A Language for Describing the Generating Structure of the Educational Material in the Individually Adaptive Learning Management System

Svetoslav Zabunov

Abstract: *This paper summarizes the structure of a complete Individually adaptive learning management system (IALMS) and defines a formal language for describing the generating structure of the educational material in IALMS. It tends to establish the structural model and functionality characteristics of the Adaptive subsystem of the IALMS model. The generating structure language is explained and fully syntactically defined.*

Key words: *E-Learning, Individually Adaptive Learning Management System, Adaptive Learning Systems, Intelligent Learning.*

INTRODUCTION

Modern learning management systems (LMSs) utilize the multimedia and communicational computer means, but they scarcely take into consideration most resources for strategically planned conducting of educational process. Intelligent LMS should have the ability to interact with the user and thus to perceive information characterizing the user's behavior. Then on those bases the intelligent e-learning system should change its behavior to achieve the best result out of the educational process.

We call the process of LMS's behavior alternation according to the user's actions *adaptation* of the LMS towards the educated [3].

The existing systems and theoretical works on adaptive educational systems define models of e-learning systems with strictly predefined categories and structures of the individual information about the learner and the educational material. Kinshuk and Taiyu Lin give a good example [5]. The work presents a formalization scheme of the Exploration Space Control (ESC). ESC is described by Kinshuk et al. [4] and defines the process of adaptation in two categories: adaptation of educational content and adaptation of learning paths (links). ESC also defines a number of adaptation control levels. These are certain operations on the adaptation process categories. The Adaptive subsystem of the individually adaptive learning management system, the subject to this article, explores learning on generalized bases without predefined pedagogical categories and strategies. This approach proposes a versatile mechanism for formalization of different adaptive learning models. To achieve this task, a formal language for describing the generating structure of the educational material is used as a part of the Adaptation subsystem [1]. The latter is represented with its flexibility of defining both the categories of the individual information and the educational material structure.

INTRODUCTION TO THE IALMS STRUCTURE

An intelligent LMS has the following functionality. The behavior of the LMS adapts to the behavior of the educated, aiming at achieving the assigned results. The behavior adaptation is carried out by the *Adaptive subsystem* (AS). Individualization is required in order to attain adaptation. The LMS individualization is performed by the *Individualizing subsystem* (IS). IS is responsible for the creation of an individual profile for each learner. To enable LMS to follow the learner's behavior, it is needed a system for tracking events that carry characteristic information. This system is called *tracking subsystem* (TS). Finally, to let the user communicate with the system, an auxiliary interface unit is required. Thereby, the structure of an intelligent, adaptive, individualizing, and tracking system arises. It is called the *Individually Adaptive Learning Management System (IALMS)*.

The information sent to the educated and received back from him/her, is called *educational material*. In this case, this concept is more general than its standard sense. The educational material consists of *sessions*. A session is composed of educational

blocks. Blocks are passive and active. Being an information stream between the educated and the system, the educational material should carry two-way information: from IALMS to the learner (outgoing information) and from the learner to IALMS (incoming information).

All blocks may convey outgoing information, but only a part of them may convey incoming information. The latter are called active blocks, and the rest – passive blocks.

ADAPTATION SUBSYSTEM

The main function of the adaptation subsystem is the generation of educational material. The educational material is structurally composed of sessions. Each session corresponds to a lesson from a classic textbook. The difference is that the session has alternating content, aiming at adaptation to the qualities and knowledge of the learner, while lessons have permanent content. Variations in the session's content take place in the educational material frame of one lesson. The degree of alternation is determined at the definition of the *Educational material content (EMC)*. The Educational material content along with the *Generating structure of the educational material (GSEM)* are the major informational units of the Adaptation subsystem. EMC is fully programmable as it is defined for each session. On the other hand, GSEM is defined only once for the whole educational course and is the same for all sessions.

After an educational course has been encoded into IALMS the generating structure of the educational material and the educational material content have been defined for each session of the course. When the user starts working with the system or finishes working on a certain session and needs to continue his or her education, IALMS generates a new session on three stages:

1. Choosing a new working session - carried out by GSEM
2. Choosing the working session content - carried out by GSEM
3. Generation of the working session content - carried out by EMC

EDUCATIONAL MATERIAL CONTENT (EMC)

EMC consists of two types of components: generators and evaluators. These are functions. To each type of block in GSEM corresponds a pair of functions – one generator and one evaluator. The generator is a function that generates educational material for a given block of a given session. It looks like this:

```
<generator_id> ( <session_id> );
```

`<generator_id>` is the identifier of the generating function; `<session_id>` is the identifier of the session, which the current block belongs to. As a result the educational content of the given block of the given session is returned.

