

DESIGN OF LEARNING OBJECTS TO SUPPORT
CONSTRUCTIVIST LEARNING ENVIRONMENTS

A Thesis
presented to
the Faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

by
YUANLIANG LIU

Dr. Hongchi Shi, Thesis Supervisor

DECEMBER 2005

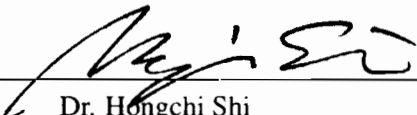
The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled

**DESIGN OF LEARNING OBJECTS TO SUPPORT
CONSTRUCTIVIST LEARNING ENVIRONMENTS**

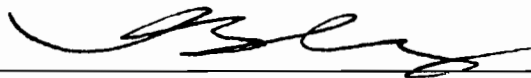
presented by Yuanliang Liu

a candidate for the degree of master

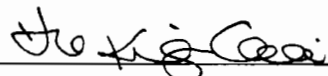
and hereby certify that in their opinion it is worth acceptance.



Dr. Hongchi Shi



Dr. Yi Shang



Dr. Dominic K.C. Ho

DEDICATION

This thesis is dedicated to my parents
Mr. Fuquan Liu
and
Mrs. Jufang Yang
Who have given me invaluable opportunities of education

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my advisor, Hongchi Shi, Ph.D., for his pioneering work in this research area, and his continuous understanding, kindness and support to me in this very meaningful and worthwhile research project. I realize that the study under his guidance is a very important step in my life. I would like to thank Yi Shang, Ph.D. for his guidance and discussion. His enthusiasm and attitude are something I always need to learn from. I would also like to thank Dominic K.C. Ho, Ph.D., for serving on my committee and for his valuable comments on my thesis. My thanks should also go to Steven Cummings, for all the busy hectic days we worked together and all the ideas we discussed. I learned a lot from both his skill and attitude towards computer programming. I would also like to thank Hairong Liu, M.S. for her help in this study. Thanks also go to the following people for their early work and exploration in this project.

Othoniel Rodriguez, Ph.D.

Su-shing Chen, Ph.D.

Special thanks to my parents who have strived very hard in their lives to ensure that I can receive my best education.

DESIGN OF LEARNING OBJECTS TO SUPPORT
CONSTRUCTIVIST LEARNING ENVIRONMENTS

Yuanliang Liu

Dr. Hongchi Shi, Thesis Supervisor

ABSTRACT

Using Constructivism to guide the design of learning objects, we develop a generic structure that classifies knowledge into different types on different levels. With a simple generic structure of learning object, learners can easily share knowledge on the Internet, and knowledge can be rendered in various ways according to different patterns. In addition to the patterns rendered, the ease and efficiency of viewing the whole picture of knowledge and zooming into any degree of details at run time allow the learner to learn the material iteratively in different ways according to her current sense-making, setting up her learning strategies at each iteration of her learning. Thus, by putting learning back into the hands of the learner, our system assists the learner to construct knowledge efficiently in the real constructivist learning environment.

List of Figures

4.1	Graphic representation of the schema of a section	19
4.2	A fragment of the XML schema of IDEAL learning objects: sectionType complexType and section_root attributeGroup	21
4.3	A fragment of XML schema of IDEAL learning objects attribute content- Type	22
4.4	Graphic representation of the schema of a component	23
4.5	A fragment of XML schema of IDEAL learning objects: componentType complexType, component_root attributeGroup, and component_vertexAttri attributeGroup	23
4.6	Graphic representation of the schema of a cloud	24
4.7	A fragment of XML schema of IDEAL learning objects: cloudType com- plexType and cloud_vertexAttri attributeGroup	25
4.8	Graphic representation of the schema of a course	26
4.9	A fragment of XML schema of IDEAL learning objects: courseType, sessionType complexType, and displaymode attributeGroup	27
5.1	IDEAL Learning Desktop displaying "My Courses" for the learner	29
5.2	IDEAL Learning Desktop displaying a cloud of JINI, which consists all components related to JINI	30
5.3	A fragment of course map configured by a course author/instructor	33

5.4	IDEAL Learning Desktop displaying a component preface	34
5.5	IDEAL Learning Desktop displaying a section in the same component shown in Figure 5.4	37
5.6	IDEAL Learning Desktop displaying a section	38
5.7	IDEAL Learning Desktop displaying a "bird view" of the course	42
5.8	IDEAL Learning Desktop displaying a course by profile	43
5.9	IDEAL Learning Desktop displaying a "shallow knowledge" session by profile	44
6.1	Architecture of IDEAL Learning Object System	51

Contents

ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
LIST OF FIGURES	iv
Chapter	1
1 Introduction	1
2 Related Work	5
3 Constructivism and Our Design	10
3.1 Using Constructivism to Guide Learning Object Design	10
3.2 Our Design	11
4 Learning Object Schema Design	17
4.1 Section Design	19
4.2 Component Design	19
4.3 Cloud Design	21
4.4 Course Design	22

5	Authoring and Rendering of Learning Objects	28
5.1	View Default	31
5.2	View by Patterns	42
5.3	View by Profile	43
6	Implementation	46
7	Conclusion	53
	REFERENCES	55

Chapter One--Introduction

There have been many efforts for authoring of learning objects. However, very little research on learning theory based design of learning objects has been conducted (Wiley, 2002). Yet there are many research questions regarding learning objects that need to be answered. Using Instructional Design to guide learning object design is problematic. Instructional Design assumes that based on learner's model we can apply pedagogic rules of Instructional Design to select learning objects for the learner in a certain way to achieve effective knowledge transferring (Jonassen, 1993). However, a learner at any moment may want to view the material in many different ways. Restricting the learner to only one way of viewing the learning material is very prohibitive to active learning.

Another assumption of Instructional Design is that knowledge is well-structured. Constructivism, on the contrary, realizes the fact that knowledge is often ill-structured (Jonassen, et al., 2004). In reality, those rigidly-structured learning objects are very difficult for the course authors to work on and cooperate with each other. It is also very difficult to apply rigidly-structured learning objects to various learning domains/subjects.

Contrary to Instructional Design, which assumes that knowledge can be transferred to the learner by rearranging the learning material according to pedagogic rules, the assumption of constructivism is that learning is associated with context, that learning occurs in activities, and that to achieve effective learning the learner has to be put in a context and involved in an activity (Jonassen, et al., 2004; Brown, et al., 1989; Henning, 2004).

Although a departure from objectivism learning theory that is represented by Instructional Design and addressing the problems of objectivism, constructivists learning environments still rely on traditional knowledge carriers. A few researchers (Bannan-Ritland, et al., 2002; Orrill, 2002) making efforts to combine learning object and constructivism together focus largely on how learning object can be used in specific constructivist learning environments instead of seeking a generic structure which will help learners learn in many possible creative constructivism ways.

We combined constructivism and objectivism in our design of learning objects. Since learning is associated with context and activities, the learning material has to be viewed many times corresponding to different learning contexts. By choosing appropriate activities and viewing the material differently each time, the learner goes through the learning material for several times to construct her knowledge. In real constructivism learning environments, e.g. the real world, for the learner to set up her best strategy in going through these iterations, it is highly desired that a system can help the learner to collect related learning materials, viewing the whole picture of knowledge, focusing on important information at a time without being distracted by unnecessary information, and zooming into various degrees of details at run time. Furthermore, the ability of easily sharing knowledge is also very important since we regard teaching as sharing learning experience.

