

A Framework for the Generation of Adaptive Courses based on Semantic Metadata

Freddy Duitama, Bruno Defude, Amel Bouzeghoub, Claire Carpentier

Department of Computer Science
GET-Institut National des Télécommunications
9, rue Charles Fourier - 91011 Evry cedex France
Firstname.Lastname@int-evry.fr.

Abstract. Our approach proposes the creation and management of adaptive learning systems by combining component technology, semantic metadata, and adaptation rules. A component model allows interaction among components that share consistent assumptions about what each provides and each requires of the other. It allows indexing, using, reusing, and coupling of components in different contexts powering adaptation. Two adaptive learning strategies are proposed: course-based and goals-based. These strategies use rules to rewrite user query and user model. In this way, it is possible both searching components developing concepts not appearing in the user query but related with user goals and inferring user knowledge that is not explicit in user model.

1. Introduction

Most of adaptive self-learning systems provide adaptation to a specific course using a hard-coded adaptation model [3, 5, 10]. Adaptive Hypermedia Architecture (AHA) [11] proposes to perform adaptation and to update the user model according to a set of adaptation rules specified in the Adaptation Model (AM). The Karina project [4] uses conceptual graphs to express semantic for both domain model and pedagogical model. Our approach proposes the creation and management of adaptive learning systems by combining component technology, semantic metadata, and adaptation rules.

Today, learning systems must support teaching objects of different nature (html pages, multimedia presentation, web-services, etc.). A component model allows interaction among components that share consistent assumptions about what each provides and each requires of the other. In our approach, the learning system is the framework to deploy educational component. It allows indexing, using, reusing, and coupling of components in different contexts powering adaptation. Our aim is to provide customized content from different knowledge domains. We provide a rule language, through which domain experts can specify how different aspects of domain model, user model, component semantic, and user query interact to generate a user-tailored presentation. Rules are used to rewrite user query and user model. In this way, it is possible both searching components developing concepts not appearing in the user query but related with user goals and inferring user knowledge that is not explicit in user model.

This article is structured as follows: Section 2 presents the logical architecture of our proposal; we describe Domain Model (DM), User Model (UM), and Component Model (CM). Section 3 presents the Teaching Model (TM). The rule language and the adaptation engine are described in section 4. Section 5 concludes the paper and presents some open issues.

2. Overview of our proposal

This section presents the logical architecture of our system. *Figure 1* illustrates the three levels of modeling we have defined. A more detailed description is given in [2].

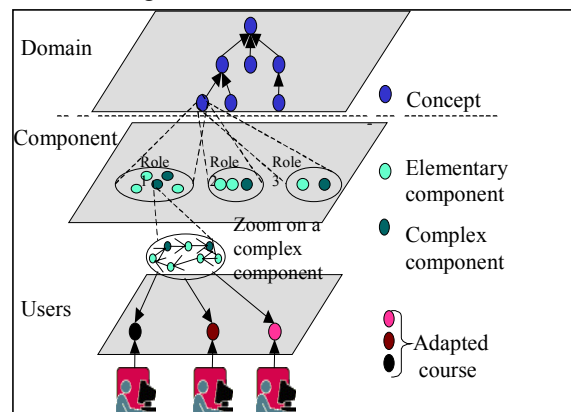


Figure. 1. Modeling levels

2.1. Domain Model

Our approach uses an ontology to represent all concepts from the knowledge domain. The ontology contains a hierarchical description of important concepts in the domain. Thus, additional relations between concepts can be defined using rhetorical relationships (Figure 2 gives an example of a domain model).

Let the domain model be a graph $G = \langle C, A \rangle$, where C are nodes representing concept from the DM model and A are edges representing relationships between two concepts. There are two kinds of possible relationships:

- *broader/narrower* relationship. Concept A is broader than concept B whenever the following holds: in any inclusive search for A all items dealing with B should be found. Conversely B is narrower than A [6]. Concepts organized with this relationship represent a tree.
- *rhetorical* relationships. We use a set of predefined rhetorical relationships took from [8]: *Antithesis*, *Background*, *Contrast*, *Extend* and *Restatement*.

