A Computer-Based Learning tool using XML to enable learning styles

Aleardo Manacero Jr. and Alisson G. Casagrande and Odnei C. Lopes
Dept. of Computer Science and Statistics, Unesp
São José do Rio Preto, Brazil - 15054-000
aleardo@dcce.ibilce.unesp.br

Abstract— **This paper describes a computer-based learning tool that uses the Kolb learning styles inventory to distinguish students while presenting the contents of a undergraduate course. This tool overcomes the problems related to the volume of information offered to the student and also the replication of informations oriented to different learning styles. It uses the XML (eXtended Markup Language) in order to provide an optimized information structure and a good environment for online, internet based, application. This approach was applied in the build on a tool aimed to help on the learning of real-time systems by computer science students. Results from this application are also presented.**

I. Introduction

Use of computer-based tools to aid the teaching and learning processes is a common technique, found at many institutions and courses ([6], [9], [10]). It has been used on a wide variety of applications, including virtual laboratories and online (distance learning) courses, ranging from pre-school up to graduate levels. The importance of such tools cannot be neglected, giving them a great relevance for the educational process.

However, most of the work done in this area is strictly conventional, that is, the contents are linear condensations of material already found in books. Exceptions to this approach are concentrated on courses aimed to young children, where the contents are usually presented in a game-oriented structure. At undergraduate level these tools are a collection of pieces of course's content and tests ([1], [3]), with few attempts on the game-oriented structure [8].

From this conventional characteristic come two other problems with computer-based tools: the volume of data that is presented and the lack of personal styles on the presentation. Both problems are avoidable during the design of a conventional tool, but their solution imposes different constraints that lead to other problems.

The volume of data can be reduced at expense of eventually needed information. Since this is often undesirable, most of the tools are built with overloading information, leaving the student with the filter task. On the other hand, the adoption of learning styles on conventional models is very hard since all the material has now to be produced in as many styles as the inventory defines. For the Kolb inventory, for example, the course material has to be presented in four distinct ways, what significantly increases the work to be performed by the instructor on its implementation.

With this work we propose a XML (eXtended Markup Language) approach to implement computer-based learning tools that use learning styles to improve the performance of the students, reducing the volume of data that is presented (and stored) and presenting the contents in a more fashioned way.

The next section discuss some of the problems related with computer-based learning tools. The methodology of the XML approach is presented on section 4. Results about the application of this approach to a real-time systems computer-based learning tool are shown and discussed during the last sections.

II. Problems on computer-based learning

The conventional structure for material presentation has a clear advantage against non-conventional ones: the speed on which the course can be assembled. This speed is a natural consequence since most courses are simple copies of material already in use on classroom courses. It has, however, a number of drawbacks:

1. it overloads the students with information;
2. it does not correctly adapt the contents to the computer media;
3. it does not take into account the differences on learning styles.

The first two problems lead to boring material, either by its extension or by its book-like style. The tools usually take full control of the learning process, dictating what has to be done and how it is done. Since most of the students like to take control over these actions, some improvements have been attempt in order to provide control to the student. However, most of the solutions strongly rely on hypertexts (texts links spread over a conventional text), what could turn things even worse since the student can be caught inside a knot of links.

Several solutions have been provided, during the past few years, aiming the convenient adaptation of a course content to the computer media. This includes computer

games, animated illustrations, voice, discussion rooms (chats), customization of interfaces, and so on. Although they actually improve the tools, the still lack of a more oriented communication with the student, who has preferences on how he/she learns better from distinct forms of material assembly. This is, in fact, the third drawback just presented.

Differences on learning styles have been neglected by most of computer-based tools. Some of these tools actually attempt to deal with this issue, but usually their solutions on customization are restricted to how the tool looks, that is, its colors, luminance, size of characters, etc. Those parameters, indeed, have implications on how the student learns [4], but they are only a small part of their preferences. Curry shows on [2] that the learning style is built on several levels (environmental, interaction, information processing and personality preferences). The interface customization attacks only the environmental preferences, leaving the other levels unattended.