How technology, especially learning object technology, can help implement such a system?

Using the minimalist approach, we come up with a generic structure of learning object, which classifies knowledge into different types on different levels and makes it

possible and easy for authors at different levels to cooperate with each other to generate the learning objects.

Our design of learning objects for our IDEAL e-learning environment (Shang, et al., 2001; Shi, et al., 2002) makes it easy for the learner to collect related learning material, have the learning material rendered in “pattern” (to render the learning material in different ways to allow learners to focus on the important information that they need and can make sense of without being distracted by information they do not need or cannot make sense of) according to different learning needs, and navigate and adapt the material flexibly at run time. More specifically:

Our system can automatically generate course maps from related knowledge components according to some patterns. For example, the learner can have a bird-view of the course, or have a novice view of the course. In addition to automatically generating course map according to some patterns, the system is capable of generating the learning material in several other ways.

The learner is given the flexibility to further refine the course map by changing the sequence of the knowledge components, adding or deleting some components, or changing the display mode of the components. By changing the display mode of the component, the learner can have the component display more or less difficult content or detailed content. This way, the learner can focus on the important things for her learning at a certain time.

In addition, the learner can have the section (a component may have several sections) display more or less content at run time. So the learner can always view more stuff if she feels interested or needed.

The system emphasizes on awareness of the whole picture of knowledge. So the whole picture of knowledge is always rendered first and very convenient to access at any time.

With these features, our system can greatly assist the learner to construct their knowledge in the real constructivism learning environment.

Chapter Two--Related Work

There have been many efforts for development of learning objects in industry, and there are a few committees making standards for specification of learning objects. But most of these efforts have been “learning theory neutral” (Wiley, 2002). They ignore an important question: how learning objects support learner learning and thus “fail to provide solutions for many current learning environments” (Orrill, 2002). The discussion on learning objects largely focuses on technical development (LTSC, 2002). Their use of metadata is intended to provide a library card catalog function that will help to retrieve learning objects from digital libraries (Wiley, 2002). But problems arise when they want to compose new learning objects by reusing existing learning objects (Wiley, 2002). How can learning objects be combined to be instructional meaningful? The reusability, scalability, and interoperability are not addressed in their development. Thus, the combination of learning objects in their development is very likely to be instructional useless. Bannan-Ritland argues that it is very crucial at this point to consider the implication of learning object use and implementation in the instructional context prior to a full-scale implementation of learning object technology (Bannan-Ritland, et al., 2002).

The few researchers trying to combine learning object development and learning theory focus primarily on Instructional Design theory (Wiley, 2002). However, using Instructional Design to guide learning object research is very problematic (Bannan-Ritland, et al., 2002).

Instructional Design assumes that based on a learner's model we can apply pedagogic rules of Instructional Design to select learning objects for the learner in a certain way to

achieve effective knowledge transferring. But a learner, even at one moment, may want to view the material in many different ways. Learning style, as in Instructional Design, is meaningless for active learners since active learners use all kinds of learning styles according to her current learning needs (Bereiter, et al., 1989). Rendering the learning material in a certain way while disallowing the learner other ways of viewing the material is very prohibitive to active learning.

Another assumption of Instructional Design is that knowledge is well-structured. However, constructivism realizes that knowledge is often ill-structured, complex, and dynamic. People's mental model, the structure of knowledge by which they perceive the world, is always changing. Constructivism holds that the change of knowledge structure is a learning process. Knowledge is always reconstructed in the context of the individual's understanding and purposes. Thus, every individual holds a unique perspective of the structure of knowledge. To have a rigidly-defined structure of knowledge and impose it on everyone makes it very difficult for the authors to cooperate with each other to author learning objects by reusing each other's work. It is also very difficult to apply those rigidly-structured learning objects to various learning domains/subjects.

Constructivism is a name given collectively to a wide variety of views, theories, and instructional models. Constructivism assumes that learning is an active process of constructing rather than acquiring knowledge, and that instruction is a process of supporting that construction rather than communicating knowledge content (Jonassen, 1993). Constructivism generally believes that "most learning domains are ill-defined (complex), that learning outcomes are largely metacognitive in nature, and that learners

are required to actively participate in the learning process to construct meaningful knowledge rather than acquire a predetermined set of skills in a pre-specified manner” (Bannan-Ritland, et al., 2002). Constructivism learning theory emphasizes that learning is learners’ sense making of the world, that learning needs arise from activities, and that learning is strongly associated with activities (Brown, et al., 1989; Henning, 2004). Since learning is learners’ sense making, learners’ feel in learning is very important. Learners need to derive learning needs from activities, feel learning gaps, and actively locate resources to meet learning needs.

Further examination of the foundation of Instructional Design and Constructivism will help us select appropriate learning theory to guide learning object design. The prominent related theories are as follows:

- According to Bannan-Ritland, Instructional Design is based on so called traditional Cognitive Information Processing (CIP), which holds that “information undergoes a series of transformations in the mind in a serial manner until it can be permanently stored in long-term memory in packets of knowledge that have a fixed structure” (Bannan-Ritland, et al., 2002).
- Parallel Distributed Processing (PDP) “perceives long-term memory as a dynamic structure (or network) that represents knowledge in patterns or connections with multiple pathways instead of concept nodes and propositions.” “A fundamental distinction between the traditional view of CIP and PDP models of memory is that information processing occurs in parallel instead of a serial manner, activating

knowledge patterns simultaneously and continuously adjusting them as a function of new information to resolve cognitive dissonance” (Bannan-Ritland, et al., 2002).

- Cognitive Flexibility Theory holds that “learners ought to be able to assemble situation specific knowledge in a domain, and this demands the attuning of special cognitive processing skills,” thus “in sharp contrast with the traditional view of CIP in which knowledge is thought of as discrete and static entities to be retrieved intact from memory to demonstrate a learned capability” (Bannan-Ritland, et al., 2002).
- Situated Cognition holds that “the context or the activity which frames knowledge in a particular domain is as important as the content that is learned because it is referenced by that activity” (Bannan-Ritland, et al., 2002).
- Distributed Cognition states that “the social processes *themselves* should be considered as cognitions” (Bannan-Ritland, et al., 2002).
- Generative Learning Theory holds that “the learner is not a passive recipient of information but an active participant in the instructional experience.” Thus, “the generative learning process requires the learner to manipulate, interpret, organize or in some active manner make sense of his or her environment” (Bannan-Ritland, et al., 2002).

Through analysis of the grounding assumptions of various learning theories we can see that Instructional Design misses the whole picture of learning, and that using it to guide learning object design is problematic or at least very insufficient.

To summarize, Instructional Design is itself problematic. Using it to guide learning object research is misleading and has not had satisfactory result (Orrill, 2002). Compared

to Instructional Design, Constructivism is closer to the truth of learning. However, there has not been an effective media to support learners to construct knowledge efficiently. We propose, constructivism, as it presents a more complete picture of learning, can be used to guide learning object design. Furthermore, we hold that learning object designed under the guideline of constructivism provides a new kind of knowledge carrier that can support learners constructing knowledge in a very efficient way.

Chapter Three--Constructivism and Our Design

3.1 Using Constructivism to Guide Learning Object Design

Under the guideline of Instructional Design, the goal of learning object design is to design “architecture of instructor-defined content that is configured and automatically generated for the learner based on a specific selected instructional strategy” (Bannan-Ritland, et al., 2002). If we deviate from such a design goal and instead use constructivism to guide learning object design, what would be our goals of design and features of such systems?