The declaration of the evaluator function follows:

```
<evaluator_id> ( <session_id> );
```

`<evaluator_id>` is the identifier of the evaluator function; `<session_id>` is the identifier of the session, which the current block belongs to. As a result, a relative value for the importance of the given block of the given session is returned.

GENERATING STRUCTURE OF THE EDUCATIONAL MATERIAL (GSEM)

GSEM has the structure of a reversed tree. To each leaf there is a corresponding type of block. GSEM determines the structure of all sessions of the educational course. The type-block is an informational structure that conforms to a certain type of block from the educational material. As it was noted, educational content consists of sessions, while they on their part consist of blocks. Blocks are the smallest divisible compounds of the educational material. To the type-block is assigned a generator and an evaluator both of that belong to EMC. A type-block or a leaf is declared in the following way:

```
{ leafp | leafa } ( <generator_id>, <evaluator_id> )
```

`<generator_id>` is the identifier of the generator, attached to the type-block; `<evaluator_id>` is the identifier of the evaluator, attached to the type-block. `leafp` and `leafa` are predefined functions, creating passive and active blocks respectively.

During the first stage of educational content generation (Choosing a new working session) the importance of each session of the educational course is evaluated. The one with highest importance is chosen. The evaluators in the GSEM leaves determine values of importance for each block from each session. The intermediate nodes in the GSEM tree represent relational functions with several arguments and one return value (fig.1). To estimate the need for assimilation of a given session, the results from the evaluators gradually come to the top of the tree, passing through the relational functions until reaching the GSEM root. On this stage relational functions execute the operation "addition" to the entry values. The sum is being returned as a result and is used for an entry to the next relational function. The value reaching the GSEM root is the searched value for the need of assimilation of the given session.

At the second stage of the session generation (Choosing the working session content) the Adaptation subsystem has already determined the session that is to be passed to the learner. The task is to determine its content. This problem is reduced to specifying the subset of type-blocks that should be included in the current session variant. Again, the GSEM is used. Each type-block importance value is calculated using the evaluators in the GSEM leaves. These values come to the root of the tree, passing through the relational functions. At this point, relational functions perform their real purpose. There are two types of relational functions: OR-type and XOR-type. When passing through a relational node of type OR, all type-blocks whose leaves have generated evaluation values greater than or equal to the required value are being combined and passed through. This threshold value is stored in the relational node. When passing through a relational node of type XOR, the type-block whose leaf has generated the greatest evaluation value is determined and if its evaluation value is greater that or equal to the required value, the type-block is let to pass through the relational node. Again, the threshold value is stored in the relational node. Eventually, there is a subset of all leaves (type-blocks of the session) reaching the GSEM's root. This subset represents all type-blocks that should be included in the current variant of the session.

Relational functions declarations. These functions are predefined in the language:

- OR-type:

```
<node> or ( <level>, <node>, ... );
```

- XOR-type:

```
<node> xor ( <level>, <node>, ... );
```

The `<level>` argument presents the threshold level of passing through the relational node and it is a constant in the GSEM language. Only type-blocks with evaluation values greater or equal to the threshold level are being let to pass through the relational node. The node list (`<node>, ...`) represents the input set of child nodes of the current one. The topology of the GSEM tree is being built defining the child nodes as parameters when declaring each of the relational nodes. The last nodes are, of course, leaves, declared using the functions `leafp()` and `leafa()`.

At the third stage (Generation of the working session content) generator functions are used to generate each block's content from the subset of type-blocks corresponding to the current session variant. Every generator is executed. The result is then merged and represents the educational material for the current session. It is then handed to the learner.

On fig.1 there is a sample scheme of a GSEM shown. There are type-blocks defined. They are of two types: active and passive. The active ones have dark background.

