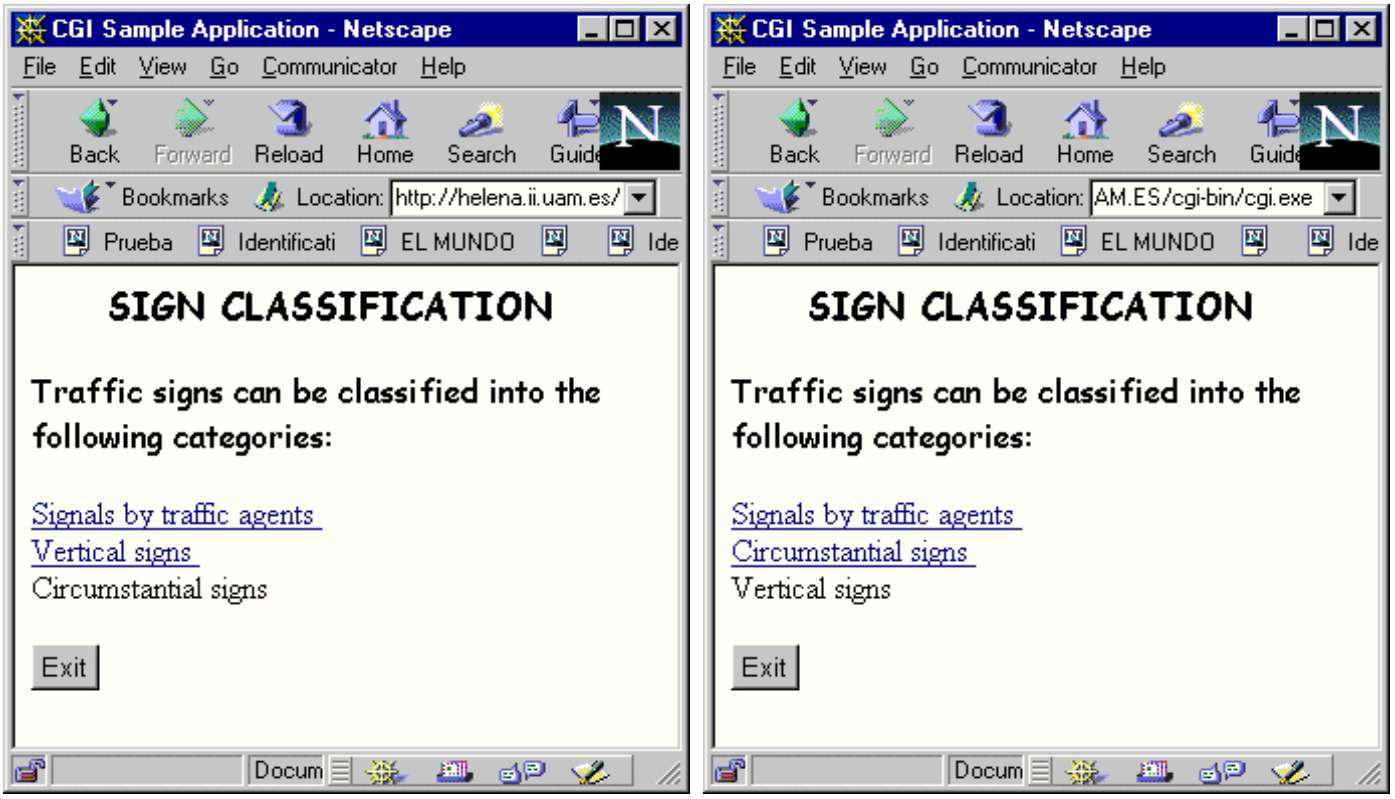

Figure 4: An example of two different lists of achievable tasks

In addition, every time a task is finished, the *Task Manager* checks its execution results (i.e. the number of exercises done, the number of exercises well done, the number of pages visited, etc.) as stored in the *dynamic tree* and calculates a success value for the task. The task's execution results are then propagated up in the *dynamic tree* to be used in the future for student guiding purposes. Then, the *Task Manager* checks if the finalisation requirements of the ancestor task are satisfied. If this is the case, the student does not need to perform any other subtasks and the ancestor task ends automatically. The *Task Manager* goes up the *dynamic tree* checking the ancestors finalisation requirements until a task that has not been completed yet is found. Finally, it constructs a list of achievable tasks following the process described above. By using the *dynamic tree* only a subset of the existing teaching tasks has to be examined in order to construct the list of achievable tasks. This makes the curriculum sequencing process very effective.

The student may navigate between pages related to the same task that are generated 'on the fly'. In the case of practical exercises, the answer given by the student (by clicking on a link, selecting an image, or filling a questionnaire) is stored by the *Task Manager*, and the next exercise is presented.

If the student wishes to leave the system, the *Process Manager* closes the connection with the corresponding *Task Manager* process. Before exiting, the *Task Manager* asks the *Page Generator* to display the results achieved by the student so far. It also stores the sequence of tasks performed and the results of their execution in the *Users DB*. The stored results will be used to reconstruct the personal *dynamic tree* in future sessions. From the student point of view, the immediate effect of this reconstruction process will be that (s)he will be presented with the last page visited.