Bannan-Ritland has attempted to propose new principles or guidelines for learning object systems that are guided by constructivism learning theories. Briefly, she proposes that learning object designed with constructivism learning theories should generally be able to:

- support learner-generated artifacts by incorporating learner contributions;
- consist of multiple levels of granularity to afford reusability, flexibility, accessibility and adaptability of learning objects; and
- contain frameworks as learning objects that provide structure for instructional experiences and incorporate a linking system to facilitate their content population.

A flexible navigation is very important in constructivist learning environments.

Orrill holds that constructivist learning environments require revisiting existing knowledge as learners construct new understandings (Orrill, 2002). Orrill points out such

systems are to provide learning environments that are rich with learning experiences and resources and these environments should be learner-centered in that learners are responsible for determining how to learn and what to learn. Orrill further points out that “if we cannot support student movement between and within the objects based on their evolving needs and understandings, the objects will not be serving the scaffolding functions they are intended to serve.”

3.2 Our Design

Our design of learning object is mostly inline with Bannan-Ritland’s proposed guidelines, and our design shares some similarity with Orrill’s design. The major difference is that in our system, we are less concerned with a specific type of constructivism learning theory and guiding the learner through a specific constructivism learning process. The reason is based on the following assumptions:

- Learners can make the best judgment what to learn and how to learn if they are presented the whole picture and allowed to try various components; and
- Our learners are learning in the *real* constructivist learning environment, e.g., the real world, instead of in some artificial learning environments.

With the first assumption, instead of teaching the learner how to learn constructively, it is more important to let the learner see the whole picture of the topic and be able to view the material quickly in a flexible and adaptive way. By this, we are holding to the most fundamental assumption of constructivism that the learner explores learning in

many creative ways according to her learning needs and rich resources provided. Confining learners to a certain constructivism environment discourages learners to explore learning and solve learning problems creatively. As Brown and Duguid point out, once people see the need for learning and the resources are easily accessible, people will devise ways to learn in whatever way suits the situation (Brown, et al., 1989). Furthermore, in constructivist learning environments, scaffolding plays a key role for the learner to construct her knowledge (Jonassen., 1993). Being able to view the whole picture and try things quickly help the learner locate the scaffolding efficiently. For the learner to be able to do that effectively and efficiently, e.g., quickly obtain the whole picture of the knowledge, the learner needs to collect all the related learning components and know what these components are about without reading through all the content. However, with the conventional knowledge carriers, a very significant amount of time is wasted in locating the needed information and reading unnecessary information.

With the second assumption, the learner is constructing her knowledge in the real world, instead of in some artificial learning environments. The real world is the best constructivist learning environment. People learn in the community of practice (Barab, 2000), and they learn through enculturation in society (Brown, et al., 1993). So it is not our task to build an artificial learning environment to help engaging learners in constructivism learning as mentioned by Savery (Savery, et al., 1995) and followed by Bannan-Ritland and Orrill in their thinking and experimentation of constructivism-guided learning object design. Our system is to *help* the strong active learner to learn constructively in the real constructivist learning environment. The real constructivist learning environment provides the real challenging constructivism tasks mentioned by

Savery (Savery, et al., 1995) compared with those artificial constructivist learning environments. However, our system, of course, can be used in various artificial constructivist learning environments although it is not our primary intention for it to be used that way.

Another difference of our approach from Bannan-Ritland and Orrill's is that we emphasize learners building up the mental model of knowledge although such a mental model is under constant change and refining. Our system provides the facility to help learners building up the mental model of a domain by constructing the course map. As Duff and Jonassen argue that the best learning environment is to combine constructivism with objectivism (Jonassen, 1993), our support for learner building up mental model reflects a better combination of constructivism and objectivism.

So different from Bannan-Ritland and Orrill's approach, our goal of learning object design is a system that allows learners to easily share knowledge on the Internet, collect learning material, view the learning material quickly through several iterations, and be able to focus on current learning within each iteration.

In constructivist learning environments, the learning material has to be viewed several times. Orrill points out that constructivist learning environments require revisiting existing knowledge as learners construct new understandings (Orrill, 2002). Since Parallel Distributed Processing, as mentioned above, "activates knowledge patterns simultaneously and continuously adjusts them as a function of new information to resolve cognitive dissonance" (Bannan-Ritland, et al., 2002), existing knowledge and learning material has to be reviewed frequently to "resolve cognitive dissonance" and construct new understanding. So we think that constructivism demands the learner to: first, use all

kinds of learning material that she can make sense of at the present time to achieve her current learning goal; secondly, after she achieves a new level of understanding, thus has new learning needs and goals, and can make sense of more things, she needs to find again all the learning material that she can make sense of or she can feel her interests in now. So in constructivism, the learner has to do this iteratively for many times. Thus, the learning material has to be read for many times and be read differently every time according to what the learner can make sense of. This demands that learning material be read very fast and navigated very easily and flexibly. The easier the learner can find information she needs without being distracted by unnecessary information, the faster the learner can accomplish her current layer of learning and thus improve the efficiency of her learning. What the learner can make sense of during each layer of learning is very difficult to predict, and it changes from moment to moment as she reads the learning material. We do not think any Instructional Design theory is sufficient to meet such a demand. In this work, we study how learning object system can be designed to help the learner to meet such a demand.

Additionally, as mentioned above, we assume that learners can make the best judgment on what to learn and how to learn if they are presented the whole picture and allowed to try various components. In constructivism, learners are at the center of learning. To guide their learning and construct knowledge more effectively, the learners have to have a complete picture of the domain, and such a complete picture is usually built up by extensive reading. The learner needs to collect all the related material and read iteratively according to her current sense-making, diving into any material quite randomly to deepen her understanding in one area and consequently in the whole subject

domain. Thus being able to read fast, locating information needed quickly and thinking on the level of the whole subject domain are the important skills to learn efficiently in constructivist learning environments. On the contrary, not able to read fast, distracted by unnecessary information and unable to see and think through the complete picture are hindering efficient learning in constructivist learning environments.

To support learners viewing learning material at a certain time for a certain need, we use the concept of “pattern” instead of learning style. As we mentioned above, the concept of Learning Styles is problematic. The assumption behind it is not appropriate for active learners since active learners normally apply all kinds of learning styles just according to her current needs. It is better to use the concept of pattern. By pattern, we recognize that learning material should be adapted to suit different purposes of viewing and such patterns are what the learner can “choose” from. For example, we have a bird view pattern, by which learners can have a bird view of the material. According to this pattern, we select appropriate components with appropriate display mode and the sections will be displayed accordingly with the content exposed more or less. By choosing different patterns, the learner can have the learning material rendered differently to achieve her learning goal at the present time more efficiently without being distracted by unnecessary information. By making patterns available to the learner, we encourage the learner to view material differently and learn more actively. The learner can change the display mode of the component and display level of the section to let them display more or less at run time.

Thus, our primary goal of learning object design is to facilitate the learner to easily collect related learning material, view the material through several iterations, in each

iteration have the material rendered in some “pattern,” have the whole picture of knowledge easily accessible, navigate flexibly, and adapt the material at run time.