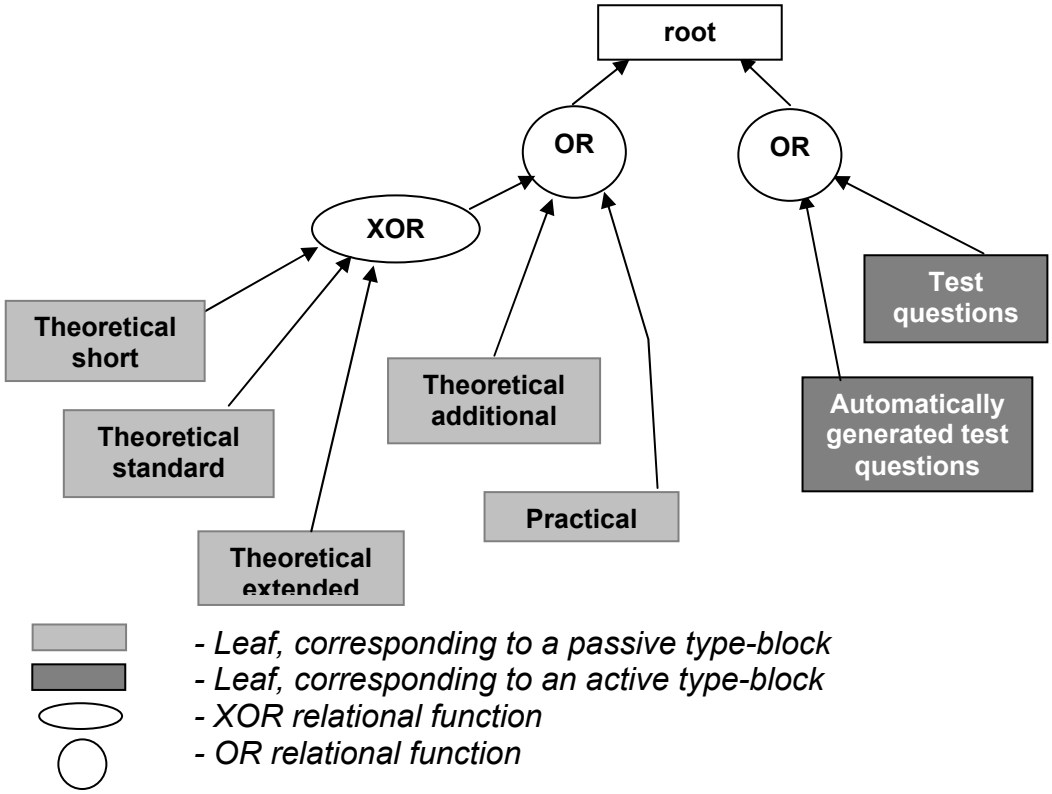

Figure. 1. *A sample structure of GSEM - one for all sessions.*

GSEM DESCRIPTION LANGUAGE DEFINITION

The GSEM description language is procedurally oriented. Each sentence consists of a set of expressions, separated with ";". There are two operators supported: assignment operator "=" and function call operator "()". The programming objects of the language are: variable, function, floating point scalar constant, generator, evaluator. All variables have the same type – node. All functions are predefined. All generators and evaluators are defined in EMC and from the GSEM point of view appear to be predefined programming objects. There should be paid attention to the fact that generators and evaluators are separate programming objects and are different from the functions in the GSEM description language.

The function call operator has the following syntax:

```
<function_call> ::= <func_id> (<argument_list>)
```

The assignment operator is used for node variables creation and their association with a node-object. Node-objects are created with the predefined language functions: `or(), xor(), leafp()` and `leafa()`. The operator "=" has the following syntax:

```
<assignment> ::= <lvalue> = <rvalue>
```

`<lvalue>` is always a variable or the predefined variable `root`. The latter has the node type as well and represents the tree's root. `<rvalue>` is either a variable or a function that returns a node.

For each leaf from the GSEM's tree a type-block is defined, consisting of generator of educational material and an evaluator function. The leaves correspond to two types of type-blocks: active and passive. Declarations of leaf-functions have the following form:

Declaration of a function that creates passive leaf/type-block:

```
<node> leafp ( <generator_id>, <evaluator_id> );
```

Declaration of a function that creates active leaf/type-block:

```
<node> leafa ( <generator_id>, <evaluator_id> );
```

The arguments <generator_id>(), <evaluator_id>() are defined functions in EMC.

Relational functions declarations have the form:

<node> or (<level>, <node>, ...) – Relational node OR

<node> xor (<level>, <node>, ...) – Relational node XOR

<level> is a floating point constant, corresponding to the threshold level of passing through. <node> is a tree node.

