CHAPTER 7

THE DESIGN OF ADAPTIVE WEB-BASED LEARNING MATERIALS INCORPORATING THE LEARNINT EXPERIENCE

7.1 INTRODUCTION

This Chapter puts forward a methodological approach for the development of educational AH applications that integrates the evidence gathered throughout the research.

Using the Dexter model as a reference framework, the conceptual structure of AH systems is presented that, building upon current design and development approaches, reconsiders the implications of the research and proposes a design approach that extends the functionality of LEARNINT.

In addition, a number of emerging technologies and approaches are discussed that may further influence the design and development of AH systems.

7.2 EXTENDING THE FUNCTIONALITY OF LEARNINT

Throughout the research the Dexter model [Halasz & Schwartz, 1994] has been used as a principled basis for understanding the conceptual structure of hypermedia systems in general, and of adaptive hypermedia applications in particular. This model is organised into three layers, namely the run-time layer, the storage layer and the within-component layer. Its graphical representation in Figure 7.1 has been further elaborated to present in more detail the main components of each layer that are relevant for the design of educational AH applications.

The run-time layer describes how the application content will be delivered through the user interface, while the within-component layer corresponds to the physical storage of the system's data objects, i.e. the actual files of learning content.

The storage layer represents the basic structure of nodes and links that are the essence of any hypermedia application –i.e. the domain model. A learner model is also included, which is a conceptual representation of all the aspects of the user that are relevant for

the functionality of the system. An adaptation model also exists that describes how the domain model and the user model are combined to generate adaptation.

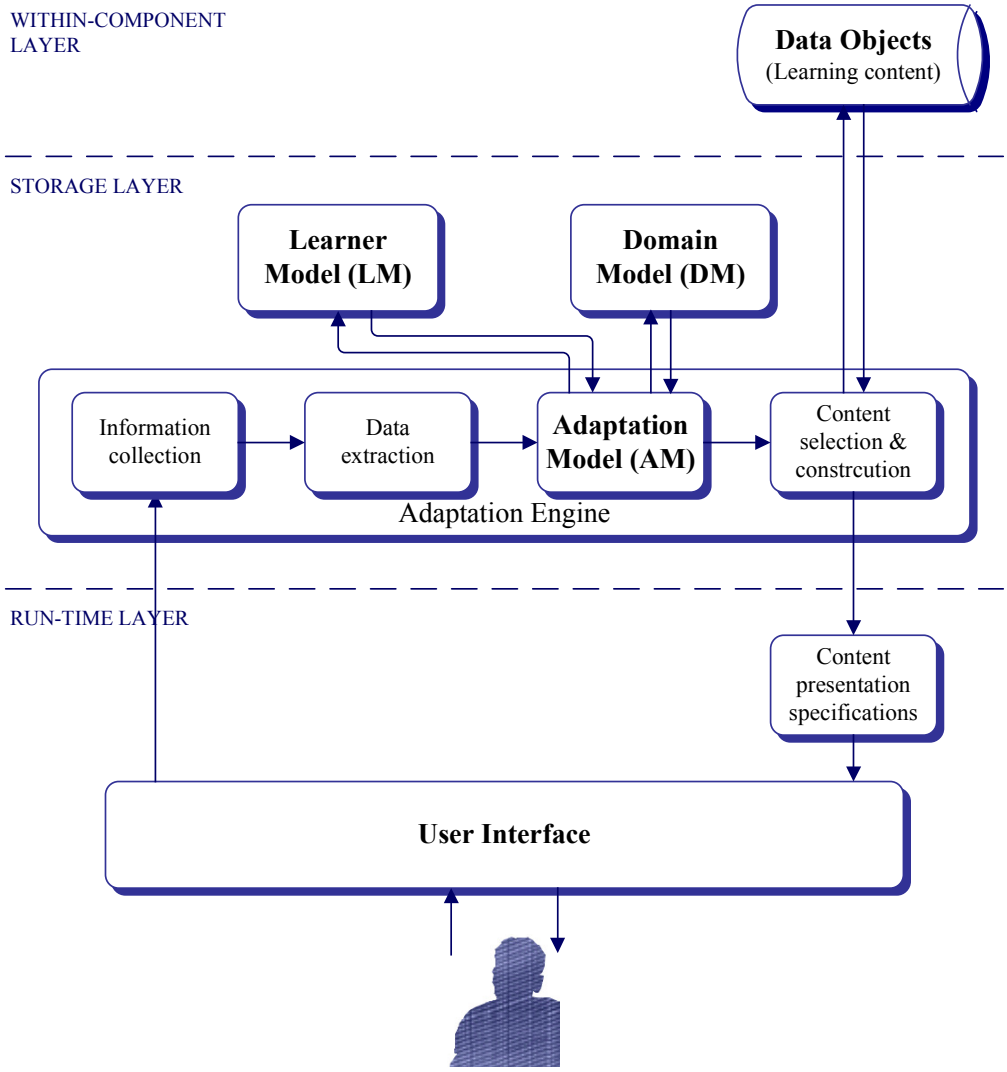

Figure 7.1: A detailed conceptual structure of AH systems based on the Dexter reference model.

Based on the findings of the research and building upon current approaches to the development of adaptive applications, a methodology is put forward that extends the functionality of LEARNINT. Following the structure proposed by the Dexter model the constituent parts of each layer are presented in detail.

7.2.2 Domain Model

According to the Dexter model, a subject domain can be conceptually described in terms of components. Components are abstract entities that either represent domain content or express relationships between content items. Figure7.2 illustrates how the

content of a course can be organised into domain components interrelated through relationships of different kinds.