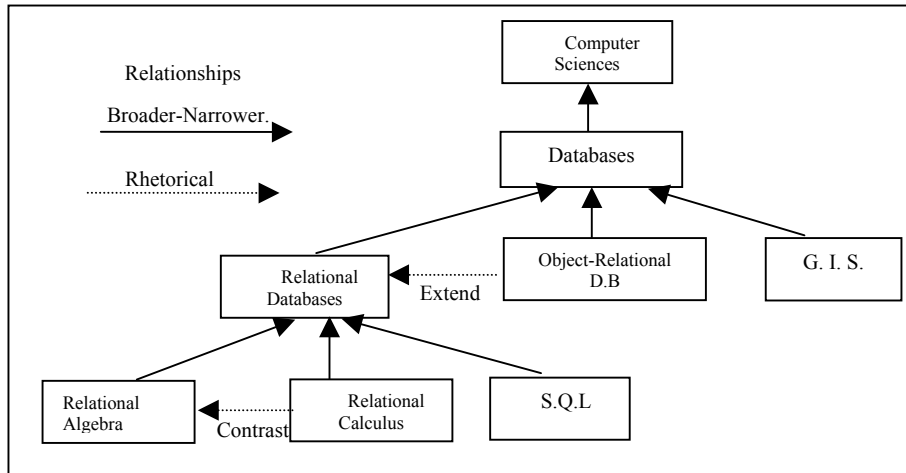


Figure 2. Domain Model Fragment

2.2. User Model

Adaptive learning systems adapt content to experience, knowledge, goals, and/or preferences of learners. We have considered the last three aspects designing UM. Some of this information is given statically by the user (e.g. his/her graphical preferences); while some others can be captured or modified dynamically by analyzing his/her behavior (e.g. his/her level relative to the concepts). We use an overlay model [3] to maintain an evaluation of learner's educational state relative to each concept. In consequence, the UM is defined by the tuple:

$$UM = \langle learner, preferences, domain-knowledge \rangle$$

Where *learner* corresponds to the user Id, *preferences* = {<attribute, value>} and *domain-knowledge* = {<role, concept, educational-state>}. Item *role* is optional and represents a link between a *concept* and an *educational component* (e.g. "introduction", "definition", "description", "application"), *concept* is taken from the domain model and *educational-state* is one value in ("not-visited", "visited", *knowledge-level*). Finally, *knowledge-level* is a value among the set {"very low", "low", "medium", "high", "very high"}.

If there are no information about the user, system administrator predefines stereotypes taking into account the student population characteristics.

2.3. Educational Component Model

An Educational Component is a unit of composition with an interface providing information about requirements for its use, coupling, or replacement during the

technology-based learning. This unit can be used independently or for composition by third parties.

A component can be primitive (atomic) or composed (structured). It should be noticed that its author fixes the component granularity; we do not define any constraint on this even if a fine granularity will imply a better reuse of the component.

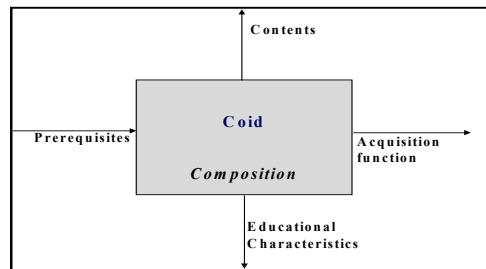


Figure. 3. An Educational Component

An educational component (Fig. 3) is described by a set of metadata:

$\langle Coid, Educational-Characteristics, Semantic, Composition \rangle$

with the following definition:

- *Coid* is the component unique identifier in the system. From now, C_i will denote it.
- *Educational-Characteristics* is a set $\{M_i\}$, $i = 1 \dots k$, where $M_i = \langle \text{tag}, \text{value} \rangle$. This set is used to describe non-functional properties of the components. Our approach uses Learning Object Metadata [7] (e.g. $\langle \text{"title"}, \text{"Relational Databases"} \rangle$, $\langle \text{"author"}, \text{"Dupont"} \rangle$).
- *Semantic* is defined by the tuple $\langle Content, Prerequisites, Acquisition-function \rangle$.
- *Composition* is null if the component is primitive or is an acyclic directed graph denoted by a canonical expression (see Fig. 4) if the component is structured.

Component Semantic

Content = $\{Rc\}$, where $Rc = \langle Coid, \{role\}, concept \rangle$. The set of roles describes topics developed by a component with respect to a domain concept, $role \in \{introduction, definition, description, application, evaluation, demonstration, summary\}$. These roles are based on the Educational Ontology [9].

Prerequisites. These ones are a set of binary relations between one component and one domain concept. It means that, during the learning process, a user must know this concept before using the component. Each prerequisite is represented by the tuple:

$Rp = \langle Coid, \{role\}, concept, knowledge-level \rangle$ where knowledge-level takes one of the following values: $\{\text{"very low"}, \text{"low"}, \text{"medium"}, \text{"high"}, \text{"very high"}\}$.