From the levels described by Curry, a very important one is the one that deals with information processing preferences, which usually is not present on computer-based tools. The task of information processing is rather important, since the actual understanding of a given content depends on how deep that information could be processed. Every student has personal preferences about the way the better process information. Some prefer to take a more active role, others prefer to receive it passively. Others prefer concrete experiences while some like abstractions. If the content is presented in the form the students have their best performance on information processing, then the content will be soundly understood. If not, they will have to spent more time and effort to achieve similar results.

One inventory that deals with the information processing level is the Kolb Learning Styles Inventory [5]. It defines four different styles (described on the next section) by the composition among concrete/abstract and active/reflective preferences.

In order to implement a tool that uses the Kolb inventory, or any other inventory by the way, one has to provide the information on every different style. Although that is feasible, it largely increases the work that has to be done by the instructors in order to make such course available. In fact, if the course designer does not take the needed care, the material could increase the students difficulties by the lack of a clear separation between styles.

Therefore, besides this area has received a lot of attention during the past years, producing several interesting tools, it still avoids the use of strong learning styles by the implicit difficulties imposed by them. One of these difficulties is the level of abstractions involved with some of the learning models. Kolb's model for information processing modeling, besides its subtle characterizations on some aspects of learning styles, is highly suited for a computer-based tool. The model and the reasons that led to using it into this approach are described next.

III. The Kolb learning styles inventory

Kolb's model of learning styles defines four types of learners: assimilators, convergers, divergers and accommodators. Each of these styles is described by a defining question and by a set of characteristics related to how the person receives and process the information. The determination of what style fulfills the person's profile is done by a set of twelve questions about preferences on how, when and what study and learn.

The answers for those questions area accounted and define a pair of main profiles, related to the preferences on abstract or concrete experience, and on active or reflective posture. The combinations of these main profiles lead to the four styles defined in the inventory. A short description of these styles is given next.

- **Diverger (type 1)**
  Their defining question is "Why". Students of this type prefer concrete experiences based on their feelings. They like group interactions, working well in brainstorming sessions. They are called divergers because they can see things from different perspectives. Since they like working in groups, they also want a close interaction with the instructors.

- **Assimilator (type 2)**
  Their typical question is "What". They succeed when the information come in a logical and organized fashion. Students of this type get information from abstract conceptualization and process it through reflective observation. This is the profile of a conventional student, who performs well in traditional environments. They are called assimilators because they use pieces of data, that are assembled in order to be assimilated. They prefer individual work and see the instructor as an expert, who is an authority on that field.

- **Converger (type 3)**
  They have "How" as their typical question. They like active experimentation in environments that enable them to try and fail safely. The information is collected by abstract conceptualization and, as active persons, they want to test them, watching how things work. They do not want to remain long periods in one activity (classes, reading, watching, etc.) and like doing things fast, going directly to the point, reason behind their title. To learn better they must go through examples and practices, avoiding too much theoretical work.

- **Accommodator (type 4)**
  Their defining question is "What if". They like

to apply the received information into new situations, "accommodating" it to their own needs (that is why their denomination). Information is captured through concrete experience, being processed by active experimentation. They are problem solvers, taking risks on their own. They like working with people, usually to act as a leader who will teach the fellows. Rules and procedures are obstacles to be beaten and the instructor should not be a supervisor of their work.

Most of these characteristics are easily achievable by a computer tool. As an example, it is quite obvious that a person who prefers abstract experiences, with conventional lectures, would be satisfied by a book-like online course, where he/she could read the material and watch some animations about experiences. More active postures would demand virtual exercises, which are reasonably easy to provide. Some other functionalities are also easy to maintain, like e-mail, newsgroups and chats.

Therefore, the use of Kolb's inventory in a computer-based learning tool is feasible and desirable. However, in order to use such styles one has to create conditions for all kinds of preferences. As said before, this would lead to a multiplication of informations to be inserted by the instructor. On the other hand, several characteristics are present in more than one style, what indicates the possibility of information reusability. On the next section it is described a tool implemented using Kolb's inventory, the RTtutor (**R**eal-**T**ime **tutor**), where the use of XML on its creation enabled the reuse of information by the course designer.