Overall, to incorporate constructivism into learning object design, our system has the following features:

- a simple and generic structure that would allow learners to easily participate in the construction of learning objects and allow the learning object easily changeable throughout its usage;
- rendering of learning objects according to some “patterns” and at the same time allow learners to further configure and refine the “pattern”, and have the content adapted to learners’ needs at run time;
- always have the complete picture rendered first and have it easily accessible at any time;
- to achieve the above, learning objects will consist of multiple levels of granularity to afford reusability, flexibility, accessibility and adaptability of learning objects; and
- a mechanism to group (linking) things together.

More theoretical ground of our design of learning object can be found in Interactive Computation (Wegner, 1997) and Christopher Alexander’s theory regarding living centers, which is best summarized in his book Nature of Order (Alexander, 2003). It is beyond the scope of this paper to discuss them here.

Chapter Four--Learning Object Schema Design

We propose a simple generic structure that can be applied to various domains and that authors can easily work on and cooperate with each other. Our design of learning object divides knowledge into a few layers that can easily fit into various situations and we defines a simple set of attributes at each level. At this stage, we pick the attributes of content type, difficulty level and detail level as the basic attributes we want to apply to knowledge at each level. These attributes are commonly important attributes and can be very useful. We explore how knowledge can be reused with such a simple structure and to demonstrate that learning objects designed with constructivism principle can be supportive of learning. More attributes, if proved important and useful, can be added later. Furthermore, it is more important to keep the design simple and useful so more people can participate in the cooperation online than to complicate it at an early stage of design with unnecessary artifacts.

More importantly, we noticed that some attributes are strongly associated with the context it is within. For example, difficulty level and detail level are much more meaningful and accurate when they are used to compare a set of knowledge units within the same parent knowledge unit. Some other attributes, such as content type, are relatively less associated with the context. So for those attributes strongly associated with the context, it is better to only assign attributes to them when they are put together to construct a bigger knowledge unit instead of requiring authors to describe their independent standalone knowledge units with such attributes. This way, we lift the burden off the authors of independent standalone knowledge units so they don't have to

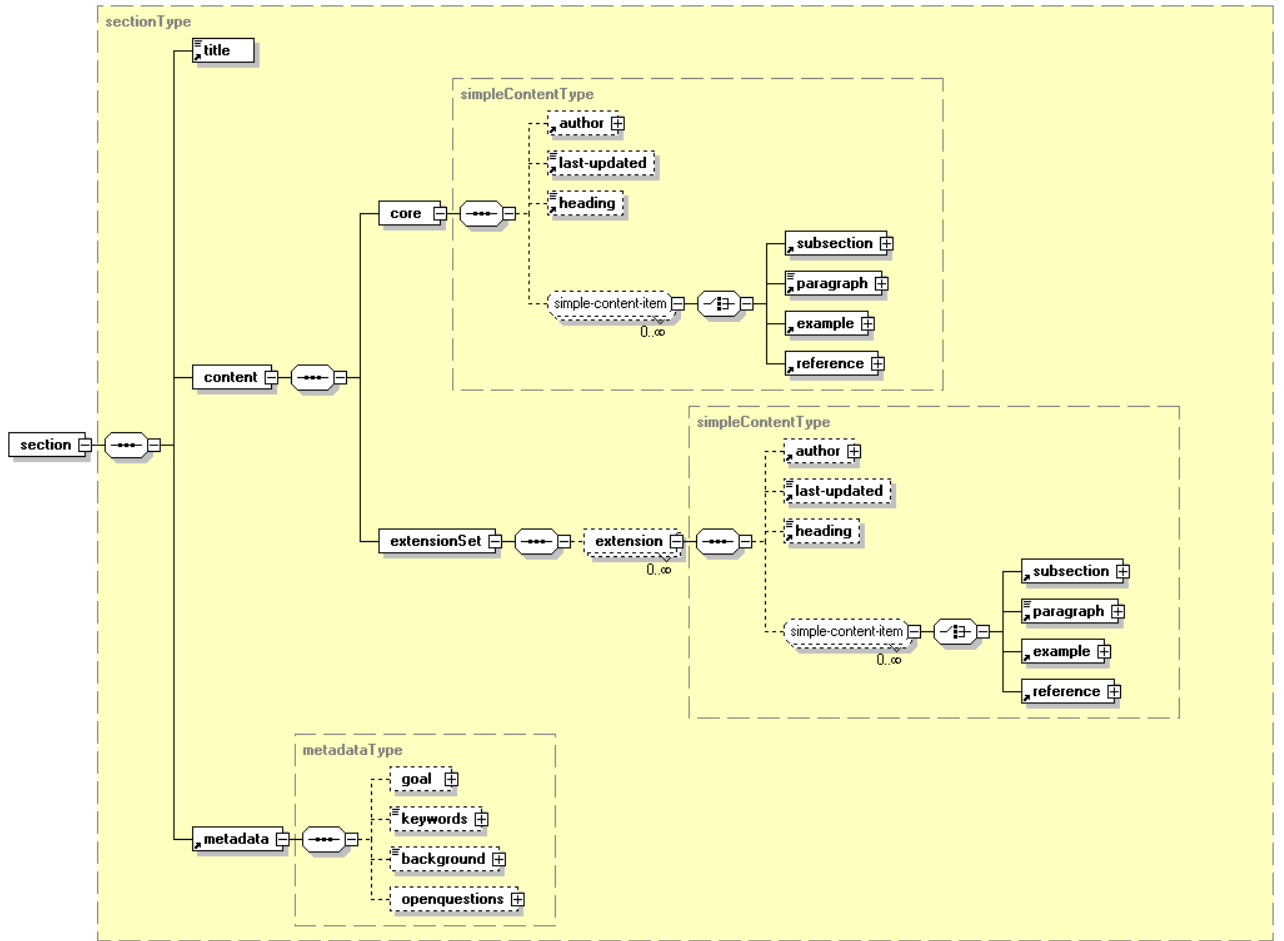
make those difficult tricky decisions. But for independent standalone knowledge units, we require authors to give them attributes such as content type and this will greatly help authors at the higher level to easily locate knowledge units they need. So we regard the issue of reusability of learning object as an issue of cooperation, e.g. how authors at different levels can cooperate with each other so that every author can focus on her job and help out each other. As we are aware of, this issue is not sufficiently addressed by the learning object development community.

A learning object in our system is regarded as a *component* of knowledge. So for one learning topic, there might be several components related to it. These components related to one topic can be grouped together as a *cloud*. Each component consists of several *sections*, which are the smallest units of knowledge.

4.1 Section Design

The smallest unit of knowledge is defined as *section* as shown in Figure 4.1. A section has one *core* and several *extensions*. The core is the most essential part of a section. The core uses the least words to convey the meaning of the section, while extensions can be added to provide more detailed explanation of the section. Metadata of a section is used to provide some other information about the section such as keywords.

Figure 4.1: Graphic representation of the schema of a section



Generated with XMLSpy Schema Editor www.xmlspy.com

Since a section might be used by different learning objects, we use a group of attributes named `section_root` to depict this section as shown in Figure 4.2. Noticeably, the author needs to specify the content type of this section. The value of content type is as shown in Figure 4.3.