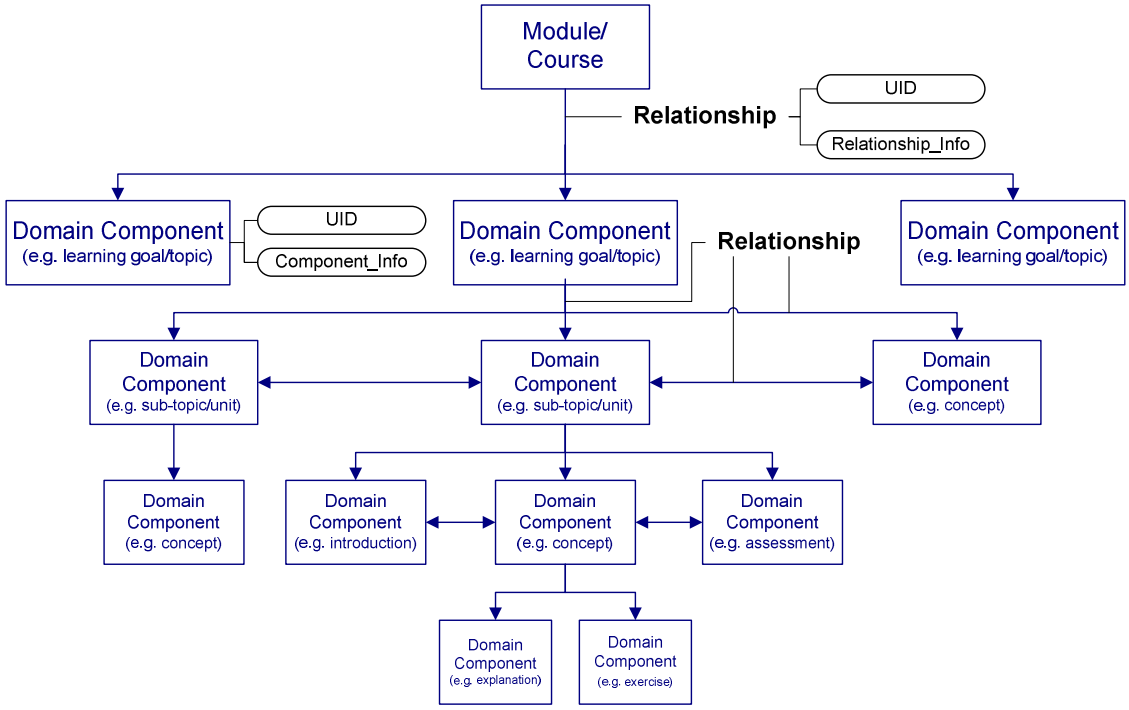

Figure 7.2: Graphical representation of a domain model organised into components interconnected through relationships of different kinds.

Different terms are used across different disciplines and software systems to refer to the constituent parts of the domain model. Educational AH applications such as AHA! [De Bra & Calvi, 1998; De Bra et al., 1999; De Bra et al., 2005] and INSPIRE [Papanikolaou et al., 2003] use *concepts*, while other systems, such as TANGOW [Carro et al., 1999], use *teaching tasks* instead. The term *component* is used here in accordance with the notation proposed by the Dexter model.

A component is an abstract representation of an information item from the subject domain [De Bra et al., 1999]. Components can be atomic or composite depending on the level of granularity of the subject content they represent. When a component refers to a specific piece of learning content, it is referred to as an *atomic* component. When a component is defined in terms of other components, it is referred to as an *abstract* (or *composite*) component. Relationships are entities that represent interconnections between components.

In Figure 7.3 an extract from the domain model of the "Computer Hardware" module (the content used for the LEARNINT prototype) is presented. In this example it is possible to identify a series of atomic and composite components, as well as various

relationships. For instance, the components "Combinational Circuit Design" and "Sequential Circuit Design" clearly are composites since they consist of other components, whereas "Introduction" and "Negative edge-triggered" are atomic components since they refer to specific pieces of content.

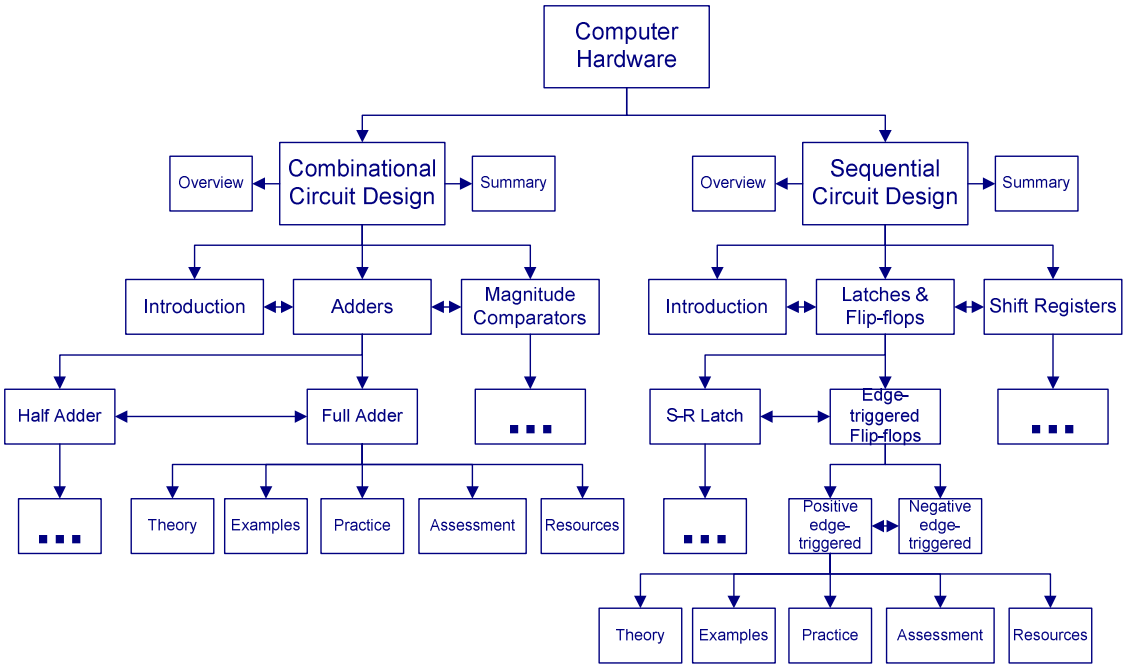

Figure 7.3: Graphical representation of an extract of the domain model for the "Computer Hardware" module.

Arrows between components denote relationships. While in hypermedia systems relationships usually refer to navigational links between nodes, in AH applications relationships may refer, for example, to prerequisites or related concepts in addition to navigational links. Thus, Figure 7.3 may indicate that "Positive edge-triggered" and "Negative edge-triggered" are prerequisite components of "Latches & Flip-flops", whereas the link between "Half Adder" and "Full Adder" may indicate that these are related concepts. The exact nature of the relationship has to be established when the domain model is defined.

Components can be expressed in terms of tuples of the type <*uid, c-info*>, where:

- *uid* is a unique identifier for the component, and
- *c-info* refers to the component information, which consists of:
  - a set of <*attribute, value*> pairs
  - a sequence of sub-components

Uid's are individually assigned across the entire system and give each component a unique identity. The set of *attribute-value* pairs is used to attach a list of properties to the component (e.g. name, type or content description). Furthermore, this list of attributes has to include all the variables that will be used to generate adaptation during runtime; in educational applications for example, it is common to include a "knowledge" attribute to store the system's idea about the level of knowledge a student has about the learning content of a component. In the case of composite components, the component information (*c-info*) also includes the list of other components of the domain model.

Relationships are also treated as components within the Dexter model. These include a sequence of *specifiers*, each of which corresponds to a component, in the form <*uid*, *dir*>, where:

- *uid* is the unique identifier of a component, and
- *dir* is the direction of the relationship, e.g. FROM, TO, BIDIRECT, etc.

