Domain Modelling To Support Educational Web-based authoring

J.I.Mayorga, M.F.Verdejo, M.Rodríguez, M.Y. Calero

Departamento de Ingeniería Eléctrica, Electrónica y de Control, Escuela Técnica Superior de Ingenieros Industriales, Ciudad Universitaria, s/n, 28040 Madrid, Spain

E-mail: {nmayorga, felisa, miguel, ycalero}@ieec.uned.es
URL: http://sensei.ieec.uned.es/~steed/

Abstract

This paper describes an approach to web-based authoring of educational material. We define a model for the class of subjects of our interest (those including both theoretical and practical issues). From this model, specific content outlines can be derived as subclasses and then instanced into actual domains. The last step consists in generating interactive documents, which use the instanced domain. Students can explore these documents through a web browser. Thus, an interactive learning scenario is created. This approach allows reusing and adapting the contents to a variety of situations, students and teaching purposes.

1. Introduction

Our main purpose is to develop educational activities through the World-Wide Web. That means creating a series of interactive documents that convey the subject matter in a proper way to our distance learners. The technology can help to make this work easier in two ways. On the one hand, it allows a broader, faster and more effective spreading of this material which, in turn, enhances its usefulness for teaching at a distance. On the other hand, it facilitates adapting and reusing the resulting material to a variety of situations and for a range of didactic purposes. Nevertheless, we still keep the usual design metaphor, thus making smoother the change to a technology-driven framework for teachers and learners (see [6]).

In Open Universities, the rate between students and teachers is quite often a big figure. This fact prevents an individual tuition leaning on human interaction. Therefore, we have set the emphasis in developing a richer and more interactive material rather than copying the usual model based on lecturing.

Technology alone can not cope with these goals. It is to be laid upon a solid ground to get the most out of it. Two well-known principles that we have taken into account are the level of abstraction and reusability. They have been widely proposed, for instance, in [3], [7] and [2]

from the Instructional Design perspective, [1] from the educational multimedia viewpoint and [5] from the software engineering field. Following this line of thought, our idea is to have a domain model separated from the media technologies that will be used to navigate through it. In fact, we set up a taxonomy of domain models from the most generic description to the one that has the actual contents to be learnt. The rationale is to have a description for a class of subjects (a metamodel) which can be refined into a number of subclasses that represent specific domains (domain models). Each subclass can be instanced to hold the actual and usable contents.

Structured documents addressing particular teaching purposes can be created from these domain instances. These documents include text and a variety of references to objects within the domain model (such as topics, problems, descriptions, hints or common misconceptions among others) which hold the appropriate information. We claim this approach to be both flexible and suited for developing educational activities through the Web.

2. The metamodel as a generic description of information

A metamodel is a high-level generic description of information that draws a class of subjects. This representation can be refined into any number of domain outlines and is shaped in terms of entities and relationships between them. Thus, it defines a class of models that set up the entities and relationships needed to specify a particular knowledge field.

Metamodels contain a number of domains that are also types of models that have their own specific entities and relationships. For each domain, the metamodel has its list of entities and their attributes, which stand for their properties. A domain can incorporate entities that belong to other ones as well as adding some extra attributes to them.

Hence, defining a domain means stating its entities and the relationships that can be held between them. The first ones characterise the objects which are relevant for the subject matter to specify. Any two entities will differ in the values of their respective attributes. The second ones hold the connections between those objects and can be divided into two classes. Structural relationships allow setting up taxonomies between entities (such as class-subclass or meronymy). They hold the transitive property, which will allow making inferences through the object taxonomy. As a common rule, there will be, at least, two structural relationships: *type-of* and *part-of*. Domain-specific relationships have a meaning associated only to a particular subject and hence they carry their own semantics.

As a metamodel gathers the commonalities of its derived models, it sets the pattern from which all of them inherit their shape. Thus, its usage is, mainly, to be instanced to create a domain model. Nevertheless, it could be useful also for spreading any necessary modifications among its subclasses. The possibility of changing the type of objects and relations at the metalevel makes building up and maintaining the domain easier.

