A Computer-Based Learning Tool Using XML to Enable Learning Styles

Aleardo Manacero Jr., Alisson G. Casagrande, and Odnei C. Lopes

Dept. of Computer Science and Statistics,
São Paulo State University - UNESP
aleardo@ibilce.unesp.br

Abstract. The use of learning styles inventories is an interesting approach to adapt teaching procedures to student capabilities. Each type of inventory uses different characteristics in order to classify profiles and each profile needs different teaching approaches. In cases that these approaches mean differences in content it becomes difficult to maintain a computer-based learning tool, since the information has to be replicated for each style. In this paper a computer-based learning tool that uses Kolb's inventory is presented. It overcomes problems related to the volume of data offered to students and also the replication of information oriented to different styles. The tool uses the XML (eXtended Markup Language) in order to provide an optimized information structure and a useful environment for online application. A prototype was built and used to help learning real-time systems by computer science students.
Keywords: Learning styles, computer-based learning, XML.

1 Introduction

Use of computer-based tools to aid teaching and learning processes is a common technique, found at many institutions and courses ([1–3]). Computer-based tools have 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 direct condensations of material already found in books. Exceptions to this approach are concentrated on courses aimed 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 ([4, 5]), with few attempts on the game-oriented structure [6].

When using conventional computer-based learning tools two problems arise: the volume of data that is presented and the lack of personal styles on content presentation. Both problems are avoidable during the design of a tool, but their solution imposes different constraints that result in other problems.

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

In this work we propose an approach based on XML (eXtended Markup Language) to implement computer-based learning tools that use learning styles to improve the performance of the students. This approach reduces the volume of data that is presented (and stored) and enables presenting the contents based on learning styles.

The next section discusses some of the problems related with computer-based learning tools, as well related work. It is followed by a review about Kolb's learning styles. The methodology of the XML approach is presented on section 4. Results about the application of this approach into a computer-based learning tool for teaching real-time systems are shown during the last sections.

2 Problems on Computer-Based Learning

A major advantage on using conventional structure in computer-based learning tools is that the course can be assembled very fast. Most of the material is already in some electronic form, what makes easy to put it together in the tool's database. However, this procedure has some drawbacks:

1. it overloads the students with information (sometimes with useless data);
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 result in 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 attempted in order to provide that control. However, most of the solutions strongly rely on hypertext (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 given 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 represent improvements, they 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.

Differences on learning styles have been neglected by most of computer-based tools. Some of these tools attempt to deal with this issue, but usually their solutions are restricted in the tools appearance, that is, its colors, luminance, size of characters, etc. Although these parameters have some impact on how the student learns [7], they are only a small part of their preferences. Curry [8] shows that the learning style is built on several levels (environmental, interaction, information processing and personality preferences). The interface customization, dealt by the mentioned approaches, attacks only the environmental preferences, leaving the other levels unattended.

The information processing level in Curry's model for learning preferences is quite important since the actual understanding of a given content depends on how that information can be processed. Every student has personal preferences about the way they process information. Some prefer to take a more active role, others prefer to passively receive information. Others prefer concrete experiences while some students 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 spend more time and effort to achieve similar results. Besides this clear impact, computer-base tools usually do not implement information processing preferences in their structure.

One inventory that deals with the information processing level is the Kolb Learning Styles Inventory [9]. It defines four styles (described in 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 proposed style. Although feasible, this approach largely increases the work that has to be done by a instructor in order to make such course available. Actually, 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.

The hazards related to the application of more complex learning styles in computer-based tools are somewhat restricting their impact in the learning process. One of the reasons is the level of abstraction involved to distinguish different styles. 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 conducted to its use into this approach are described next.

3 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 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 are 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 provide the classification in one of 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, including the instructors, working well in brainstorming sessions. They are called divergers because they can see things from different perspectives.
- **Assimilator (type 2)**
  Their typical question is "What". They succeed when information comes in a logical and organized fashion (the conventional student). They get information from abstract conceptualization and process it through reflective observation and perform 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.
- **Converger (type 3)**
  They have "How" as their defining question, enjoying 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. They do not want to remain long periods in one activity (classes, reading, watching, etc.), preferring to go directly to the point (that is why they are called convergers). In order 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 (the reason behind 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.

Most of these characteristics are easily achievable by a computer tool. For 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 preferences, demanding a multiplication of information that has to be inserted by the instructor. Fortunately, several characteristics are present in more than one style, enabling information reusability. In the next section it is described a tool, RTtutor (**R**eal-**T**ime **tutor**), implemented using Kolb's inventory where the use of XML enabled the reuse of information.

4 RTtutor's Design

The RTtutor is a tool aimed to help the learning of real-time systems by computer science students. Its creation followed a previous project named RTsim

(**R**eal-**T**ime **sim**ulator)[10], which is a simulator of real-time scheduling algorithms that has been used as a laboratory tool to teach scheduling algorithms defined for real-time systems. From RTsim's use it became evident that a broader tool should be implemented in order to approach other relevant topics of the undergraduate course. Among the requisites for this tool was the use of learning styles strategies to expose the course's contents, which is performed by RTtutor.