Follow on the example illustrated in Figure 7.3, if a relationship of the type "prerequisite" is defined between "Full Adder" and "Adders", such that the first component is prerequisite for learning the second one, the *specifier* for "Full Adder" might be expressed as <*FullAdder.uid*, *FROM*>, and the *specifier* for "Adders" as <*Adders.uid, TO*>.

Thus, a relationship can be expressed as <*uid, ss, c-info*>, where:

- *uid* is the unique identifier of the relationship,
- *ss* is the sequence of *specifiers,* and
- *c-info* refers to the properties of the relationship, and consists of a set of <*attribute, value*> pairs

The *uid* assigned to each relationship is individually determined across the entire system. The set of *attribute-value* pairs is used to attach a list of properties to the relationship, such as type of relationship between components.

In practice, the implementation of the domain model is carried out differently across different systems. TANGOW [Carro et al., 1999], for example, defines its teaching tasks using a database (see Figure 2.8); whereas in AHA! [De Bra et al., 2005] the domain model is described in terms of concepts and concept relationships (see Figure 2.6). However, their underlying structure is similar to that described above: a series of

components and subcomponents that are interconnected through various types of relationships. Unique identifiers are assigned to each component, as well as additional information that translates into or is used by adaptation rules at runtime.

Incorporating the LEARNINT Experience in the Design of the Domain Model

Designing the domain model is one of the most difficult tasks in the process of developing educational AH applications. On the one hand, planning the content requires to carefully determining what concepts of the subject domain to focus on. Planning the delivery, on the other hand, involves making decisions about how to present the concepts previously identified.

According to the findings of this research, to provide a flexible **sequence of instruction**, different types of learning materials/tasks are required, such as those supported in the domain models of INSPIRE [Papanikolaou et al., 2003] and TANGOW [Carro et al., 1999] – i.e. theory, examples, exercises, assessment, etc.

In addition, learning materials have to be available using different media elements in order to provide the most suitable **mode of presentation** according to the current state of the learner model.

Also, to support students in understanding the **structure of the content**, advance and post organisers are suggested in the form of overviews and summaries for each component of the domain model. Additional support for the students can be provided in the form of complementary resources and links to related concepts within the system and elsewhere.

Figure 7.4 exemplifies how the domain structure for the component "Full Adder" might look like following the previous guidelines. This component includes various types of learning content such as theory, examples, practice, assessment and resources. These can be presented following different sequences (e.g. examples first and then theory) according to the characteristics of the current student as reflected in their learner model. Each sub-component can point to more than one file, meaning that different modes of presentation exist for the same content (e.g. verbal or imager content). In this example, an overview and a summary for the "Combinational Circuit Design" component exist; additional resources are also included, and lists to related topics could be easily added.

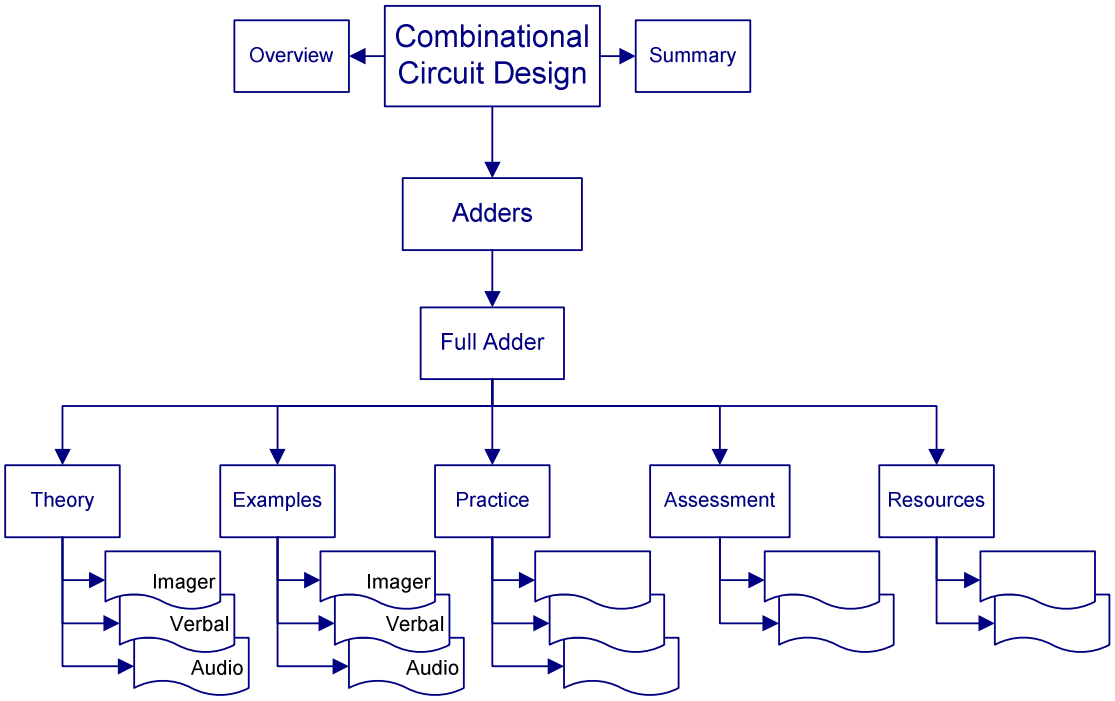

Figure 7.4: Domain model structure for the "Combinational Circuit Design" topic.

Closely related to the design of the learning content is the design of its corresponding assessment. Achieving a series of desired learning outcomes requires not just determining the suitable learning content and planning its delivery, but also to carefully determining the appropriate assessment tasks [Biggs, 2002; Osborne, 2004].

For educational AH systems two major implications of this alignment process refer to the provision of differentiated feedback and support, and to the adaptive behaviour of the system based on the students' performance. To provide differentiated feedback, an application would require a range of examples and exercises with various levels of complexity. Then, approaches such as "Prerequisite-based help" and "Intelligent problem solving" could be implemented – as in ELM-ART [Brusilovsky et al., 1996] and INTERBOOK [Brusilovsky et al., 1996]. Adaptive tests and feedback can also be implemented following the strategies proposed by CAT and associated IRT algorithms [Jettmar & Nass, 2002; Lilley et al., 2004; Lilley & Barker, 2005].

The assessment strategies also impact the general adaptive behaviour of a system. As it has been argued, adapting the interface for Web-based learning materials has to be based on both the user preferences and their performance. Therefore, the kind of assessment tasks included in the system and the way in which the student's learning is registered and used are very important and have to be considered for the design of the domain model.

7.2.3 Learner Model

In educational AH systems the learner model is commonly based on the components of the domain model plus a series of attributes such as *read* and *knowledge-level*, which respectively indicate whether the user has read the content included in a component, and how much the user knows about it. These attributes are used in Table 7.1 to illustrate how a learner model may look like.