4.2 Component Design

A component consists of several sections as its vertices as shown in Figure 4.4. When each section is put as a vertex into a component, the author needs to use a group of attributes named `component_vertexAttri` to depict this section within the context of this

component. The attribute group `component_vertexAttri` is shown in Figure 4, which depicts the difficulty level (lower bound and upper bound) and the detail level of the section within the context of this component. The lower bound difficulty level and upper bound difficulty level together give the range of difficulty levels of this section within the context of this component. The values of difficulty level are novice, beginner, intermediate, advanced, and expert as shown in Figure 4.5. The detail level means how detailed this section is within the context of this component. A component can have some sections serve as the abstract information of this component and some as more detailed information. So the values of detail level are abstract, description, normal, and detail as shown in Figure 4.5. Each component has metadata to annotate other information about this component. A group of attributes named `component_root` is used to depict the component. The attribute `contentType` classifies components into different types.

4.3 Cloud Design

Several components related to one topic can be grouped together as a cloud. Components are put as vertices of the cloud, with each of them depicted with a group of attributes named `cloud_vertexAttri`.

Sections, components, and clouds are created by the authors. One author can group several sections (possibly created by other authors) together as a component, or group several components (possibly created by other authors) together as a cloud. Thus, sections and components can be reused.

Figure 4.2: A fragment of the XML schema of IDEAL learning objects: *sectionType* *complexType* and *section_root* *attributeGroup*

```

<xsd:complexType name="sectionType">
  <xsd:sequence>
    <xsd:element ref="title"/>
    <xsd:element name="content">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="core" type="simpleContentType"/>
          <xsd:element name="extensionSet">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="extension" type="simpleContentType" minOccurs="0"
                  maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="metadata"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="section_root"/>
</xsd:complexType>

<xsd:attributeGroup name="section_root">
  <xsd:attribute ref="id" use="optional"/>
  <xsd:attribute ref="media" use="optional"/>
  <xsd:attribute ref="contentType" use="optional"/>
</xsd:attributeGroup>

```

4.4 Course Design

A course can be manually built up by an author. It can also be automatically generated by the system. A course is generated from a cloud, which contains components related to one topic.

A course consists of several sessions as shown in Figure 4.8. Each session consists of several components as the vertices and achieves some learning goals. Each vertex has a group of attributes depicting the display mode of the component as shown in Figure 4.9.

The `displaymode` attribute group includes `difficulty`, `level`, `lowerdisplay`,

difficultylevelupperdisplay, and detailupperdisplay. They describe how this component is going to be displayed. The values of each are the same as the component_vertexAttri described above. If a vertex has a difficultylevellowerdisplay of “beginner” and difficultylevelupperdisplay of “advanced,” only the content of difficulty level ranging from “beginner” to “advanced” of this component will be displayed while the content of “novice” and “expert” of this component will not be displayed. If a vertex has a detailupperdisplay of “normal,” only the “abstract,” “description,” and “normal” parts of the component will be displayed while the “detail” part will not be displayed.

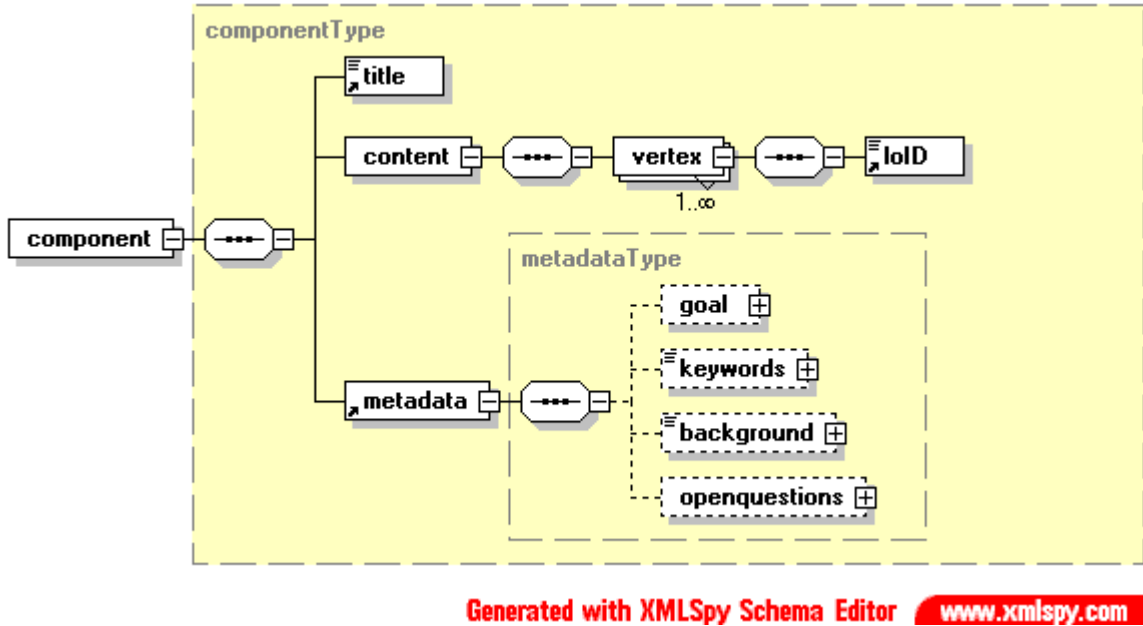
Figure 4.3: A fragment of XML schema of IDEAL learning objects: attribute contentType

```

<xsd:attribute name="contentType" type="loType"/>
<xsd:simpleType name="loType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="example"/>
    <xsd:enumeration value="examplereallife"/>
    <xsd:enumeration value="examplesamplecode"/>
    <xsd:enumeration value="reference"/>
    <xsd:enumeration value="practice"/>
    <xsd:enumeration value="core"/>
    <xsd:enumeration value="relaxing"/>
    <xsd:enumeration value="relaxinghistory"/>
    <xsd:enumeration value="relaxingcomments"/>
    <xsd:enumeration value="resource"/>
    <xsd:enumeration value="resourcespecification"/>
    <xsd:enumeration value="resourcedownload"/>
    <xsd:enumeration value="resourcecommunity"/>
    <xsd:enumeration value="resourcecolumn"/>
    <xsd:enumeration value="resourcetutorial"/>
    <xsd:enumeration value="question"/>
  </xsd:restriction>
</xsd:simpleType>

```

Figure 4.4: Graphic representation of the schema of a component



Generated with XMLSpy Schema Editor www.xmlspy.com

Figure 4.5: A fragment of XML schema of IDEAL learning objects: componentType complexType, component_root attributeGroup, and component_vertexAttri attributeGroup

```

<xsd:complexType name="componentType">
  <xsd:sequence>
    <xsd:element ref="title"/>
    <xsd:element name="content">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="vertex" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="loID"/>
              </xsd:sequence>
              <xsd:attributeGroup ref="component_vertexAttri"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="metadata"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="component_root"/>
</xsd:complexType>

<xsd:attributeGroup name="component_root">
  <xsd:attribute ref="id" use="optional"/>
  <xsd:attribute ref="media" use="optional"/>
  <xsd:attribute ref="contentType" use="optional"/>
</xsd:attributeGroup>