A metamodel is also an information model. As such, it allows answering complex queries to the domain model. This can be done by supplying the linking data between them. For instance, the metamodel permits retrieving all the relationships that have a particular object as

their consequent. Finally, it is also useful for checking the data consistency of the models and their instances while the information is entered or modified.

As an example and study case we have created a metamodel for defining subjects in the Computer Science field. Those subjects share the feature of having a formal corpus of theoretical topics which should be applied to solve practical problems. We have used three domains to describe that metamodel: the *conceptual* domain holds the contents to be learnt. The *instructional* domain includes the objects which will be used to perform the actual teaching and learning processes. The *didactic* model allows relating the other ones. Its elements are actually references to objects that belong to the other domains to which some new attributes are added. These extra properties permit assessing if the objects suit a particular didactic purpose.



Figure 1. *A view of the metamodel for a subject on Computer Science*

The three referred domains are shown in figure 1. An author could see them through a web browser (notice that there are three different web pages in the figure). For each domain, there are a number of entities and relationships defined in tables. Each object is given along with its

attributes. Furthermore, a subset of the relationships is listed stating which entities are connected by each relation. Thus, for instance, there are two conceptual entities (*concept* and *activity*) which have attributes *name* and *definition*. The didactic relationship *prerequisite* can be held between didactic concepts and activities (*Concept_D, Activity_D*). Those are the same conceptual entities but viewed from the didactic domain (and so have the extra attributes *difficulty* and *acquisition_level*). Both entities can be either antecedent or consequent for this relationship. In the instructional domain, for example, we can see that a problem can have subproblems through the structural relationship *part-of* or that a solution can be linked to a *multiple-choice-question* or a *problem*.

3. Models

A model is an instance of the metamodel and therefore a class derived from it (which is its superclass). It defines a knowledge structure which will be filled with actual data as the model is instanced. It includes a description of a subject matter in terms of elementary units and their relationships. Those constituents have their own meaning and purpose as objects within the context of a specific domain.

The domain model supplies an explicit characterisation and a flexible access to the contents which could be retrieved in a variety of ways. They are also defined to be reusable and therefore to ease the creation of new domain models and didactic materials. A library of generic models providing a range of ontologies-which could be adapted to a wide variety of subjects-would help authors and is being created.

4. An instance of a model: Algorithm Design and Verification.

As an application example of the approach described before we have developed a course on Algorithm Design and Verification. The goal for the students of this subject is to be able to write small programs and to prove their correctness. The course includes a theoretical corpus of topics and techniques that are to be applied to build those algorithms. Hence, theoretical issues are the grounds on which the applied knowledge is based in this practical course. In order to cope with these two perspectives, we have defined two different domain models (conceptual and instructional ones) and a didactic layer which stays upon them, following the metamodels shown in section 2.

4.1 The Conceptual Domain model

This model collects the theoretical corpus stated before. It covers a number of topics dealing with both, declarative and procedural knowledge.

There are two kinds of objects belonging to this domain: concepts and activities. The concepts represent the topics stated as declarative knowledge. They describe the basic vocabulary of the field providing accurate definitions for every relevant term. Examples of concepts are *specification, precondition, postcondition, predicate* or *variable.*

The activities represent the procedures that are to be learned and then performed by the students. They take a number of concepts and produce a result, which will be another concept. Examples of activities are *specify, derive, verify* or *prove*. Examples of how activities and concepts relate to each other could be "To *specify* means taking a sentence by enunciating the requisites and desired results for an algorithm and producing a formal *specification* that defines them" or "*derive* means building an algorithm that satisfies its (formal) *specification*". The attributes of both entities are their *name* (a unique identifier) and their *definition* (a rather formal text that describes the topic).