Concept uid	Concept name	Read	Knowledge-level
uid01	Combinational Circuit Design	true	Learned
uid02	Adders	true	Learned
uid03	Half-Adder	true	well known
uid04	Full-Adder	false	not known

Table 7.1: Example of a learner model.

Additionally, the learner model usually contains other concepts to record information that is independent from the application domain, such as the students' name, age or preferences.

While the use of a table is helpful to illustrate the possible structure of a learner model, its actual implementation is carried out differently across different applications. TANGOW [Carro et al., 1999], for example, uses a database to store all the characteristics required from its users, while in AHA! [De Bra et al., 2005] information is stored using XML files.

All the attributes of the learner model require an initial value for the system to deliver its learning content. Commonly, most of these properties are initialised as part of the set-up process of the software, either by using default values or by interacting with the users and asking them specific pieces of information. Some systems also have built-in facilities to allow the students to inspect and change the values stored in their learner model.

Some current applications that support cognitive or learning styles as part of their learner model use a specific <*attribute, value*> pair to store this information. Different strategies are then implemented to provide adaptive content based on this property of the learner. In AHA! [Stash et al., 2004] the attributes of the concepts from the domain model are used in combination with the style of the student through a series of rules

specified in the adaptation model; MOT [Cristea & Calvi, 2003; Stash et al., 2004] uses generic adaptive strategies, while in INSPIRE [Papanikolaou et al., 2003] the knowledge level and the learning style of the learner are used as the main sources of adaptation.

Incorporating the LEARNINT Experience in the Design of the Learner Model

It has been argued in the thesis that while the proposed adaptive variables derive from an extensive analysis of key characteristics of cognitive styles, the results from the experimental stage of the research do not support the identification of cognitive styles for classifying students in order to provide adaptive Web-based learning materials. Two further arguments for this standpoint refer to the process of identifying the students' cognitive style and to the dynamic nature of the learner model.

In determining the cognitive style of students, an issue that should be taken into account is that the majority of the instruments used for this purpose are paper-based or computer-based questionnaires and students may see the process of completing a questionnaire as external to their main goal: learning from the learning material itself [Lord, 1998]. As reported by Papanikolaou et al. [2003], if the students are given the opportunity to choose between answering a questionnaire or not, it is likely that more than half of them would choose not to answer the questionnaire.

The second argument is that a critical characteristic of the learner model is its capacity to evolve through time and interaction with each individual user, taking into account their behaviour, performance, expressed preferences and affective reactions. Therefore, rather than applying a stereotype depending on the value stored in the "cognitive style" property of the learning model, the approach suggested is to have a series of variables that can be combined and re-combined according to the changes identified in or expressed by the individual learner. While the number of variables and their possible combinations still represent a finite number of options for the users, this approach is considered more flexible and closer to their individual requirements and preferences.

According to the findings from the research, the learner model should contain information about the student in terms of:

- **sequence of instruction** – Do they prefer theoretical content before examples?, Do they perform better with a more practical approach (i.e. starting with some exercises)?

- **content presentation** – If the same content is available in more than one mode of presentation, what would the student prefer: verbal content, graphical content, audio?
- **structure of the content** – Do they find outlines useful?, Do they prefer having an overview at the beginning of a new concept, a summary at the end of it, or none of them?, Do they prefer the information to be presented using frame arrangements or overlapping windows?
- **control strategy** – Do students prefer moving freely around the learning content or having a fixed path through the material?, Do they find annotation of links useful?

While the initial value of these variables can be set by different means –e.g. using default values, or asking the learner through a registration form, the system should have the appropriate mechanisms for recording the user's behaviour –e.g. reading order of different types of media – and combine this information with their learning performance and affective reactions to infer the most convenient way of adapting its learning content.

7.2.4 Adaptation Model

The basis for the adaptive functionality of AH systems is found in their adaptation model, which combines information from the domain model and the learner model to determine how the content is presented.

Generally, the adaptation model consists of a series of adaptation rules which are described as *event-condition-action* rules. When a rule is activated by an *event*, its *conditions* are evaluated. Conditions are Boolean expressions that use attributes from components in the domain model or the user model. A condition, for example, may refer to whether the "read" attribute of a component is true or not –i.e. whether the content of a component has been read by the user. If the conditions stated in the rule are satisfied, an *action* is executed. Actions commonly update the values of one or more attributes of the user model, which in turn may determine how the learning content is delivered to the student.

Most adaptive rules are thus conditional rules which take the following form [Calvi & Cristea, 2002; Cristea & Calvi, 2003]:

```
If <Prerequisites> then <Action>
```

Some examples are presented below that may help to clarify how rules are used within AH applications. The following rule indicates that when a component *C* is *accessed*, the learner model attribute called *read* is set to *true*:

```
<access(C) => C.read := true>
```

The next rule states that when a component *C* is *ready-to-read* and is *accessed*, the value of the *knowledge* attribute is set to "*well-learned*".

```
<access(C) & C.ready-to-read=true => C.knowledge:= "well-learned">
```

As in the case of the domain and learner models, the actual implementation of the adaptation model is carried out differently across different applications. In TANGOW [Carro et al., 1999] for example, adaptation rules are used to describe how a teaching task is divided into subtasks (Figure 2.9 presents an example of this approach).

In AHA! [De Bra et al., 2005], different relationship types between components convey different adaptation rules. A relationship of type "*prerequisite*", for example, would state that a component *C* is ready to be learned only when its prerequisite content *Cp* has been well-learned. Generically, this can be expressed as:

```
<Cp.knowledge="well-learned" => C.ready-to-read:=true>
```

An example of the actual syntax of adaptation rules in AHA! is presented in Figure 2.7.

Some authors [Calvi & Cristea, 2002; Cristea & Calvi, 2003] argue that although this approach is very flexible, it may lead to difficulties in the definition of relationships. Based on the generic conditional rule – `If <Prerequisites> then <Action>` – they have proposed a set of adaptation rules that can be applied at the component level rather than at the relationship level.

- *Level* rules - an action is executed if a number of prerequisites are fulfilled:

  ```
  If enough (<Prerequisites>) Then <Action>
  ```

  For example if `Prerequisites = time_spent, and Action = "go to next level"`, the rule becomes:

  ```
  If enough (time was spent on current level) Then "go to next level"
  ```

  where `enough` may be defined as:

  ```
  enough (advanced topic) = 10 time units;
  enough (medium topic) = 5 time units;
  enough (beginner topic) = 2 time units;
  ```

- *Temporal* rules – certain action is repeated as long as one or more conditions hold:

`While <Condition> Do <Action>`

- *Repetition* rules – certain action is repeated for a predefined number of times:

`For <i=1 … n> Do <Action>`

- *Interruption* commands – the action the user is performing is interrupted:

`Break <Action>`

- *Generalisation* commands – show the learner a more general concept compared to the one currently being studied:

Generalise (Cond, $Cond_1$,…, $Cond_n$)

- *Specialisation* commands – show the learner a more specific concept compared to the one currently being studied:

Specialise (Cond, $Cond_1$,…, $Cond_n$)

These rules allow authors to write simple rules instead of complex ones, and since they have been derived from the Genetic Graph [Goldstein, 1982], these rules have the advantage of modelling the evolution of the user's knowledge (abstraction, exemplification, generalization, etc.), which in turn can help to determine several ways in which the domain components can be interconnected.