```



```

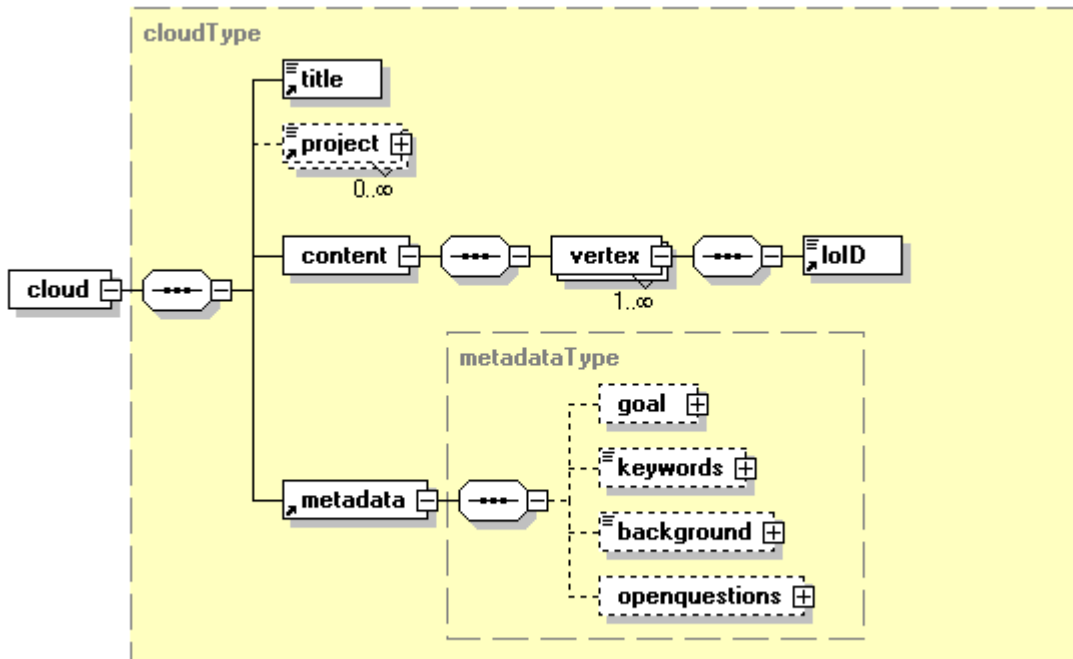
<xsd:attributeGroup name="component_vertexAttri">
  <xsd:attribute name="difficultylevellowerbound" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="difficultylevelupperbound" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="detail" type="detailLayer" use="optional"/>
</xsd:attributeGroup>

<xsd:simpleType name="detailLayer">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="abstract"/>
    <xsd:enumeration value="description"/>
    <xsd:enumeration value="normal"/>
    <xsd:enumeration value="detail"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="loDifficultyLevel">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="novice"/>
    <xsd:enumeration value="beginner"/>
    <xsd:enumeration value="intermediate"/>
    <xsd:enumeration value="advanced"/>
    <xsd:enumeration value="expert"/>
  </xsd:restriction>
</xsd:simpleType>

```

Figure 4.6: Graphic representation of the schema of a cloud



Generated with XMLSpy Schema Editor www.xmlspy.com

Figure 4.7: A fragment of XML schema of IDEAL learning objects: cloudType complexType and cloud_vertexAttri attributeGroup

```
<xsd:complexType name="cloudType">
  <xsd:sequence>
    <xsd:element ref="title"/>
    <xsd:element ref="project" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="content">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="vertex" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="loID"/>
              </xsd:sequence>
              <xsd:attributeGroup ref="cloud_vertexAttri"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="metadata"/>
  </xsd:sequence>
  <xsd:attribute ref="id"/>
</xsd:complexType>

<xsd:attributeGroup name="cloud_vertexAttri">
  <xsd:attribute name="difficultylevellowerbound" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="difficultylevelupperbound" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="detail" type="detailLayer" use="optional"/>
</xsd:attributeGroup>
```

In summary, we use the minimalist approach in our design of learning objects. We design a leaning object with a generic structure, classifying knowledge into different types at different levels and making it possible and easy for authors at different levels to cooperate with each other by reusing each other's work.

Figure 4.8: Graphic representation of the schema of a course

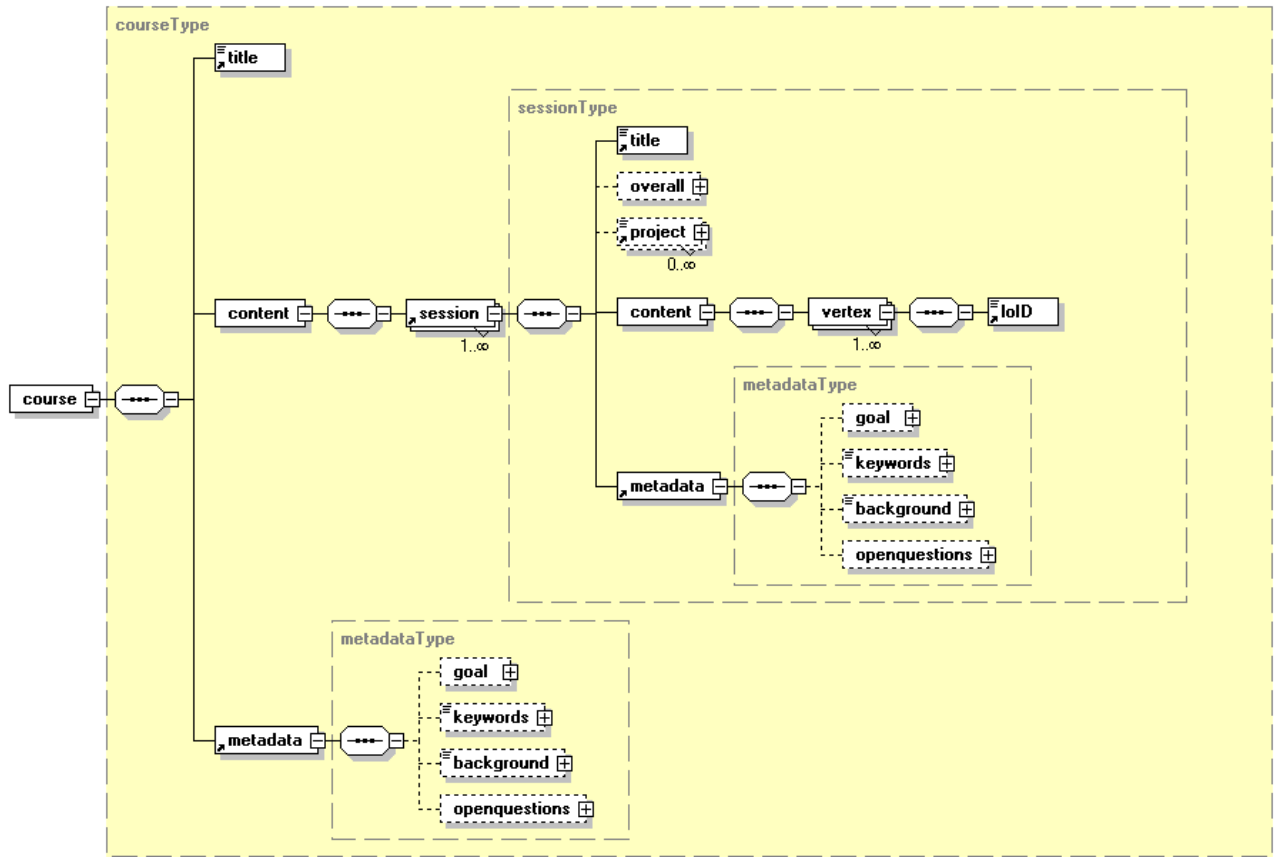


Figure 4.9: A fragment of XML schema of IDEAL learning objects: *courseType*, *sessionType* complexType, and *displaymode* attributeGroup

```

<xsd:complexType name="courseType">
  <xsd:sequence>
    <xsd:element ref="title"/>
    <xsd:element name="content">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="session" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="metadata"/>
  </xsd:sequence>
  <xsd:attribute ref="id"/>
</xsd:complexType>

<xsd:complexType name="sessionType">
  <xsd:sequence>
    <xsd:element ref="title"/>
    <xsd:element name="overall" type="loIDType" minOccurs="0"/>
    <xsd:element ref="project" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="content">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="vertex" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="loID"/>
              </xsd:sequence>
              <xsd:attributeGroup ref="displaymode"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="metadata"/>
  </xsd:sequence>
  <xsd:attribute name="seqnum" type="xsd:long" use="required"/>
</xsd:complexType>

<xsd:attributeGroup name="displaymode">
  <xsd:attribute name="difficultylevellowerdisplay" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="difficultylevelupperdisplay" type="loDifficultyLevel" use="optional"/>
  <xsd:attribute name="detailupperdisplay" type="detailLayer" use="optional"/>
</xsd:attributeGroup>