IV. RTtutor's design

The RTtutor is a tool aimed to help the learning of real-time systems by computer science students. Its creation follows a previous project named RTsim (**R**eal-**T**ime **sim**ulator)[7], which is a simulator of real-time scheduling algorithms that has been used as a laboratory tool to teach the scheduling algorithms defined for real-time systems. From the RTsim's use it became notorious that the tool must be expanded in order to approach other relevant topics of the undergraduate course.

One of such expansions led to the use of learning styles strategies to expose the course's contents. This tendency, summoned with the need for a more portable graphical interface, made obvious that the RTsim should be redesigned from scratch. The decision, although, was to keep two different tools, the already functional RTsim and a new, learning styles oriented, tool named RTtutor, which could make use of RTsim results when needed.

Therefore, before talk about RTtutor it is reasonable to describe the current developments on RTsim, which include its porting to Java, the implementation of a module that verifies schedules made by its user, and a stronger algorithm analysis. The port to Java will solve the performance problems introduced when its interfaces were rewritten in Java, since the time that will be lost on poor arithmetic execution is easily gained by the absence of several calls to the Java machine. The module for schedule verification actually is a demand by the RTtutor, since some styles require a more active posture by the students, who would be making the schedules by themselves, instead of just getting the results from RTsim. A better algorithm analysis is aimed to improve the potential use of RTsim by real-time systems designers, who would be able to get stronger analysis of their applications by RTsim's simulations.

Back to RTtutor, its design started from the definitions of two important specifications: the use of Kolb's inventory and its application on online learning. While the latter imposed the use of Java as its programming language, the former led to the adoption of XML as the course specification language. This paper is more concerned with the learning models and, therefore, will concentrate in the XML part, following a brief description of the Java client-server model.

A. The Java client-server structure

The Java structure divides the RTtutor into two modules, the server module, where all the management occurs, and the client module, where the user interacts with the system. The interactions between client and server occurs through message-passing calls executed by threads started on both sides of the communication channel.

On the server side, the main components are:

- **TalkWithClient**, which is in charge of keeping the communication alive. Messages would be requisitions from a client, answers from the server or warnings/commands issued by the server.
- **ServerUserData**, that performs the manipulation of users data (login name, password, track in the course and personal informations).
- **TutorServer**, which provides a graphical interface to server's control and configuration.
- **Cache**, that reduces the overloading of content translations by the storing of information that has already been translated during the current session.
- **XMLXSLManipulator**, that actually performs the translation of encoded XML information to HTML pages, using the XSLT library.
- **ContentTreeMngr**, which provides a tree-like structure for each learning style. This tree is presented by the client module in order to make the navigation process easier for the user.

The server is also responsible for issuing updates on all pages currently displayed by the clients in the case

the XML content gets modified by author intervention. Therefore, if the instructor wants to change some information, it gets automatically loaded into all users contents. This process work through a cleaning of the *Cache* and the issue of a command ordering a cache update from the clients.

The client module, is composed by some other components, which are:

- **KolbsTest**, which is an implementation of the twelve questions in the Kolb model, that is presented to the user when he/she does the first login in the system. This component executes all tabulations needed to classify the user into one of the four types defined in the model. The user cannot execute any other movement before its completion, when the system will store the user's profile, that should be used by the server during the next sessions, at the XML→HTML translation.
- **Login**, which is an interface that provides the user authentication.
- **UserData**, that is the component in charge of providing all the communications between the client and the server through a pipe channel.
- **ReceiveFromServer**, that receive warnings and commands issued by the server and redirects them to the *UserData* component.
- **Tutor**, which is the main component inside the client, being responsible for the activation of the remaining components.
- **ContentMngr**, that provides a graphical interface, where two windows are displayed to the user. The left window displays the content tree, as it is given by the *ContentTreeMngr* from the server. The right window displays the actual content currently selected by the user. This content assumes different forms for each learning style.