Acquisition function $f(A)$. It records how well students learn concepts developed by component. At learning time, for each couple (*domain-concept*, *role*) from *Content* of C_k : $f(A)$ adds $\langle \text{domain-concept}, \text{role}, \text{new-knowledge-level} \rangle$ in UM or FAIL. Note that $f(A)$ defines a mapping between knowledge representation used by this model and knowledge representation used by teacher inside component.

Example:

Let C_2 be a component. It has CT as *Content*, P as *Prerequisites* and F as *Acquisition Function*. Let suppose $CT = \{ \langle C_2, \{ \text{"introduction"}, \text{"definition"}, \text{"application"} \} \text{"Data-Model"} \rangle \}$ describes content for component C_2 . It introduces, defines, and applies the domain-concept *Data Model*. Whereas, $P = \{ \langle C_2, \{ \text{"definition"}, \text{"First-Order-Logic"}, \text{"High"} \} \rangle \}$ states that component C_2 requires preparation at *high* level on *First Order Logic* definition. Furthermore, if the component C_2 is validated, the couple (*Data-Model*, *definition*) belonging to its content is added to the user model with the value *medium*; i.e. $\langle \text{"Data-Model"}, \text{"definition"}, \text{"medium"} \rangle$. F proceeds in the same way for the other roles.

Note that to classify a component, author only needs defining relationships with concepts from DM. Our approach contrasts with LOM standard, which proposes metadata *Relation* to define relationships among learning objects. In our system, components are completely independent each other; it will facilitate reusability.

Composition

A structured component is an acyclic directed graph whose nodes represent primitive or structured components and the edges represent a crossing condition (see Fig. 4). Composition is obtained by applying composition operators of educational components. We formalize different composition operators to allow building (may be recursively) a complex component, starting from primitive ones (or from structured components in the recursive case):

- (i) **sequence**: ($C_i \text{ SEQ } C_j$) learners have to access components C_i and C_j . C_j can be accessed only if C_i is successful;
- (ii) **parallel**: ($C_i \text{ PAR } C_j$) learners have to access components C_i and C_j ; but C_i and C_j can be accessed independently; there is no order relation between them.
- (iii) **alternative**: ($C_i \text{ ALT } C_j$) learners have to access component C_i or_{exclusive} C_j . At learning time, the system will choose between C_i and C_j according to a particular UM.

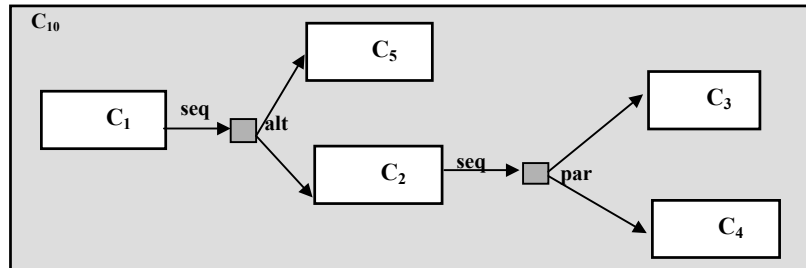


Figure. 4. Example of a structured component

Figure 4 shows a component that has been built with the following expression:
 $C_{10} = C_1 \text{ SEQ } (C_5 \text{ ALT } (C_2 \text{ SEQ } (C_3 \text{ PAR } C_4)))$

For a composed component, its composition provides information to process automatically almost all its metadata. For example, *Contents* of $C_m = C_i \text{ ALT } C_j$ is equivalent to contents of C_i or_{exclusive} contents of C_j . *Prerequisites* for $C_m = C_i \text{ PAR } C_j$

includes all prerequisites of C_i and C_j , except concepts developed by them. On the other hand, *Acquisition Function* for composed component can be defined from acquisition functions defined for C_i and C_j (for instance, by function composition).

3. Teaching Model

Our approach supports two learning strategies: course-based and goals-based learning. The latter allows user to define their goals from domain model, whereas the former provides guides and helps to meet course objectives. This section describes teaching model and introduces scenarios where adaptation is required. Adaptive system is materialized by combining the three levels of modeling above described: UM, DM and component semantic.

3.1. Course-Based Learning

Figure 5 describes the course-based learning process.