Furthermore, a three-layer approach has been proposed to represent the dynamic behaviour of AH systems [Cristea & Calvi, 2003]. The highest level allows the definition of adaptive strategies corresponding to instructional strategies which, in principle, can be applied over different domain models. Taking Kolb's *Converger* learning style as an example, Table 6.3 shows the definition of the corresponding adaptive strategy and its translation into the lower layers of this approach.

This approach has been implemented in MOT [Stash at al., 2004], where generic strategies can be defined for different cognitive styles. An adaptive strategy can use, for example, the *generalization* command to present the learner with more general (and easier) concepts if their assessment results were poor; if the results were good, then the strategy would use *specialise* rules instead.

LEVEL OF ADAPTATION	CONVERGER COGNITIVE STYLE EXAMPLE
High level *Adaptation Strategies* Goal oriented wrapping layer	*Convergers* are abstract and active; they like to feel in control, start with content for intermediates at medium adaptivity level; repeat for a number of times: -evaluate state of learner and start increasing difficulty & decreasing adaptivity level of result = good -evaluate state if the learner and start decreasing level if result = bad

Medium level
Adaptation Language
Goal/domain oriented adaptation techniques that embrace low-level adaptation rules
	AdaptLevel=5; N=AskUser(); //control levels: 1=min to 10 =max
	`For <i=1 … N> Do`
	`{`
	`  Specialise(Enough(Result));`
	`  If (AdaptLevel > 1) AdaptLevel--;`
	`  Generalise(Not(Enough(Result)));`
	`  If (AdaptLevel < 5) AdaptLevel++;`
	`}`

Low level
Direct Adaptation Rules
Techniques usually based on threshold computations of attribute-value pairs
	DiffLevel=3; AdaptLevel=5; //No predefined number of repetitions
	`IF <Action> Then`
	`{`
	`  If(Result1+Result2)/2 > 5 && DiffLevel<10 Then`
	`  {`
	`  DiffLevel++;`
	`  IF(AdaptLevel>1) AdaptLevel--;`
	`  }`
	`  If(Result1+Result2)/2 < 5 && DiffLevel>1 Then`
	`  {`
	`  DiffLevel--;`
	`  If(AdaptLevel<5) AdaptLevel++;`
	`  }`
	`}` //Note that ”Enough” and Specialise must be redefined each time

Table 7.2: Three layer model for adaptive strategies.
An example of an adaptive strategy for the *Converger* cognitive style.
[Adapted from Cristea & Calvi, 2003, pp. 5, 9]

Incorporating the LEARNINT Experience in the Design of the Adaptation Model

While the concept of adaptive strategies has been applied in MOT for implementing instructional strategies associated with specific learning styles, their scope is not limited to such an approach. Adaptive strategies could be defined to monitor the interaction between learners and the AH system and to take into account their preferences, learning performance and affective reactions in order to provide the most convenient configuration for each learner. Used in this way, adaptive strategies would be in line with the results of the research because, as has been argued before, results from the experimental stage do not support the sole use of cognitive styles for the provision of adaptive Web-based learning materials.

Adaptive strategies could be defined for the continuous tuning of the various variables proposed in the research. To provide flexible **sequence of instruction**, for example, adaptation strategies might be defined to evaluate learner performance and determine whether a different sequence of learning tasks would be more advantageous. A basic, high-level definition of this strategy would look as follows:

```
Evaluate state of learner
    If learning performance is not satisfactory
        If current sequence of instruction != learner's preferred
        sequence
            Suggest a different sequence of instruction
```

The sequence of presentation would depend on the structure of the domain model and based on a suitable curriculum sequencing algorithm to determine what types of learning tasks are presented and in what order.

The adaptive strategy for selecting the suitable media elements to present the content identified through curriculum sequencing –i.e. **content presentation** – would be closely related to the structure of the domain model, the availability of different types of materials and the current state of the learner model in terms of learning performance and expressed preferences:

```
For each learning task selected through the curriculum sequencing
strategy
    Evaluate the state of the learner
        If learning performance is satisfactory
            type of media elements = learner's preferred content
            presentation
        else
            suggest different presentation of content
    Locate media files for assembling learning content
```

The provision of structural aids such as content outlines, link annotations, and advance and post-organisers would depend mainly on the preferences expressed by the students and their recorded learning performance.

```
Evaluate state of learner
    If learning performance is satisfactory
        If learner's content outline = activated
            Include content outline
        If learner's advance organiser = activated
            Include overview of current learning content
        . . .
```

Adaptive rules of the type *specialisation* and *generalisation* can be implemented for determining the links to background and advanced concepts. Learning content would then be assembled and presented according to the interface's look and feel preferred by the user – frames, simultaneous windows, use of menus and toolbars, verbal or visual elements, etc.

To allow for different degrees of **student control** over the adaptive behaviour of the system, a variable controlling the adaptivity level can be implemented, which may be used in combination with the adaptive strategies defined for other variables; for example, in order to give the students more flexibility to follow their chosen path through the content, the adaptivity level could be decreased when the content outline is activated:

```
. . .
    If learner's content outline = activated
        decrease adaptivy
        include content outline
```

Other navigation features such as maps and buttons would be available to the extent to which these correspond to the level of adaptivity in place. However, the students should always have the possibility to activate or deactivate the adaptive functionality of the system and to modify (customise) the presentation of the learning content.

While the adaptive strategies proposed above have been stated in simple, high-level statements, the actual adaptive strategies would be closely intertwined with each other and their implementation would require their translation into low-level adaptive rules; therefore, their design and implementation would require careful planning.

Of particular relevance would be the assessment strategies in place. While suitable assessment tasks should had been defined as part of the domain model, the adaptation model might include a formative assessment strategy that registers the learners' performance to provide appropriate feedback and determine whether further assessment tasks have been carried out. In turn, the learners' performance recorded as a result of summative assessment would be used in most adaptive strategies in combination with the learners' preferences for determining possible changes to the learning environment.

Registering learning performance (users' knowledge) is carried out differently across different AH applications. In AHA! [De Bra et al., 2005] for example, knowledge level is recorded for each component of the domain model as an integer value between 35 and

100, which is increased when a learner accesses a content page that is ready-to-read. This value is propagated upwards the domain structure based on the prerequisite hierarchical relationships between components. In TANGOW [Carro et al., 1999] every time a learning task is completed by the student, their execution results are recorded (an example of a task rule is presented in Figure 2.9). The task execution results are also propagated up the domain structure to determine the next set of suitable learning tasks and subtasks.