```

Chapter Five--Authoring and Rendering of Learning Objects

In our design of learning objects, sections represent the smallest significant units of knowledge with basic attributes such as content type used to categorize sections. The attributes help component authors to choose which sections to be included into appropriate components.

Inside a component, some attributes, such as content type, difficulty level, and detail level, are used to depict sections with the context of the component. Components related to a topic can be grouped into a cloud.

Inside a cloud, some attributes such as difficulty level and detail level are used to depict components within the context of the cloud. Since a cloud contains all the components related to a topic, a course is generated from the cloud, either manually by an instructor/author or automatically by the system.

When a course is generated from the cloud by an instructor/author, the expert experience can be embedded into the configuration of the course, such as what components are chosen, what sessions are made, what sequence of components are chosen, what display mode for each component, etc. When it is generated by the system, certain patterns can be applied to the cloud to generate a course corresponding to the learner's profile and the goal of learning.

Authoring of a learning object with our design of learning objects becomes easy and can be done collaboratively by several authors across the Internet. Authors at each level (section, component, and cloud) have to accomplish tasks that are easy to carry out at that level and will significantly help the authors of the next level. For example, for section

authors, it is rather easy for them to depict what content type their sections are. Knowing the sections' content type will significantly help the component authors when they choose sections to include into components. On the contrary, attributes such as difficulty level and detail level are associated with a certain context. Should we require a section author to describe what difficulty level this section is, it would give the section author tremendous difficulty when making sections. Once the section is put into a component, however, difficulty level and detail level need to be given to describe this section within the context of the component, which can be easily done since it is within a certain context.

Figure 5.1: IDEAL Learning Desktop displaying "My Courses" for the learner



Figure 5.2: IDEAL Learning Desktop displaying a cloud of JINI, which consists all components related to JINI

Course: jini

Raw components related to this course

loID	loTitle	content type	detail	minimum difficulty level	maximum difficulty level
ctjini050101	One simple JINI example: a printer service	example	normal	novice	beginner
ctjini050201	One complex but super example: The Network Revolution	example	normal	intermediate	intermediate
ctjini010101	JavaSpace	core	normal	intermediate	expert
ctjini040101	JINI resource	resource	normal	novice	expert
ctjini070101	API of JINI	practice	normal	novice	expert
ctjini070201	Sample Code of JINI	practice	normal	intermediate	advanced
ctjini070301	Organization of JINI code packages	practice	normal	intermediate	advanced
ctjini010301	JINI architecture	remote	normal	intermediate	advanced
ctjini010401	JINI communication	remote	normal	intermediate	advanced
ctjini010501	Processes: no special processes	remote	normal	intermediate	advanced
ctjini010601	Naming--jini lookup service	remote	normal	intermediate	advanced
ctjini010701	Synchronization	remote	normal	intermediate	advanced
ctjini010801	Fault Tolerance	remote	normal	intermediate	advanced
ctjini010901	Security	remote	normal	intermediate	advanced
ctjini010A01	Event in details	remote	detail	intermediate	advanced
ctjini010B01	Lookup service in details	remote	detail	intermediate	advanced
ctjini010C01	JINI Transaction in details	remote	detail	intermediate	advanced

keywords :
 JINI
 JavaSpace

background :
 Java

foreground :

background and foreground :
 Distributed System

By categorizing learning material into different content type on different levels and using difficulty level and detail level to describe them within a context, we not only make it easy for authors to cooperate with each other by reusing each other's work, but also make the rendering of learning objects easily adaptive to different learning conditions.

The student can have several ways of viewing the material as shown in Figure 5.1. She can click "view default" to choose to view the course created by the author/instructor if such a version is available. Instead, she can also choose to have the course generated automatically by the system according to some patterns. She also has the choice to view

the learning material by her profile, by which the learning material will be generated session by session. Finally, she can also choose to just view all the raw components related to the topic as shown in Figure 5.2. When viewing raw components, the content will not be adaptive.

5.1 View Default

After clicking “view default” as shown in Figure 5.1, the course map configured by the author/instructor is rendered to the student as shown in Figure 5.3. In the course map, the components are grouped by the author into several sessions, with each session corresponding to a certain goal of learning. The author/instructor defines the title, goal, keywords of each session. For each component in the session, the author/instructor pre-defines its display mode according to the author/instructor’s teaching experience and expertise.

The learner can accept this course map and view the component one by one. She can also add more components to the session from the raw components, delete some components, change the sequence of the components, and change the display mode of the component.

By clicking “View this,” the learner can view the component according to the display mode configured as shown in Figure 5.4. For example, for the learning object “JavaSpace,” if the display mode is “normal,” “novice,” and “advanced,” respectively, the component will have these sections to be displayed as shown in Figure 5.4(a). If the MaxDetail is changed to “abstract,” only one section, which is of “abstract” in this component, will be displayed as shown in Figure 5.4(b). Changing both min and max difficulty level to “novice” will only have one section displayed, whose difficulty level

includes “novice” as shown in Figure 5.4(b). The XML document of this component is shown in Figure 5.4(c).

Clicking on the “View Content” button of the component preface page, the learner can view the component’s sections sequentially.

Here the section is displayed according to the `component_vertexAttri` of this section within the component as shown in Figure 5.5. In our design, we only use the “detail” attribute for adaptation when rendering this section. If the “detail” attribute’s value is “abstract,” which means this section’s detail level is “abstract” within the context of this component, we will show more content of this section than if its value is “detail.” For example, the first section of the component “JavaSpace” is displayed as shown in Figure 14(a), while the second section is displayed as shown in Figure 5.5(b), since the first’s “detail” value is “abstract” while the second’s is “normal.”

The learner can always increase or decrease the display level of the section at run time by clicking the corresponding button on the “View Section” page as shown in Figure 5.6.

The learner is given the flexibility to further refine the course map by changing the sequence of the knowledge components, adding or deleting some components, or changing the display mode of the components. By changing the display mode of the component, the learner can have the component display more or less difficult content or detailed content. This way, the learner can focus on the important things for her learning at a certain time. The learner can save her configuration of a course map into the database, and the course map configured by the student will be retrieved from the database next time. Furthermore, the learner can have the section display more or less

content at run time. So the learner can always view more stuff if she feels interested or needed.