TANGOW identifies each student transaction by means of a hidden variable that is associated to a specific student session. The *Process Manager* assigns value to that variable, which is included in every dynamic page presented to the student. It is sent back to the system with every student request.

With respect to adaptivity, TANGOW makes use of three different mechanisms. The first one is task decomposition, which can be different depending on data about the student profile. For example, a task may be decomposed in two different ways: one decomposition may include subtasks related to theory and exercises and the other can contain an additional subtask with examples about the subject. Different criteria would be established to choose, at runtime, between both decompositions.

The second way of implementing adaptivity is by including in the rules preconditions related to other tasks achievement, so that a rule is not activated unless certain results have been achieved when executing a specific set of tasks (this case can be seen in figure 2). In this way, courses are somehow "customized", presenting different students with different tasks depending on their previous actions.

Adaptivity is also present in the process of building the HTML pages displayed to the student by using different multimedia elements depending on the student profile. At the moment, the media elements that appear in the pages are selected depending on the student language and age. Texts and sounds will be different depending on the student language. Moreover, the same concept can be explained in different ways depending on the student age by using words of different difficulty. The same applies to graphics, animations and videos, which may vary depending on student features.

These are the main ways of adapting the courses to each student, and depend on decisions taken by the course designer when describing the course in terms of tasks and rules. But they are not the only ones. Teaching strategies also have an influence on the order in which subtasks are performed. At the moment, there are two predefined strategies: "theory presentation first" and "practical exercises first". The student can select his/her preferred strategy at the beginning of a session. If the "theory presentation first" strategy is selected, the student will be presented with theoretical tasks first, and only after they have been performed, (s)he will be able to tackle examples and practical tasks. The reverse order will be used if the "practical exercises first" strategy is selected. In the future, more teaching strategies will be implemented, and it will be possible to change the active strategy during the learning session in order to improve the guidance received by the students.

4. Related work

Of special interest in the field of intellingent tutoring systems is the work developed by the ELM group in recent years [5] [6].One of their implementations is ELM-ART II [7], an adaptive tutoring system on the Web that supports learning programming in LISP which received the European Academic Software Award in 1998. Adaptivity in ELM-ART II is implemented by selecting the next best step in the curriculum on demand. Links in HTML pages are annotated according to a traffic lights metaphor, where different colors are used to indicate, among other things, that a section is ready to be learned and recommended, ready but not recommended or not ready to be learned yet. This annotation process is performed whenever a learning unit is finished by reviewing all concepts that are prerequisites to this unit. This differs from TANGOW, where the *dynamic tree* is used to restrict the set of teaching tasks that need

to be reviewed whenever a task is finished.

As for the process of dynamic page generation, the AHA system [1] can be mentioned where filters for content fragments are encoded by means of conditional sentences that are included as comments in HTML pages. The main difference with our approach is that we create the HTML pages by linking media elements, whereas in the AHA system the pages are already created and it is decided, at runtime, which portions of them are shown to the student. In other systems [8], pages related to teaching materials are generated by formalising the structure of the documents using SGML and by asking the designer to specify the concrete contents for this general structure. In TANGOW there is no need of defining different page structures, because each page is composed 'on the fly' by choosing from the media elements associated to the active task those that will appear on the HTML pages just before building them.

There also exist different approaches to structure course contents. In [2], the structure of the domain is based on concepts which are linked to documents and to other concepts through typed and weighted links. The student is guided towards appropriate documents based on information about his/her knowledge of each concept. Other formalisms such as the prerequisite graph model [3] partition course contents into cells, where a cell is a single small topic to be studied. Relationships between cells are established by means of a prerequisite graph which states what topics should be known before studying further. This formalism differs from ours in that cells correspond to static documents and the relationships established by the prerequisite graph are also static, whereas in TANGOW the documents associated with teaching tasks are generated dynamically and the prerequisites for a Teaching Task may vary depending on the student’s profile, his/her previous actions, and the teaching strategy in use.

Another Internet-based teaching system which is currently in the design phase is SKILL [9]. In a way similar to the prerequisite graph model described above, in SKILL the course material is organised according to their prerrequisites. Adaptivity is implemented by taking into account the student's previous knowledge of the subject. This is done in TANGOW too but, in addition, student features such as his/her age or preferred language also determine which documents are presented to him/her. As TANGOW, SKILL stores the results of the learning session which are used in future sessions. One of the main differences between the SKILL architecture and that of other existing systems is that it incorporates an annotation facility suited for collaboration work.

A different approach is followed in the DCG system [11][12] where the concept structure of a course is represented as a road-map which is used to generate a plan for the course. The planner searches for sub-graphs that connect the concepts known by the learner with the goal-concept, and changes the plan if the student is not able to achieve a result higher than a given threshold score for a given concept. The main difference between DCG and TANGOW is that in the former, adaptivity is implemented by modifying the plan to achieve the goal-concept, while TANGOW adapts the course contents to the student's learning progress by changing the set of possible subtasks at every learning step.