RTtutor's design started from the definition 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.

4.1 The Java Client-Server Structure

RTtutor is composed by 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**: keeps the communication alive. Messages are requisitions from clients, answers or warnings/commands issued by the server.
– **ServerUserData**: performs the manipulation of users data (login name, password, course history and personal information).
– **TutorServer**: provides a GUI to server's control and configuration.
– **Cache**: reduces the overloading of content translations through the storage of information that has already been translated during the current session.
– **XMLXSLManipulator**: translates encoded XML information to HTML pages using the XSLT library.
– **ContentTreeMngr**: provides a tree-like structure for each learning style. It is presented by the client module in order to make the navigation easier.

The server is also responsible for updating all pages currently displayed by clients in case that the XML content gets modified by author intervention. Therefore, if the instructor wants to change some information, it gets automatically loaded to all users contents. This process works through the cleaning of the *Cache* and the issue of a command ordering a cache update from the clients.

The client module is composed by the following distinct components:

– **KolbsTest**: implements the twelve questions in the Kolb model, being presented to the user in his/her first login. This component classifies the user into one of the four types defined in the model. The user cannot do anything before its completion, creating the user's profile, that will be used by the server in the following sessions, during the XML→HTML translation.
– **Login**: provides a GUI for user authentication.

– **UserData**: provides all the communication, through a pipe channel, between client and server.
– **ReceiveFromServer**: receives warnings and commands issued by the server and redirects them to the *UserData* component.
– **Tutor**: is the main component inside the client, being responsible for the activation of the remaining components.
– **ContentMngr**: provides a GUI, where two windows are displayed to the user. The left window displays the content tree, as provided by the server's *ContentTreeMngr*. The right window displays the actual content currently selected by the user, assuming different forms for each learning style.

4.2 The XML Structure

Since the four learning styles defined in Kolb's inventory have differences and similarities between them, the adoption of XML, where the contents must have a strong structure is very attractive. All the topics of a course can be assembled into a chapters-sections organization that are mapped by the DTD (Document Type Definition) definitions file. Links between topics are also 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 when translation of XML contents to HTML pages actually occurs.

The complexity of this task is the DTD definition, where the XML organization will be created. Here the DTD must define all profiles that an information can get based in the learning styles defined by the Kolb's model. This is done in two steps: one with a general organization structure, which is easily defined, and another with a personal profile organization structure, which needs a detailed description of the similarities and differences among all styles.

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

```
<!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)>
```

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 below.

```
<!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)>
```

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 later explained, enable the reduction in the amount of data that has to be stored by the system.

Profile structure - This part of the DTD is in charge of the definition about what style must be used at each moment. It does, actually, the definition of what parts of the XML material must 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. 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, therefore students of assimilator profile are the only ones that see materials tagged this way;
– **converger** → does the same for the converger type;
– **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 students of 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.

5 Results

All definitions and conversions found in the DTD and XSL files were used during the implementation of RTtutor. The contents of a Real-Time Systems course, in Portuguese, were partially stored using the XML directives and attributes. That enabled a set of benchmarking tests and its validation as a learning tool. Tests verified RTtutor's effectiveness with respect of portability, performance and content presentation.

Several combinations of platforms were used to execute clients the server in order to verify their portability. RTtutor executed fine in all system configurations, including 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 had a larger processing load.

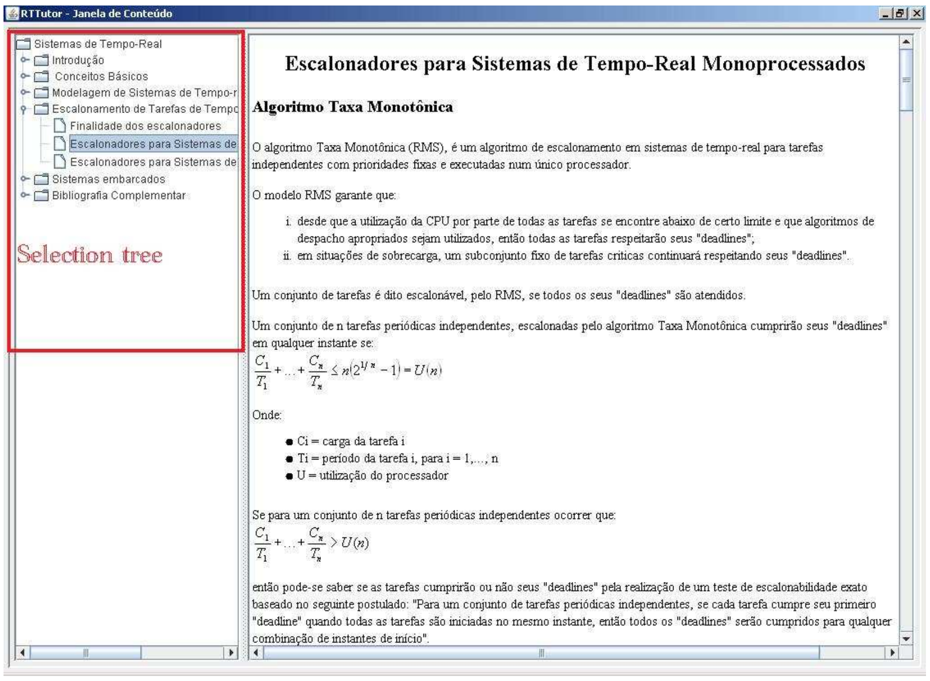
The system's performance is very adequate, in all possible configurations. The time spent to load pages is lower than a second in most cases, even when the server was overloaded (more than 20 clients) and no caching was available. Among all configurations attempted, those where the server was running 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 significantly improves the system's performance. The lowest improvement came for the Windows server, which had an improvement rate 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. Some of the measured times appear in Table 1.