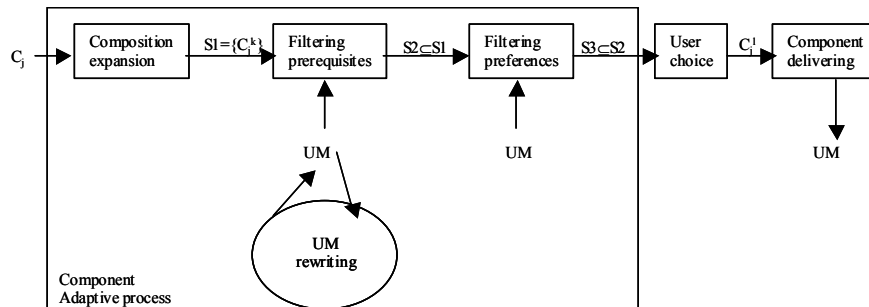


Figure 5 : course-based learning process

A user selects a component C_j from the component base. The composition graph of C_j is transformed to obtain a set S_1 of composition graphs without ALT operators. For example the following composition graph:

$C_{20} = C_1 \text{ SEQ } (C_2 \text{ ALT } C_3)$ is transformed into $S_1 = \{C_1 \text{ SEQ } C_2; C_1 \text{ SEQ } C_3\}$

Further steps will select the “best” composition graph in S_1 according to the user model. Firstly S_1 is filtered using the user domain-knowledge to obtain set S_2 . If S_2 is empty the user model is updated using rewriting rules inferring new prerequisites. For example, if the component C_1 has concept “*Relational Algebra*” as prerequisite and learner knows the concept “*Relational Calculus*”, relationship “*contrast*” could be used to infer that learner is able to develop C_1 . The rewriting process can be repeated several times, but it always takes the original UM instead of the UM obtained in the previous rewriting. After that, S_2 is filtered using user preferences and we obtain S_3 . If there are several components in S_3 , the user chooses one which will be delivered. System builds a conceptual map of course with annotations indicating user’s educational-state for each node in composition. Finally, UM is updated from acquisition function of the course.

3.2. Goals-Based Learning

Goals-based learning process is separated in two distinct processes depending on the number of concepts including in the goal (single or multiple). *Figure 6* shows the single-concept process while *Figure 7* describes the multiple-concept process.

We describe below the single-concept process step by step.

- learner selects her/his goal and expresses it using a query Q_j formulated over the ontology in a declarative query language. Let us suppose learner asks for components, which introduce and define the domain concept “*Relational-Database*”;
- the system searches the set S of components, where each component meets concept and roles asked by query Q_j ;
- if S is empty, system rewrites query by using one between two possibilities. a) It takes one relationship of the ontology to search a related concept. e.g. *narrower* relationship of the concept “*Relational Database*” can be used to obtain from the query Q_j ; Q_j^1 = definition and introduction of “*Relational Algebra*” and Q_j^2 = definition and introduction of “*S.Q.L.*”. b) It modifies roles in Q_j ; e.g. role “*example*” can be used instead of role “*introduction*” in query Q_j . To define which relationship or role will be rewritten is rule-based. This process can be executed several times using different relations and roles. Q_j will be always the input-query instead of the expression obtained on the previous rewriting;
- system filters S using user model and preferences in the same manner than course-based process (component adaptive process);
- if the new resultant set has a large amount of components, system could rewrite query Q_j in order to obtain a more precise answer. e.g. if user query asks for concept “*Relational Database*” without to specify roles, this step could adds roles to concept in query. Again rewriting is rule-based;

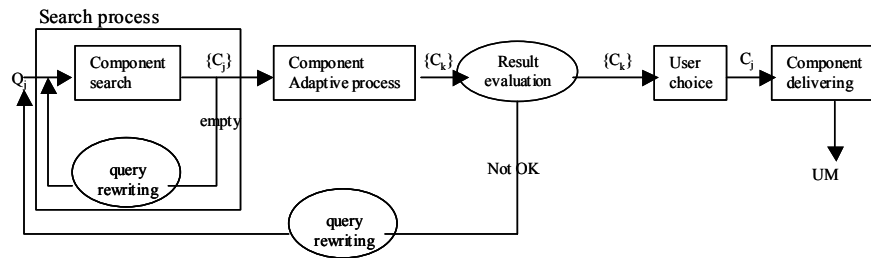


Figure 6. single-concept process

- learner chooses a component from the resultant set; then, system builds an annotated conceptual map of the component; i.e. a graph without operator ALT, but with annotations indicating user’s educational state for each node in composition (unknown components, ready to read components, etc.);
- the selected component is delivered to the user and his/her UM is updated from acquisition function of component.