B. The XML structure

As the learning styles defined by the Kolb inventory have differences and similarities, the adoption of XML, where the contents must have a strong structure is very attractive. All the topics of a course can be easily assembled into a chapters-sections organization that are mapped by the DTD (Document Type Definition) definitions file. Links between topics are also conveniently maintained by derivations of the directives found in the DTD. This structure reduces the amount of information that has to be stored into the system, leaving the work of combining redundant data for the moment of the translation of XML contents to HTML pages.

The complexity of this work is the DTD definition, where all the XML organization will be created. In this case, the DTD must define all profiles that an information can get based on the four learning styles given by the Kolb's model. This is done in two steps: one with general organization data and another with personal profile organization data. While the general step is easily defined, since it contains general information, the profile step needs a detailed description of the similarities and differences among all styles, what is a complex task.

```
<!ELEMENT Chapter (ChapTitle, ChapDescr, Section+)>
<!ATTLIST Chapter
    id ID #REQUIRED
 owner (generic | assimilator | diverger | accommodator |
  converger | accommodator_diverger |
  diverger_assimilator | assimilator_converger |
  converger_accommodator ) #REQUIRED>
<!ELEMENT ChapTitle (#PCDATA|%htmltext;)*>
<!ELEMENT ChapDescr (p)>
```

FIGURE 1. DTD section defining a chapter

B.1 General structure

The general structure is concerned with the sections/chapters organization. As one can realize, this is a very simples structure, consisting on the syntax tree for chapters, sections and subsections. A small part of the DTD is shown on figure IV-B.1. There it is possible to see that a chapter is composed by three elements: its title (*ChapTitle*), its description (*ChapDescr*), and the sections on it (*Section*).

A section has also a simple definition, being composed by five components: its title (*SectTitle*), its description (*SectDescr*), its contents (*ContSect*) or its subsections (*SubSect*), and the section tests (*Tests*), as shown at figure IV-B.2.

On both definitions a major detail is that a mandatory definition is the *owner* of a chapter or section. This *owner* is one of the types defined by Kolb plus a set of combinations of these types. These combinations, as explained later, enable the reduction in the amount of data that has to be stored by the system.

B.2 Profile structure

This part of the DTD is in charge of the definition about what style must be used at every moment. It does, actually, the definition of what parts of the XML material should be translated to HTML and sent to the client machine. It relies on the definition of nine types of material owner, which are the four types from Kolb, four combinations of these four basic types, and a generic owner, which will cover all other styles.

```
<!ELEMENT Section (SectTitle, SectDescr, (SectCont |
    SubSection)+,Tests*)>
<!ATTLIST Section
    id ID #REQUIRED
 owner (generic | assimilator | diverger | accommodator |
  converger | accommodator_diverger |
  diverger_assimilator | assimilator_converger |
  converger_accommodator ) #REQUIRED>
<!ELEMENT SectTitle (#PCDATA|%htmltext;)*>
<!ELEMENT SectDescr (p)>
```

FIGURE 2. DTD section defining a section

Therefore, the profile structure is built by the use of the *owner* attribute, with the following values and coverages:

- **generic** → covers all types, therefore a content owned by a generic attributed will be shown to all users;
- **assimilator** → covers the assimilator type, and this will be the only type to see the materials tagged this way;
- **converger** → covers the converger type, that will be the only one to see this material;
- **diverger** → does the same for the diverger type;
- **accommodator** → does the same for the accommodator type;
- **accommodator_diverger** → covers the accommodator and diverger types, presenting material tagged this way to both types;
- **diverger_assimilator** → does the same for the diverger and assimilator types;
- **assimilator_converger** → does the same for the converger and assimilator types;
- **converger_accommodator** → does the same for the converger and accommodator types.

The correct translations are commanded by the XSL file, as dictated by the *XMLXSLManipulator* component of the server module. All translations are directed by the *owner* and *id* parameters set by the server when the user logs in the system, using the personal profile stored for this user.