The information model for the conceptual domain also includes a variety of relationships. Structural relationships (*part-of, subactivity*) make up taxonomies for the concepts and the activities. For example, "a *postcondition* is *part-of* a *specification*" or "*proving* the *partial-correctness* (of an algorithm) is a *subactivity* of *verifying* (the algorithm's correctness)".

There is a range of domain-specific relationships which define the semantics of this subject: *Belong-To, Induce, Produce* or *Apply-To*. For instance, a Precondition *Belongs-To* a Specification, a Predicate *Induces* a Set-of-States, (To) Specify *Produces* a Specification (as its result) or (To) Verify *Applies-To* an Algorithm. None of those relationships have attributes, being just connections between related entities.

We can ask this domain for the entities ("which [concepts | activities] are there?") as well as for the relations between them ("what is the *type* of the [entity which name is $name_i$]?", "what is the class of the [entity (which name is $name_j$)]? (i.e., "which entity is [the entity which name is $name_j$] *part-of*?)" or "which concepts does *induce* the [concept which name is $name_k$]?")

4.2 The Instructional domain model

The actual learning of the topics that belong to the conceptual domain model requires a range of objects which are the contents of the instructional domain model. These objects include those ones used for illustrating concepts, practising procedures or evaluating the student knowledge.

The information model for the instructional domain uses three kinds of objects: explanatory, exploratory and evaluative ones.

The explanatory objects contain complementary information to help explaining a given topic. If there was such a type of information available for a topic, one of these objects could be associated to that particular concept or activity. Those explanations can play a number of different roles depending on their instructional purpose.

Some explanatory objects can be associated to conceptual domain objects (as in the case of the *examples* or the common *mistakes*). Some others relate to other instructional objects (for instance, the *hints*, which could be associated to problems, or the *explanations*, which can be associated to any instructional object). Furthermore, there are instructional objects that can accompany any object within the domain (as the *descriptions*).

The exploratory objects allow the student to navigate through the domain and to practise the procedures that she should learn to apply. For the sake of convenience, we have just one type of such objects: the *problem.* A problem has as attributes: *name*, *question*, which enunciates the exercise to be solved, and *location*, which refers to a printed collection made available for our regular students.

Finally, the evaluative objects allow the student to self-assess her proficiency on the domain. A *problem* can also be an evaluative object; in that case, the student would not be offered the answer in advance. Besides, there are *multiple-choice questions* which represent small problems with a number of possible answers. The student has to choose just one option as the correct solution to the question. Those objects attributes are: *name, question, number of options, list of options* and *correct answer.*

The relationships that connect the instructional entities are *Part-Of* (which links, for instance, a problem to its subproblems) and *Is-Solution(*which relates a problem to one solution).

An author can ask the instructional domain about the different attributes of its objects (for instance, "what is the *question* of the [problem which name is $name_w$]?"). Furthermore, it is possible to retrieve objects that hold given properties (for instance, "obtain a set of n *multiple-choice questions* having m options each one"). The relationships allow retrieving the subproblems or the solutions of a given problem.

4.3 The Didactic domain model

This model holds the information about the didactic usage and quality of the entities belonging to the other domains. Therefore, it represents a meta-layer over those ones. Its elements are entities which include a reference to conceptual or instructional objects. These units add a number of didactic attributes to those entities.

There are two kinds of didactic entities, related to either the conceptual or the instructional entities. The former ones add, as new didactic attributes, the *difficulty* of the entity and the *acquisition level* (a measure of the importance of the entity, as a part of the learning process, to the student). The latter ones add the *difficulty* to the instructional attributes that the entity already had. The didactic relationships can be split into three categories depending on the domains that they connect.

The conceptual entities can be connected by means of the *prerequisite* relationship which shows conceptual dependencies between its subject and object. If a concept is prerequisite of another one, the former is to be studied before the latter one.

The instructional entities can be related by means of two didactic relationships: D*escribe-I* and *Explain-I*. *Describe-I* links a description or example to a problem. *Explain-I* connects a hint or an explanation to a problem. This relationship has an attribute called *role* which shows the intended purpose of the explanatory object. The allowed roles are *focus-on, clarify, choose, discard, illustrate* and *reformulate*.