A different approach that integrates the design of the domain model with assessment of learners' performance and curriculum sequencing has been implemented in INSPIRE [Papanikolaou et al., 2003]. The domain model of INSPIRE is organised around different levels of abstraction and differentiated levels of user performance: *remember*, *use* and *find*. In turn, assessment tasks include questions that test the learner's ability using the same levels of performance. Based on the performance of the learner on the different categories of questions, the system makes estimations about their knowledge level using a diagnosis process that exploits ideas from the fields of fuzzy logic and multi-criteria decision making. Learner's knowledge is eventually classified as *inadequate*, *mediocre*, *advanced* and *proficient*. Knowledge level is then used for planning the sequence of the content that is to be presented to the learner. Multiple strategies are included in the system for this purpose, such as:

```
If knowledge level = Inadequate on a number of outcome concepts
    educational material = remember level
        include outcome concepts
        include prerequisite concepts
```

An assessment approach such as this implemented in INSPIRE would seem more appropriate for the design and implementation of the adaptive strategies suggested by the results derived from this research.

7.2.5 Incorporating the LEARNINT Experience in the Design of the Within Component Layer

The Within component layer refers to the physical storage of media elements used to assembly the learning content that will be presented to each learner according to their learner model and the adaptive strategies defined in the adaptation model.

While the domain model is a conceptual structure of the subject domain, its atomic components refer to specific pieces of information that correspond to actual files stored

in this layer. When the appropriate sequence and mode of presentation of the components have been defined, a suitable retrieving mechanism should be in place to construct the actual pages of learning content. An *accessor* function should exist in the adaptive engine of the system capable of mapping the *uid* of the components identified from the domain model to the actual files of media elements stored in the repository.

7.2.6 Incorporating the LEARNINT Experience in the Design of the Run-time Layer

The Run-time layer is concerned with how the learning content will be delivered through the user's interface. Once the **sequence**, type and **media content** of the learning materials have been determined through the interaction of the various models in the storage layer, and the corresponding media files have been retrieved from the within-component layer, the next step is the actual construction of the Web pages presenting the learning content to the students.

Depending on whether the interface is verbal or imager in style, all elements in the screen, such as buttons, menus, toolbars and other navigation aids should follow the same style. Also, the layout of the interface in terms framesets or simultaneous windows has to be defined. The inclusion of content outlines, as well as link annotations depends on the current **levels of control** and adaptivity. Different colour schemes for the various elements on the screen should be also available.

As with the other adaptive variables identified in the research, decisions about the final **presentation of the learning content** in the Web browser of the learner should be based on their learning performance and expressed preferences. As proposed in previous sections the sequence, **mode of presentation** and **structure of the learning materials**, as well as the **level of control** and adaptivity in the system should be based on a series of adaptive strategies that take into account various characteristics of the learner model.

This is however a serious challenge for AH systems. While most adaptive applications provide opportunities for the learners to influence the adaptive behaviour of the system –e.g. through open learner models – and customise features such as the colours used to annotate links, most of them provide a pre-defined user interface that does not change based on the interaction with the learners.

A notable exception is the latest version of AHA! [De Bra et al., 2005], which has introduced a *layout model* to implement a consistent style throughout entire applications developed using this software.

Building upon this development, extended functionality would be required in order to make possible dynamic configurations that can be updated based on the interaction between the system and their users by means of the adaptive strategies embedded in its modelling dimensions. Issues that have to be addressed include the interface style that should be used at the beginning of the interaction with the user; the options that students would have for customising the interface; the co-ordination required between the adaptive strategies in the adaptation model and their implications for the style of the interface; also the technological solution and development that would make this functionality possible.

7.3 AN EXTENDED DESIGN OF LEARNINT

To provide the adaptive functionality described in the past sections, an educational AH system requires a number of additional elements to manage and coordinate the various modelling dimensions underpinning the application.

The first of these elements is a *session manager* administering the number of users connected to the system at any given moment. It is its responsibility to identify whether the current request comes from a user for whom a session is already open, or if the user is trying to enter the system and a new session has to be launched. When a new session is open, it receives an id and is included in a mapping table of open sessions. The session manager is responsible for the administration of this mapping table, which determines who the user is and to whom a response (if any) will be send back. The session manager then passes the request to a *content planner*.

The content planner keeps control of the activities carried out by each learner and oversees the adaptation process. If the request received from the session manager indicates that a new user is trying to register to the system, the content planner launches the registration process and the response to the user's request will be the presentation of a registration form. If the request received from the session manager indicates that the user is already registered, their profile is loaded and the adaptation process is launched. As a result, a series of parameters will be transferred to the *page generator* indicating what content to present.

The page generator is responsible for rendering the actual content that is to be presented to the user based on the parameters received from the content planner; it retrieves the corresponding media elements from the within-component layer, generates the Web

pages required and sends a response to the Web server –i.e. the learning materials requested by the student.