V. Results

All definitions and conversions found on the DTD and XSL files were used during the implementation of RTtutor. The contents of the Real-Time Systems course, in Portuguese, were partially stored in the system using the XML directives and attributes. That enabled a set of tests using the tool, initially as a benchmark, and then as a learning tool. Figure V shows a snapshot of a webpage in the course.

The conducted tests tried to verify RTtutor's effectiveness in separating content by style and its efficiency with respect to portability and performance. A more qualitative evaluation of its effectiveness as a learning tool needs a larger set of students, which could not be achieved within its first application on the Real-Time Systems course, that comprised a class of only 15 students. However, preliminary results from this class indicates that the students were satisfied by the different presentations that RTtutor offered.

With the respect to its portability, several combinations of platforms were used to execute both the clients and the server. RTtutor ran fine in all system configurations, that included MS Windows, linux, and Solaris, combined in every possible way. The only restriction was that the server must be executed with special run-time parameters in order to achieve efficient performance even when the system was loaded.

The system performance is very adequate, in all possible configurations. The time spent loading pages is lower than a second in most cases. This happened even when the server was overloaded (more than 20 clients) and no caching was available. Among all configurations attempted, those where the server was running on linux or Solaris achieved the best results, specially when the server was overloaded.

The performance study also evaluated the influence of the *Cache* component inside the server. The general remark here is that the use of caching improves significantly the system's performance. The worst improvement came for the Windows server, which had an improvement of only three times. The best case was when both server and clients were running on linux systems, where the time for downloading was reduced by 100 times.

Finally, the effectiveness of RTtutor in the content separation was evaluated through the generation of several parts of the material for all styles. The HTML files generated were always different, with the same information being displayed in different forms. As an example, one given section generated a 22Kb file for a diverger person, while an accommodator person had the same section in a 16Kb file.

VI. Conclusions

Since the main goal of this work was the design and implementation of a computer-based learning tool that uses learning styles to present the course content, it is possible to claim that RTtutor is a successful accomplishment. This success is noticeable by the results just presented, where the tool showed a good performance and portability, being able to run well on several configurations of hardware and operating systems.