Finally, there are relationships that connect entities belonging to different domains. *Involve* lists all the conceptual elements associated to an instructional object. *Describe-C* relates a

description to a conceptual entity, having the attributes *role* (with the same domain than in the case of *Describe-I*) and *pertinence*(which is a central measure of the relevance of the description for the conceptual entity). *Exercise* links an exploratory object to any conceptual object, thus showing that the latter can be practised by means of the former one. And, finally, *Evaluate* connects an evaluative entity to a conceptual one. It allows assessing the knowledge of the student concerning that concept or activity.

The model for the didactic domain allows retrieving data such as "the prerequisites of a concept", "the available hints to solve a problem", "is there any available reformulation for a given problem?" or "what is the reason for using that particular solving method").

5. Structured Documents

The intended usage of our approach is generating a series of electronic, interactive documents that help the students to learn a subject. We have created a variety of document types described by means of Document Type Definitions or DTD's. A DTD contains the document structure and defines learning paths. The idea is to separate the contents from the document structure to make the authoring easier. Defining a DTD library also facilitates the author's work by providing her with a range of possibilities suited to the didactic necessities for that subject. We have developed a SGML-based language called PALO (see [4]) that allows instancing those document types and making references to domain objects. The instanced document is then compiled, the references filled in with the actual objects and then a complete HTML document is obtained. The result is an interactive learning environment that can be viewed and navigated through a web browser.

The documents can be suited to particular teaching purposes and the domain objects can be used along a wide range of different documents. Furthermore, authors can benefit not just from the available material but also from the ways in which it is described.

Furthermore, these documents may include specifications to trace the students interactions. This would allow creating student profiles by taking into account their performance while they carry out different learning activities.

Examples of documents are self-evaluation tests, programming projects or study guides. The tests help the students to monitor their learning process. The programming projects are structured as questionnaires that the students have to work out and then submit. Study guides propose a tour through the subject matter. They are organised into sections following the topics of the regular course design. They include, for each and every of their sections, a list of the most relevant topics with their descriptions and recommended readings, related exercises, common misconceptions, questions to think about, or some directions to organise their study.

For instance, in figure 2 part of the study guide for a subject on Program Design and Verification is shown. On the left-hand side frame, in italics, there appear the sections (formal verification in this case). A section has links to its contents, a description of its usage, its main concepts, related exercises, common misconceptions, questions to think about and recommended organisation for studying it. In right-hand side frame of the figure, the subsection dealing with the main concepts has been open. Furthermore, the other browser window shows a link to the definition of one of those concepts, a linear recursive function.

This definition includes a short description and a formula which depicts a generic skeleton for such a kind of functions.