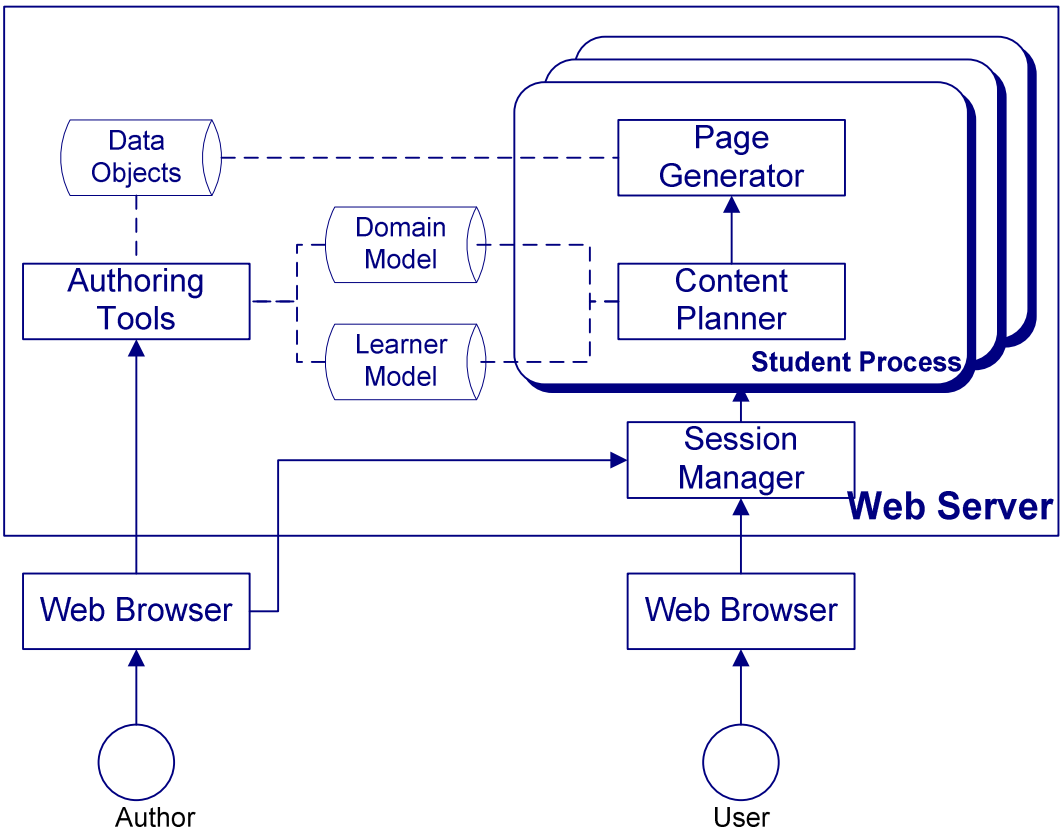

Figure 7.5: Proposed generic structure of educational AH systems.

Figure 7.5 illustrates the structure and interconnections between the components described above. In this structure an additional element refers to the facilities provided by the system for authoring AH applications. The authoring process comprises the design and implementation of the domain model, the learner model and the adaptation rules and strategies of the adaptation model.

7.4 Emerging Issues and their Implications for the Design and Development of Educational AH Systems

A series of current issues and trends are raised in this section which have to be taken into account for the design and development of educational AH systems, including aspects related to the technological platform required to build the system, approaches to open and interactive learner modelling, and some implications of the Semantic Web.

7.4.1 Technology

Most Web applications heavily rely on the technological platform used for their development and deployment, and AH systems are not the exception. The technological infrastructure used limits to a great extent the system capabilities and influences its whole design. While the technologies used in the various educational AH systems referenced in the thesis differ from one project to another, most applications have been implemented using open-source platforms. Some of these platforms are reviewed in this section, as well as emerging approaches that may be considered for improving adaptivity and personalisation of Web-based learning materials.

In terms of Web servers, APACHE[1] seems to be the preferred platform. TOMCAT[2] and COCOON[3] are then utilised over APACHE for publishing content on the Web. While TOMCAT is mainly a servlet container, COCOON is a development framework with strong foundations in XML. Using COCOON it is possible to implement XML files as well as eXtensible Server Pages[4] (**XSP**). XSP has been used in the development of some AH systems to implement the adaptation engine, manage session variables and filter lessons' content [Brown et al., 2006].

In addition to XML documents, **MySQL**[5] is also widely used to store learner and domain information in AH applications. In turn, **PHP**[6] is commonly used for front-end development, especially since it is compatible with many types of databases.

Current trends for improving the responsiveness of interactive Web-based applications include **AJAX**[7] (Asynchronous JavaScript and XML); its asynchronous characteristic means that Web applications are able to make quick, incremental updates to the user interface without reloading the entire browse page. This aims to increase the page's interactivity, speed and usability; it also represents an attractive possibility for improving adaptivity and personalisation of the interface for Web-based learning materials.

---

[1] http://www.apache.org
[2] http://tomcat.apache.org/
[3] http://cocoon.apache.org/
[4] http://cocoon.apache.org/2.1/userdocs/xsp/logicsheet.html
[5] http://www.mysql.com/
[6] http://www.php.net/
[7] http://developer.mozilla.org/en/docs/AJAX

7.4.2 Open and Interactive Learner Models

Open learning modelling has emerged in response to the difficulty of delivering fully automated models based on observational diagnosis. In 1990, Self suggested that diagnosis of learners should be made interactive and collaborative, making the learner model accessible to students and giving them an active role whereby the system would become an assistant, helping them to clarify their beliefs [Self, 1990].

From early educational AH systems such as AHA! [De Bra & Calvi, 1998], to more recent applications such as INSPIRE [Papanikolaou et al., 2003], learners have been allowed to inspect and change certain attributes of their learner models. Moreover, a number of architectures have been proposed with the purpose of making learner modelling an overt process allowing not just inspection but direct participation from the learners. [Bull et al., 1995; Bull & Smith, 1997; Greer et al., 1998; McCalla et al., 2000].

Several empirical studies have investigated different aspects of involving the learners in diagnosis [Bull et al., 1999; Morales et al., 2000; Zapata-Rivera & Greer, 2000, Mitrovic & Martin, 2002; Dimitrova, 2003], their results suggest that open models can be beneficial for students since they get engaged in reflective interactions about the subject domain, also that allowing students to inspect their learner models may boost their self-confidence and, ultimately, improve their performance. It has also been observed that students' inspection of their models leads to improved quality and robustness of these learner models.

A step forwards is the provision of interactive learner models, where interactivity is conceived as a continuous dialogue between the computer system and a learner. An example of this approach is found in STYLE-OLM [Dimitrova, 2003], an environment for interactive diagnosis where learners can inspect and discuss their conceptual knowledge and influence the content of their learner model.

Embedding open and/or interactive learner models within educational AH systems requires careful consideration. Design issues include what elements of the learner model to present to the students and how to present them in an understandable and usable manner. Some approaches, for example, present the learner with a series of elements they can update, such as their knowledge level on the main components of the domain model, or their characteristics and preferences –e.g. age, gender, preferred language, mode of presentation, etc [De Bra & Calvi, 1998, Papanikolaou et al., 2003]. Students'

confidence levels in their understanding have also been used [Bull et al., 1995], as well as peers' help, evaluation and feedback [Greer et al., 1998; Bull et al., 1999; McCalla et al., 2000].

Learner models have been presented to students through Web-based forms where they can select buttons and thereby change specific values [Bull et al., 1995; De Bra & Calvi, 1998; Papanikolaou et al., 2003], through assessment and feedback forms [Bull et al., 1999], or through graphical environments [Morales et al., 2000; Zapata-Rivera & Greer, 2000; Dimitrova, 2003].

Interactive learning modelling poses further challenges in the design and development of AH applications. The case of STYLE-OLM [Dimitrova, 2003], for example, clearly indicates the need for knowledge querying algorithms, a dialogue framework for switching between different diagnostic tactics and a suitable mechanism to dynamically update the learner model resultant from the interaction.

Morales et al. [1999] offer a summary of the main considerations for the design and implementation of overt learner models, including the following issues:

- What to make available to learners; which in turn impacts the content of the learner model and the process that leads to its integration.
- Deciding what elements can be inspected, and what others can be updated.
- Defining how to convey the learner model to students on the bases of what can be adequately shared with them –i.e. understandable and easy to use.
- Deciding to what extent changes to the learner model affect the adaptive behaviour of the system, and whether adaptive strategies can also be inspected by learners.
- The issue of the extent to which a learner model becomes less a model and more a set of system parameters as a result of its openness to learners' inspection.
- Ethical and practical considerations exist about making the model of a learner available to other learners, teachers, and interested parties.