Table 1. Loading times (in milliseconds) for a given page

Server	Client	Additional Clients	Without cache	With cache
Windows	Solaris	0	749.2	261.0
Solaris	Linux	0	751.5	98.8
Linux	Linux	0	1039.1	16.6
Windows	Solaris	20	1024.4	298.3
Solaris	Liunux	20	909.9	97.8
Linux	Linux	20	1220.5	49.5

A quantitative evaluation of its effectiveness as a learning tool was not performed since its preliminary application as an educational tool comprised only few tests over a small class (15 students) enrolled in a Real-Time Systems course taught to computer science undergraduate major. Besides this issue, and the absence of certain parts of the course content, the students answered very well to the formats that each one got during their exposal to RTtutor. A simple example of course content (in portuguese) is shown in Fig. 1. In that figure the region marked by the red box contains the selection tree for course content. The actual course content is displayed in the right region of the screen. In particular, this screen describes the Rate-Monotonic Scheduling Algorithm for a student belonging to the assimilator type.


RTTutor - Janela de Conteúdo
Sistemas de Tempo-Real
Introdução
Conceitos Básicos
Modelagem de Sistemas de Tempo-r
Escalonamento de Tarefas de Tempo
Finalidade dos escalonadores
Escalonadores para Sistemas de
Escalonadores para Sistemas de
Sistemas embarcados
Bibliografia Complementar
Selection tree

Escalonadores para Sistemas de Tempo-Real Monoprocessados

Algoritmo Taxa Monotônica

O algoritmo Taxa Monotônica (RMS), é um algoritmo de escalonamento em sistemas de tempo-real para tarefas independentes com prioridades fixas e executadas num único processador.

O modelo RMS garante que:

i. desde que a utilização da CPU por parte de todas as tarefas se encontre abaixo de certo limite e que algoritmos de despacho apropriados sejam utilizados, então todas as tarefas respeitarão seus "deadlines";
ii. em situações de sobrecarga, um subconjunto fixo de tarefas criticas continuará respeitando seus "deadlines".

Um conjunto de tarefas é dito escalonável, pelo RMS, se todos os seus "deadlines" são atendidos.

Um conjunto de n tarefas periódicas independentes, escalonadas pelo algoritmo Taxa Monotônica cumprirão seus "deadlines" em qualquer instante se:

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \leq n(2^{1/n} - 1) = U(n)$$

Onde:

- Ci = carga da tarefa i
- Ti = período da tarefa i, para i = 1,..., n
- U = utilização do processador

Se para um conjunto de n tarefas periódicas independentes ocorrer que:

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} > U(n)$$

então pode-se saber se as tarefas cumprirão ou não seus "deadlines" pela realização de um teste de escalonabilidade exato baseado no seguinte postulado: "Para um conjunto de tarefas periódicas independentes, se cada tarefa cumpre seu primeiro "deadline" quando todas as tarefas são iniciadas no mesmo instante, então todos os "deadlines" serão cumpridos para qualquer combinação de instantes de início".

Fig. 1. Illustrative example of how the content is displayed at the client interface (text is in Portuguese)

Finally, the effectiveness of RTtutor in the content presentation 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.

6 Conclusions

The tests performed over RTtutor shows that its implementation is efficient in terms of portability and processing speed. Its performance, even in presence of a larger number of clients is more than adequate for internet-based learning purposes.

The use of Kolb learning styles to arrange the course contents is an interesting approach, based on the preliminary application of RTtutor. It becomes even more interesting since the use of XML to organize the content structure reduces the amount of data inserted in the course's database. The differences and similarities found among the four learning types could be easily managed from the DTD structure.

A future improvement in RTtutor is the implementation of an authoring tool, which would enable authors to insert course materials without the knowledge of the XML patterns. Another trend in this work should consider some improvements in the capabilities of the server module, such as a stronger bookkeeping over the students activities in the system.

References

1. 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).
2. 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).
3. 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).
4. 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).
5. 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).
6. 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).
7. Dunn, R., Dunn, K., and Perrin, J.: Learning Style Inventory Manual, Price Systems, (1979).
8. Curry, L.: Integrating concepts of cognitive or learning styles: a review with attention of psychometric standards, Canadian College of Health Service Executives, (1987).
9. Kolb, D.A.: Learning Style Inventory, McBeer, (1976).
10. 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).