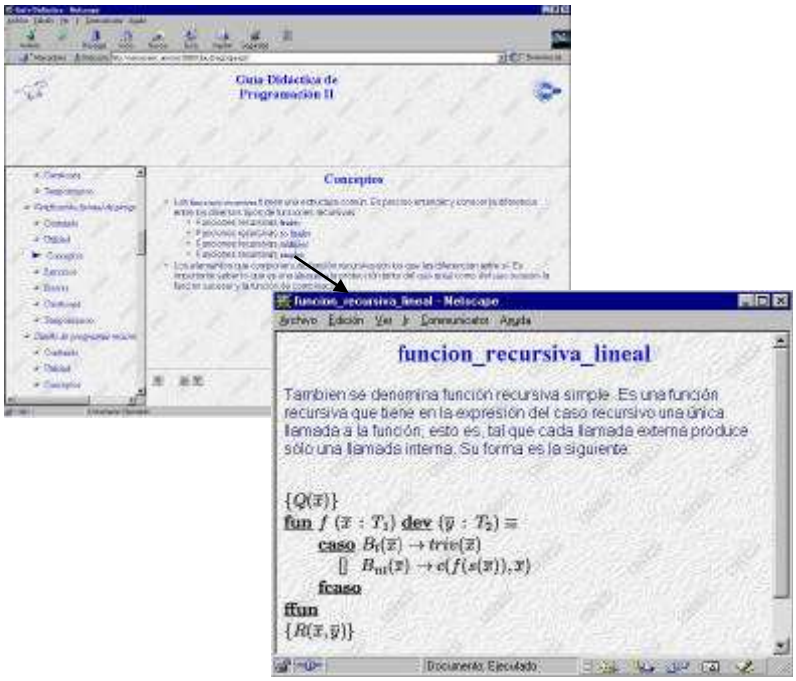


Figure 2. *A structured document in use: the study guide*.

6. Conclusions

Our environment is a traditional Open University where the high rate of students per teacher prevents an individual tuition. Our goal is improving the support to the students and setting the way we carry it out up to date. Hence, we bear in mind the introduction of the new information technologies into our set of basic teaching tools. In order to achieve it, we are developing ways of providing interactive learning support for our regular students. The technology provides the students with the accessibility and the adequate learning material independently of their geographical location.

Our approach supports the Web educational authoring process. It provides the author with a wide amount of teaching material and a flexible way to use it. Furthermore, it makes that process easier by separating the contents from the document structure and releasing the author from the HTML encoding burden.

The domain description is based on a conceptualisation work that has allowed to create a taxonomy of domain models. Modelling means defining the domains in terms of relevant and useful objects that can be retrieved in a number of ways. We first created a metamodel from which domain models can be instanced. Those domain models can be filled with the actual subject matter contents. Document types addressing particular teaching purposes can be created. Those documents define structured templates and learning paths. The author can add references to domain objects within instances of those document types by using a high level language. Finished documents are produced by compiling those instances. During that

compilation, objects are retrieved from the domain representation. Therefore, it is possible to create many learning scenarios sharing the same objects but having different teaching purposes and didactic perspectives. Facilitating reusability is a way of reducing the cost of developing web applications to teach at a distance. It also helps the authors get the most out of the materials they create. These documents have a fixed part of text and a variety of references to objects (topics, problems, descriptions, hints or common misconceptions among others) which hold the appropriate information.

7. Acknowledgements

This work has been partially funded by the CICYT (Spanish Research Agency) under project number TEL97-0328-C02-01

M.Y. Calero has a grant from the "Programa para la incorporación de técnicos a equipos de investigación científica" of the Comunidad de Madrid

8. References

[1] MAGLAJLIC, S., MAURER, H. and SCHERBACKOV, N. (1998), Separating structure and content, authoring educational web applications. In Proceedings of the ED-Media & ED-Telecom 98, pages 880-884.

[2] MAYORGA, J.I. AND VERDEJO, M.F. (1996), Authoring systems revisited: the software design cycle metaphor. In Paiva, A., Brna, P. and Self, J., eds., *European Conference on Artificial Intelligence in Education.*

[3] MERRIL, M.D. (1987), The new component design theory: instructional design for courseware authoring. *Instructional Science, 16:19-34.*

[4] RODRÍGUEZ-ARTACHO, M., VERDEJO, M.F., MAYORGA, J.I. and CALERO, M.Y. (1999), Using a high-level language to describe and create web-based learning scenarios, submitted to *IEEE Frontiers in Education Conference.*

[5] SCHWABE, D. and ROSSI, G. (1995), Object-oriented hypermedia design method. *Communications of the ACM, 8 (8):45-46.*

[6] VERDEJO, M.F., RODRÍGUEZ-ARTACHO, M., MAYORGA, J.I. and CALERO, M.Y. (1999), Creating web-based scenarios to support distance learners submitted to *IFIP International Working Conference On Building University Electronic Educational Environments.*

[7] WENGER, E. (1987), *Artificial Intelligence and Tutoring Systems: Computational and Cognitive approaches to the communication of knowledge.* Morgan-Kaufman.