When the goal is expressed with a combination of multiple concepts the process is more complicated (see figure 7). The process is changed in the following manner:

- learner expresses his/her query Q_j using multiple concepts;
- the system searches for S the set of components matching the query. If S is non empty the process is the same as for single-concept query;
- if S is empty the query rewriting is much more complicated than single-concept rewriting. In fact, the system has to decompose the query in a set of single-concept queries (Q_j^1, \dots, Q_j^k) . After that each query is processed to search for components (S_1, \dots, S_k) ;
- new components are generated dynamically from previous results (S_1, \dots, S_k) . Components in S_j ($j=1$ to k) are composed by an ALT operator which are in their turn composed by the PAR operator. We suppose that there is no order required between components;
- the end of the process is unchanged.

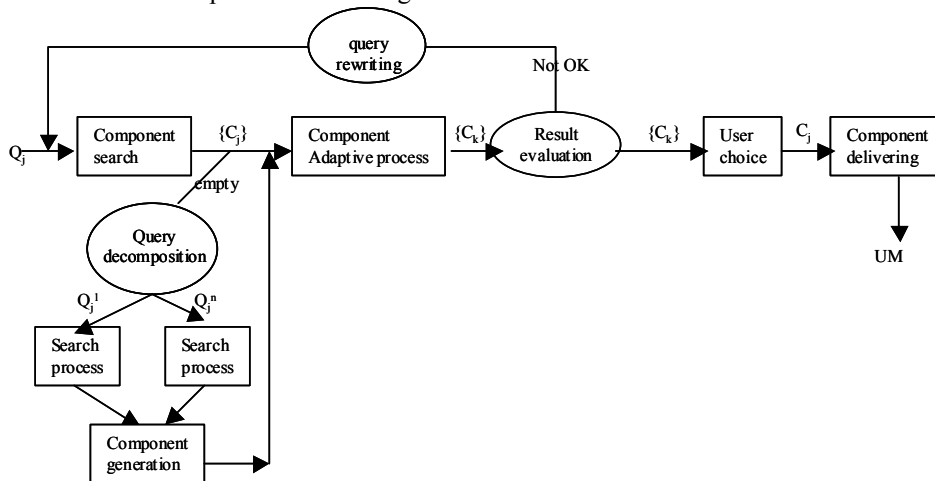


Figure 7. Multiple-concept process

4. Adaptation Engine

Adaptation Engine (AE) executes content adaptation according to a set of condition-action rules (CA-rules) and updates user model from values returned by acquisition function of components. The detailed description of the rule language and the rule execution model is out of the scope of this paper. This section just describes the rule language through several examples. AE supplies functionalities to support user query rewriting, UM rewriting, filtering of the resultant set and query decomposition.

Conditions of the rule language are constructed using predicates with variables, functions, logical operators, constants and predefined variables such as current user model or current query.

Rules are defined by domain experts. The system is initiated with a set of general rules allowing to express generic rewriting strategies. New rules can be added to overload the existing ones and to express more specific strategies well adapted to actual users and domain models. A priority mechanism allows to select the most specific rule for a given situation. Priority is processed with a metric which evaluates the specificity level of a rule.

We will use an example to describe how a rule action is executed. *Rule 1* covers concepts from all domains, but having a hierarchical level (distance from the root of the domain model using *narrower* relationship) between 1 and 3 and applies to occasional users. A rule is fired if its condition is true (it is in the active set) and there is no active rule with a higher priority. Rule 1 defines three strategies for rewriting queries. The first time, system takes concept in user query Q, the rewriting process uses relation "*contrast*" in domain model to obtain Q'. Note that, if concept in query has no "*contrast*" relation, system takes the following rewriting strategy in rule 1. On the other hand, if answer of Q' is empty, the rewriting process is executed again and it changes respectively, if used in query Q, roles "*definition*" and "*introduction*" by roles "*description*" and "*example*" in order to obtain a new query Q'. If answer of Q' is empty, the goal-based process requires rewriting process yet. In this case, system takes concept in query Q to find (if possible) "*narrower*" concepts; in this way, it builds as many queries as narrower concepts found. After that, if query answer is empty yet, system asks to user to redefine her/his goals.