In order a given sentence to be semantically completed, there should be at least one statement containing operation assignment to the predefined variable root.

SYNTACTIC LANGUAGE DEFINITION:

```
<s>                 ::= <statement_list>
<statement_list>    ::= <statement> [; <statement_list>]
<statement>         ::= <assignment>
<assignment>        ::= <lvalue> = <rvalue>
<lvalue>            ::= root | <var_id>
<rvalue>            ::= <var_id> | <function_call>
<function_call>     ::= <leaf_fc> | <rel_fc>
<leaf_fc>           ::= { leafp | leafa } (
                        <generator_id>, <evaluator_id> )
<rel_fc>            ::= { or | xor } ( <level>,
                        <node> [, <node>]* )
<level>             ::= <float_const>
<node>              ::= <rvalue>
<var_id>            ::= <c_u> <c_d_u>*
<c_u>               ::= <char> | _
<c_d_u>             ::= <char> | <digit> | _
<float_const>       ::= [+|-] ((<int>.[<int>]) |
                        ([<int>].<int>)) [(e|E) [+|-]
                        <int>]
<int>               ::= <digit> <digit>*
```

The sample GSEM from fig.1 is used to illustrate the language syntax. The passive type-blocks are: theoretical short, theoretical standard, theoretical extended, theoretical additional, and practical. The active type-blocks are: test questions (standard_test) and automatically generated test questions (auto_test).

The type-blocks "theoretical short", "theoretical standard", and "theoretical extended" are combined with a relational node XOR. The meaning of this join is that only one of them will be generated and included in the session. These type-blocks correspond to different presentation of one and the same educational material. The type-blocks "theoretical additional" and "practical" represent additional educational material to the basic one, and hence they are joined with a relational node OR. The same is true for the active blocks in the sample GSEM – they are combined with an OR function.

```
short = leafp (g_short, e_short);  // Defines a leaf type
standard = leafp (g_std, e_std );  // corresponding to a
extended = leafp (g_ext, e_ext );  // type-block.
theoretical_basic = xor (2.4, short, standard, extended);

standard_test = leafa (g_test, e_test);
```

```
auto_test = leafa (g_atest, e_atest);
tests = or (1.54, standard_test, auto_test);

theoretical_add = leafp (g_ta, e_ta);
practical = leafp (g_pr, e_pr);
passive = or (2.7, theoretical_basic, theoretical_add,
              practical);

root = or (1.98, passive, tests);
```

Here all generators and evaluators are defined in ECM. On these bases, through a concise description the GSEM tree is constructed.

CONCLUSIONS AND FUTURE WORK

The definition of a generalized model of an individually adaptive learning management system makes possible the fast creation of specialized e-learning systems for a given educational subject or a group of subjects.

In particular, the Adaptive subsystem of IALMS becomes open to different pedagogical paradigms and structures of the educational material used by the instructor.

REFERENCES

[1] Иванов, К., С. Забунов - Модел на индивидуално-адаптивна система за електронно обучение, Трудове на Научната Сесия РУ'2003, 2003.

[2] Маждраков, М., Т.Трендафилов и др. Обучението по ГИС в МГУ "Св.Иван Рилски". Международен симпозиум "Приложение на лазерни, GPS и GIS технологии в геодезията", София, 1996.

[3] Brusilovsky, P. Adaptive and Intelligent Technologies for Web-based Education - KI - Kunstliche Intelligenz, 1999.

[4] Kinshuk, A., R. Oppermannn, R. Rashev, H. Simm, A Cognitive Load Reduction Approach to Exploratory Learning and Its Application to an Interactive Simulation-Based Learning System - Journal of Educational Multimedia and Hypermedia, 9 (3), 253-276 (2000).

[5] Kinshuk, A., T. Lin, User Exploration Based Adaptation in Adaptive Learning Systems - Information Systems Department Massey University, Palmerston North, New Zealand, 2003

[6] Zabunov, S., K. Ivanov. Methods and Forms of Teaching "Information Systems" and "Computer Networks and Conmmunications" with the Use of the Internet. Annual of University of Mining and Geology "St. Ivan Rilski" - Sofia, 2003.

ABOUT THE AUTHORS

Eng. Svetoslav Svetoslavov Zabunov, Department of Informatics, University of Mining and Geology "St. Ivan Rilski" - Sofia, Phone: +359 2 9627220/564, E-mail: SvetoslavZabunov@web.de