Finally, it is worth mentioning one of the main problems which arise when developing Web courses: the designers' lack of time and experience. In the Benchmarks Project [13], several well-known tools for the creation of physical Web elements, the communication and interaction among class participants and the administration of learning environments are evaluated, analysing their advantages and disadvantages. Multi-purpose tools are proposed as good specific applications for developing on-line courses, taking into account that they are designed to work together. TANGOW could well be considered one of such tools since it can be used for developing Web courses whose organisation is described by the course designer in terms of tasks and rules. This makes the course organisation independent of the media elements the course contents is made up of. We think this independece will greatly help designers not only with the development of media materials but with course description too.

5. Conclusions and future work

The TANGOW system allows course designers to develop adaptive learning environments for the WWW by using tasks and rules to describe the courses. These tasks and rules are used at execution time to guide

the students during their learning process, so that they will be presented with different HTML pages depending on their profile, their previous actions, and the active learning strategy. HTML pages are generated dynamically from general information about the type of media elements associated to each task and their layout. The concrete media elements that appear in these pages are selected 'on the fly', depending on student's characteristics such as his/her age and language.

TANGOW is written in Java and is widely accessible through Internet by using any standard Web browser. Thanks to the storage of tasks, rules and multimedia materials in databases, the cost of course maintainance is low since designers may change, add or remove course components easily. The use of databases also allows the reuse of components in different courses.

Currently we are working on the improvement of the guidance process by providing the system with the possibility of changing the active learning strategy dynamically, depending on the student's results. Furthermore, we are planning a course designer oriented interface for making the course development process easier.

In the near future, we intend to design new courses and to analyse the effectiveness of TANGOW guidance in the students learning process by making a real evaluation with students. TANGOW architecture is suitable for supporting collaborative work among system users by establishing communication between student processes.This is another research line that we are considering.

Acknowledgements

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL97-0306.

References

[1] P. de Bra, L. Calvi, "AHA: A Generic Adaptive Hypermedia System", Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia at the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA, pp. 5-11. June 20-24, 1998. http://wwwis.win.tue.nl/ah98/DeBra.html

[2] D. Pilar da Silva, R. Van Durm, E. Duval, H. Olivè, "Concepts and documents for adaptive educational hypermedia: a model and a prototype", Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia at the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA, pp. 35-43. June 20-24, 1998. http://wwwis.win.tue.nl/ah98/Pilar/Pilar.html

[3] O. Nykänen, "User Modeling in WWW with Prerequisite Graph Model", Proceedings of the workshop "Adaptive Systems and User Modeling on the World Wide Web", Sixth International Conference on User Modeling, Chia Laguna, Sardinia, 2-5 June 1997. http://fit.gmd.de/UM97/Nykanen.html

[4] R.M. Carro, R. Moriyón, E. Pulido, P. Rodríguez, "Teaching Tasks in an Adaptive Learning Environment", 8th International Conference on Human Computer Interaction, HCI International '99, Munich, August, 1998, forthcoming.

[5] P. Brusilovsky, E. Schwarz, G. Weber, "A tool for Developing Adaptive Electronic Textbooks on WWW", Proceedings of WebNet'96 - World Conference of the Web Society, San Francisco, CA, AACE, pp. 64-69. October 16-19, 1996.

[6] P. Brusilovsky, J. Anderson, "ACT-R electronic bookshelf: An adaptive system for learning cognitive psychology on the Web", Proceedings of WebNet 98 World Conference of the WWW, Internet & Intranet, pp. 92-97, Orlando, Florida, November 7-12, 1998.

[7] G. Weber, M. Specht, "User modeling and adaptive navigation support in WWW-based tutoring systems", Proceedings of User Modeling '97, pp. 289-300. Italy, June, 1997.

[8] M. da GraÇa, J. Benedito, R. Pontin, "Tools for Authoring and Presenting Structured Teaching Material in the WWW", Proceedings of WebNet 98 World Conference of the WWW, Internet & Intranet, pp. 194-199. Orlando, Florida, November 7-12, 1998.

[9] G. Neumann, J. Zirvas, "SKILL: A Scalable Internet-Based Teaching and Learning System", Proceedings of WebNet 98 World Conference of the WWW, Internet & Intranet, pp. 688-693. Orlando, Florida, November 7-12, 1998.

[10] J. Vassileva, "DCG + WWW: Dynamic Courseware Generation on the WWW", Proceedings of AIED'97, Kobe, Japan, August 18-22, pp. 498-505, 1997.

[11] J. Vassileva, "A Task-Centred Approach for User Modeling in a Hypermedia Office Documentation System", in Brusilovsky, P., Kobsa, A. and Vassileva J. (Eds.) Adaptive Hypertext and Hypermedia, Kluwer Academic Publ. Dordrecht, Chapter 8, pp. 209-247, 1998.

[12] A. Brown, L. Hansen, "Software Tools for Distance Learning: The Benchmarks Project", Proceedings of Society for Information Technology and Teacher Education SITE'98 , Washington DC, March 10-14, 1998.