Rule 1.

```
FOR REWRITING QUERY
IF q-contain(:uq, :concept)
  AND equal(user-type(:ul), "occasional")
  AND hierarchical-level (:concept) between 1 AND 3
USE
  ADD_RELATION("contrast", all);
  MODIFY_ROLES (
    "description" INSTEAD OF "definition";
    "Example" INSTEAD OF "introduction" );
  ADD_RELATION("narrower", all);
END;
```

Rule 2 defines two different strategies for rewriting UM. First time, the rewriting process uses (if possible) relations "*contrast*"; i.e. for each concept that appears in UM, system find if it has this type of relation, in order to search among known concepts by user, those having a relation "*contrast*" with prerequisite concepts of components found in step 2. Second time, rule defines to use relation "*narrower*"; again, system takes the original UM and for each concept finds its narrower ones, in order to evaluate if prerequisites concepts of components are a narrower concept among those known by user.

Rule 2.

```
FOR REWRITING USER MODEL
IF um-contain(:um, :concept)
  AND equal(user-type(:ul), "occasional")
  AND hierarchical-level (:concept) between 1 AND 3
USE
  ADD_RELATION("contrast", all);
  ADD_RELATION("narrower", all);
END;
```

Finally, *rule 3* defines filtering strategy; if the resultant set has a large amount of components (greater than 100) query is rewritten, in order to obtain a more precise answer.

Rule 3.

```
FOR FILTERING ANSWER
IF q-contain(:uq, :concept)
  AND equal(user-type(:ul), "occasional")
  AND answer-size(:uq) > 100
  AND hierarchical-level(:concept) between 1 AND 3
USE
  ADD_ROLES({"definition", "application"}, all);
END;
```

The rule language and the associated execution model are not completely defined. The challenge here is to have a good balance between rule expressivity and computational properties of the execution model such as termination.

5. Conclusion

In this paper we describe a semantic model allowing to describe domain models, user models and educational components. This model provides authors and learners powerful mechanisms to manage components, concepts and users (e.g browsing, querying, composing, classifying, ...). We also present learning strategies which can be used with this approach, that is course-based or goal-based and we detail the associated algorithms. These strategies are highly adaptive thanks to our rule-based mechanism.

A prototype of our system is being implemented. It is based on the RDFSuite [1] which uses RDF as its knowledge representation layer and RQL a declarative query language as inference layer. This prototype will allow us to validate our model and to begin some real experimentations.

References

1. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle, The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, 2nd International Workshop on the Semantic Web (SemWeb'01). (2001)
2. Bouzeghoub, A., Carpentier, C., Defude, B., Duitama, F.: A Model of Reusable Educational Component for the Generation of Adaptive Courses. Proc. of the First international Workshop on Semantic Web for Web-based Learning. Austria (2003)
3. Brusilovsky, P. Methods and Techniques of adaptive hypermedia. User Modeling and User Adapted Interaction Vol. 6 No. 2-3 (1996) 87-129
4. Crampes, M., Ranwez, S. : Ontology-supported and ontology-driven conceptual navigation on the World Wide Web. Proceedings of the eleventh ACM on Hypertext and hypermedia. Texas (2000). 191-199
5. El Saddik, A., Fischer, S., Steinmetz, R.: Reusability and Adaptability of Interactive Resources In Web-Based Educational. ACM Journal of Educational Resources in Computing, Vol. 1. No. 1, (2001)

6. Fischer, D.: From Thesauri towards Ontologies? Proceedings 5th Int. ISKO-Conference. France (1998) 18-30
7. IEEE. Draft Standard for Learning Object Metadata. (IEEE P1484.12.1.) (2002) Available from <<http://ltsc.ieee.org/>>
8. Mann W. and Thomson S., Rhetorical Structure Theory: A theory of text organization. Technical Report RS-87-190, Information Science Institute, (1987)
9. Ranwez, S., Leidig, T., Crampes, M.: Pedagogical ontology and Teaching Strategies: A New Formalization to improve Life-long Learning. Journal of Interactive Learning Research, Vo. 11, Number 3-4 , (2000)
10. Weber, G., Specht, M.: User modeling and Adaptive Navigation Support in WWW-Based Tutoring. Proceedings of the sixth International Conference on User Modeling. Vienna, Italy, (1997) 289-300
11. Wu, H., De Kort, E., De Bra, P.: Design Issues for General-Purpose Adaptive Hypermedia Systems. Proceedings of the ACM Conference on Hypertext and Hypermedia. Denmark (2001) 141-150