7.4.3 The Semantic Web

Over the past few years the concept of the Semantic Web has gained relevance among the research community interested in Internet applications, including educational systems. The Semantic Web is concerned with the evolution of a Web that currently consists of content designed for humans to read to one that includes data and

information for computers to manipulate meaningfully [Berners-Lee et al., 2001; Devedzic, 2004; Shadbolt et al., 2006].

In terms of the Semantic Web for education, implications include the design and development of intelligent Web-based applications to personalise learning content and services according to the requirements of individual users [Devedzic, 2004]. In this setting, learning, teaching, collaboration, assessment and other educational activities will be mediated by agents. Agents will be responsible for locating browsing, arranging and otherwise using educational material from different educational servers.

For these software agents to roam the Web and readily carry out sophisticated tasks for users, it is essential that learning content is made computer-understandable, which can be achieved through the provision of semantic markup with pointers to sharable educational ontologies.

Accordingly, semantic markup and ontologies are of particular relevance for the advancement of the Semantic Web; this in turn has important implications for the kind of educational systems that will populate it, which also includes AH systems.

Semantic Markup

Research and development for the advancement of specifications and standards has involved a number of organisations, groups and initiatives world-wide. Of particular relevance has been the work carried out by the IMS Global Learning Consortium (**IMS/GLC**).

Some specifications that are particularly relevant for the design and development of educational AH systems include the Learning Design specification, the Content Packaging specification, and the Learning Object Metadata Standard. The Learning Design specification [IMS, 2003] aims at supporting a wide range of pedagogies in online learning; whilst the Content Packaging specification [IMS, 2004] describes data structures and XML bindings that are used to provide interoperability for Web-based content.

In addition to IMS specifications, markup and standardisation efforts of educational content are also based on the concept of Learning Objects (**LO**) and related metadata specifications; in particular the Learning Object Metadata (**LOM**) approved by the IEEE Standards Association as IEEE 1484.12.1-2002 Standard, which defines a set of

metadata elements that can be used to describe learning resources, including names, definitions, data types and field lengths.

The implications of markup for educational AH applications are diverse. An immediate impact refers to the availability of learning content associated to different learning tasks – e.g. theory, examples, exercises, assessment, etc. – and to different modes of presentation –e.g. text, images, video, audio, etc.

As discussed in previous sections of this Chapter, designing the domain model of an AH application requires to carefully determining what concepts of the subject domain to focus on, making decisions about how to present these concepts, and actually having access to the corresponding learning content files. Thus, availability of the required learning content can seriously undermine the adaptive behaviour of an AH system. Considering that developing content is a resource-consuming process, having access to repositories with metadata attached to LOs can greatly facilitate the location and re-use of learning resources.

Further considerations of this approach refer to the extent to which learning content explicitly developed for specific educational AH systems also need to comply with specifications and standards; the extent to which such content is made available to other systems and agents; and the extent to which these decisions affect the design and development process.

Furthermore, some recent approaches to the development of AH systems are entirely based on IMS specifications. The project of Berlanga & García [2004, 2005] for example, aims at developing an open model for authoring educational adaptive applications. The proposed architecture is based on the Learning Design specification for the definition of the domain model, the Content Packaging specification to define the interaction model, and the Learning Object Metadata standard to annotate LOs.

Ontologies

The availability of ontologies is another basic requirement of the Semantic Web. An ontology is a collection of information that formally defines relations among terms; it typically consists of a taxonomy and a set of inference rules [Berners-Lee et al., 2001]. Ontologies allow the definition of an infrastructure for knowledge sharing among intelligent systems independently of specific implementations [Aroyo & Dicheva, 2004].

In terms of incorporating ontologies in the design and development of AH systems, an immediate implication refers to the extent to which the domain model can be built to represent a domain ontology. Such ontology may elicit the concepts of interest in the domain, identify the hierarchy among them, and formally define the relationships and constrains between concepts [Mizoguchi & Bourdeau, 2000; Aroyo & Dicheva, 2004]. In turn, reasoning mechanisms can be used to dynamically exploit these ontologies and enhance the adaptive functionality of the applications. Examples of such domain ontologies can be found in [Henze et al., 2004; Denaux et al., 2005].

Ontologies have also been proposed for implementing the learner model [Henze et al., 2004; Denaux et al., 2005, Jovanovic et al., 2006], providing adaptive course sequencing [Jovanovic et al., 2006], or recording learner performance [Henze et al., 2004].

The current vision for the Educational Semantic Web relies on interoperability of learning content and system components; all grounded over extensive expression of semantics and standardised communication processes [Aroyo & Dicheva, 2004]. In contrast, current approaches for the design and development of educational AH systems offer little space for re-using or sharing content, knowledge or functional components [Aroyo & Dicheva, 2004, Mizoguchi & Bourdeau, 2000]. In an effort to serve better the needs of the education community, Web-based adaptive systems need to take careful consideration of the Semantic Web and its related tools, technologies and methods in order to achieve improved adaptation and flexibility for single and group users.

7.5 SUMMARY

This chapter puts forward a methodological approach for implementing the adaptive variables proposed in the research and extending the functionality of LEARNINT, which is in line with objective ten (O10) of the research..

Central to the system is the structure of its learner model, which evolves along time and interaction with each learner taking into account their behaviour, performance, expressed preferences and, possibly, affective reactions.

The system's learning content is organised around different types of learning activities, examples, exercises and assessment tasks that can be sequenced differently according to the needs and preferences of the learners and taking into account current learning

objectives. In addition, learning content is available in various media formats to suit individual learners.

The adaptation engine uses the system knowledge about the domain and about the current state of the learner model to determine the appropriate learning content for the individual user. It also has to decide about different features of the interaction and the functionality of the interface.

Another important requirement of the system is the degree of control that learners have over the application's adaptive behaviour. Given that adaptation can be system- or user-initiated, means are available for the students to determine when to activate/deactivate the adaptive behaviour of the system.

Assessment tasks are available for formative purposes, helping to identify learning needs, misconceptions and misunderstandings, which in turn determine to a great extent the kind and amount of feedback provided to the student as well as subsequent learning activities and exercises.

Additional functionality includes the customisation of the look and feel of the system according to the learners' preferences, and the possibility of opening the learner model for the students' inspection.

Possibly, the most challenging requirement for the system is finding the appropriate balance between users' expressed preferences, current level of knowledge, performance, and affective reactions to determine the most convenient configuration at any given moment of the learning interaction.

Towards the end of this Chapter a series of emerging issues were discussed that need careful consideration for the design and development of educational AH systems, including aspects relating to technological platforms, open learner models and the educational Semantic Web.