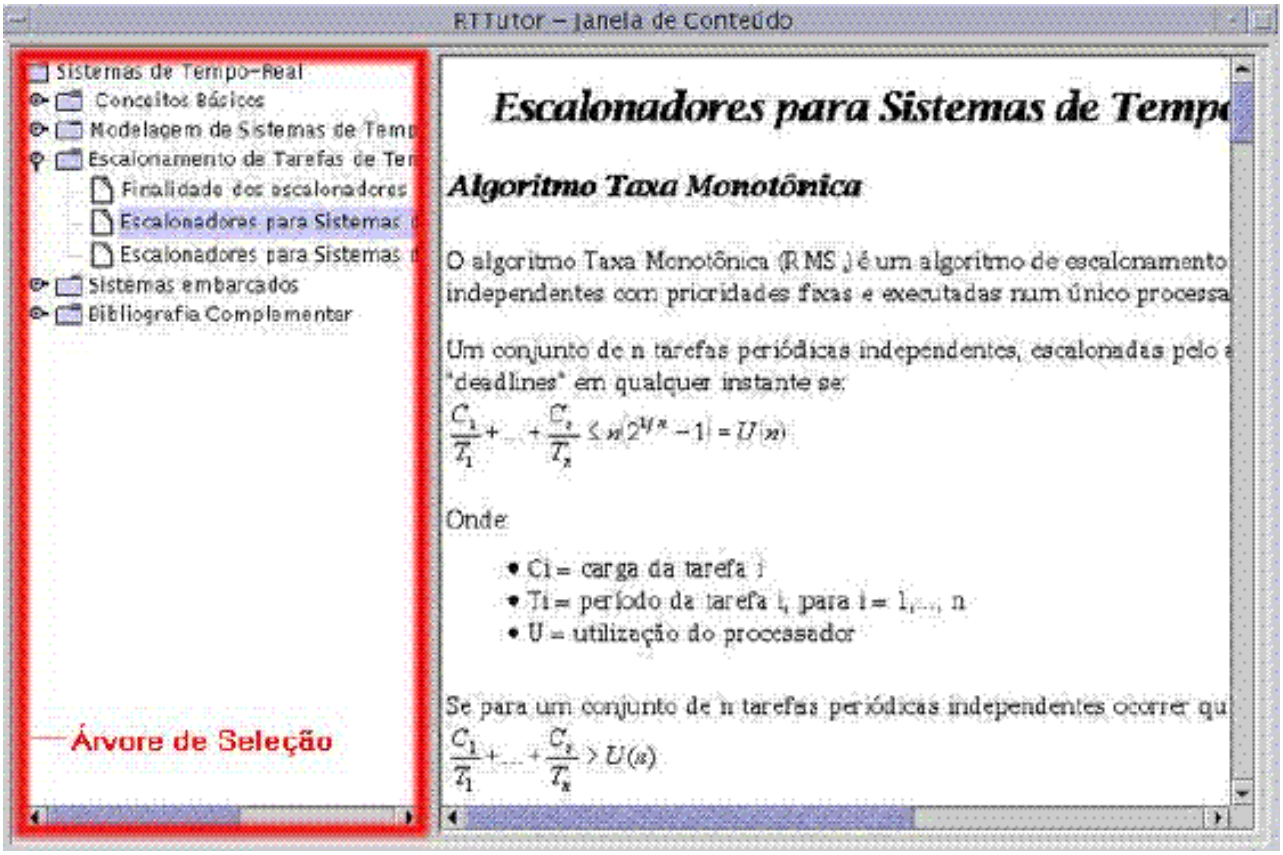

FIGURE 3. Illustrative example on how the content is displayed at the client interface

It is also noticeable that it provided a reasonable structure to manage the differences and similarities among the learning styles defined by Kolb, which was, at last, our main goal.

As always, this is not a final work. The new directions to be followed include a tool for material authoring, which would enable authors to insert course materials without the knowledge of the XML patterns, and some improvement on the server capabilities, such as a stronger management and, bookkeeping, over the students activities in the system. The absence of these improvements, however, does not reduces the effectiveness of using XML as a base language to develop a computer-based learning tool.

References

[1] Ahern, T.C., and Van Cleave, N.; *The Mentor project: from content to instruction*, in 32nd Frontiers In Education Conference, pp F2E-8:13, Boston, 2002.

[2] Curry, L.; *Integrating concepts of cognitive or learning styles: a review with attention of psychometric standards*, Canadian College of Health Service Executives, 1987.

[3] Daku, B.L.F., Jeffrey, K.; *An interactive computer-based tutorial for MATLAB*, in 30th Frontiers In Education Conference, pp F2D-2:7, Kansas City, 2000.

[4] Dunn, R., Dunn , K., and Perrin, J.; *Learning Style Inventory Manual*, Price Systems, 1979.

[5] Kolb , D.A.; *Learning Style Inventory*, McBeer, 1976.

[6] Kurtz, B.L., Parks, D. and Nicholson, E.; *Effective internet education: strategies and tools*, in 32nd Frontiers In Education Conference, pp F2E-14:19, Boston, 2002.

[7] Manacero, A. Jr., Miola, M.B., and Nabuco, V.A.; *Teaching real-time with a scheduler simulator*, in *31st Frontiers In Education Conference*, pp T4D-15:20, Reno, 2001.

[8] Richkus, R., Agogino, A.M., Yu, D., and Tang, D.; *Virtual disk drive design game with links to math, physics and dissection activities*, in 29th Frontiers In Education Conference, pp 12C3-18:23, San Juan, 1999.

[9] Spalter, A.M., Simpson, R.M., Legrand, M., and Taichi, S.; *Considering a full range of teaching techniques for use in interactive educational software: a practical guide and brainstorming session*, in 30th Frontiers In Education Conference, pp S1D-19:24, Kansas City, 2000.

[10] Sward, K., Terpenny, J.P., and Sullivan, W.G.; *Design, layout, and tools for effective web-based instruction*, in 32nd Frontiers In Education Conference, pp S1E-1:6, Boston, 2002.