Figure 5.3: A fragment of course map configured by a course author/instructor

Course: jini

Course Map

Notice: you can further configure the course the way you want to view

Title: JINI

Session 1: Welcome to the JINI world!
Goal: By some JINI application examples, get a general idea of what JINI is, what is the world of JINI
Keywords:
 JINI

loID	loTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select				
ctjini050101	One simple JINI exampl	normal	beginner	advanced	<input type="checkbox"/>	Append	MoveUp	MoveDown	View
ctjini050201	One complex but super	normal	novice	intermediate	<input type="checkbox"/>	Append	MoveUp	MoveDown	View
ctjini040101	JINI resource	normal	novice	advanced	<input type="checkbox"/>	Append	MoveUp	MoveDown	View

Add vertex Remove selected vertex

Submit configuration

Session 2: Know some theory
Goal: a peek into what the key components are.
Keywords:
 JavaSpace
 discover
 join(/write)
 read
 take

loID	loTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select				
ctjini010101	JavaSpace	normal	novice	advanced	<input type="checkbox"/>	Append	MoveUp	MoveDown	View

Add vertex Remove selected vertex

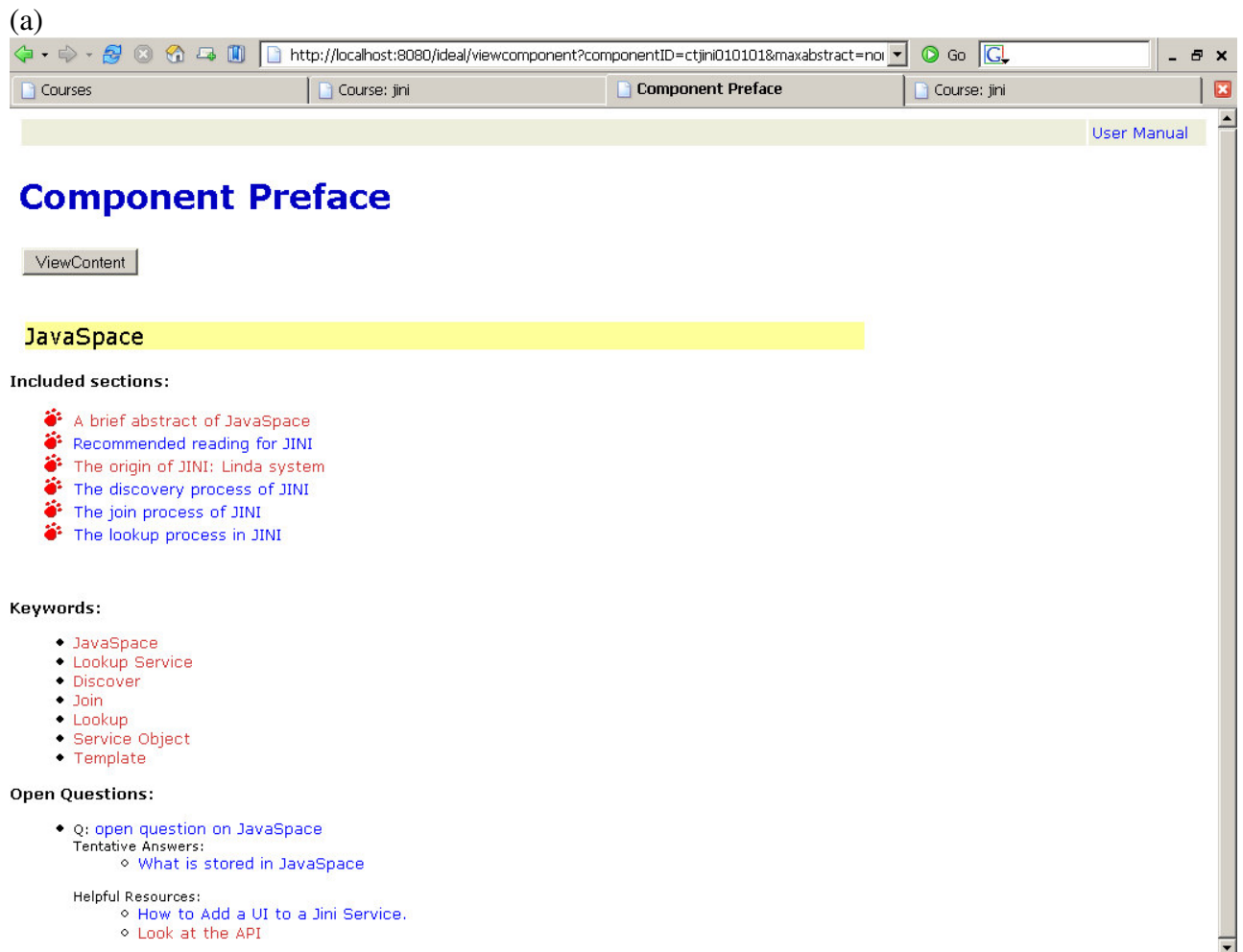
Submit configuration

Figure 5.4: IDEAL Learning Desktop displaying a component preface

(a) When the display mode of the component has the value for attribute “detail” as “normal,” the component has 6 included sections.

(b) When the display mode of the component has the value for attribute “detail” as “abstract,” the component has only 1 included section. When the display mode of the component has the value for both “difficultylevellowerdisplay” and “difficultylevelupperdisplay” as “novice,” the component has only 1 included section.

(c) The XML document of the component being displayed in (a) and (b).



(b)

The screenshot shows a web browser window with the URL `http://localhost:8080/ideal/viewcomponent?componentID=ctjini010101&maxabstract=ab:`. The browser tabs include 'Courses', 'Course: jini', 'Component Preface', and 'Course: jini'. A 'User Manual' link is visible in the top right. The main content area features a blue heading 'Component Preface' and a 'ViewContent' button. Below this, the 'JavaSpace' title is highlighted in yellow. Under 'Included sections:', there is a red icon and the text 'A brief abstract of JavaSpace'. The 'Keywords:' section lists: JavaSpace, Lookup Service, Discover, Join, Lookup, Service Object, and Template. The 'Open Questions:' section includes a question 'open question on JavaSpace' with tentative answers like 'What is stored in JavaSpace' and helpful resources like 'How to Add a UI to a Jini Service' and 'Look at the API'.

(c)

```
<?xml version="1.0" encoding="UTF-8"?>
<component id="ctjini010101" media="text" contentType="core"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ideal-xml-resources\doc\olo\schema\olo.xsd">
  <title>JavaSpace</title>
  <content>
    <vertex difficultylevellowerbound="novice" difficultylevelupperbound="expert" detail="abstract">
      <loID>snjini010101</loID>
    </vertex>
    <vertex difficultylevellowerbound="intermediate" difficultylevelupperbound="expert"
  detail="normal">
      <loID>snjini04FF02</loID>
    </vertex>
    <vertex difficultylevellowerbound="intermediate" difficultylevelupperbound="expert"
  detail="normal">
      <loID>snjini010201</loID>
    </vertex>
    <vertex difficultylevellowerbound="intermediate" difficultylevelupperbound="expert"
  detail="normal">
      <loID>snjini010102</loID>
    </vertex>
  </content>
</component>
```

```

    <vertex difficultylevellowerbound="intermediate" difficultylevelupperbound="expert"
detail="normal">
      <loID>snjini010103</loID>
    </vertex>
    <vertex difficultylevellowerbound="intermediate" difficultylevelupperbound="expert"
detail="normal">
      <loID>snjini010104</loID>
    </vertex>
  </content>
  <metadata>
    <keywords>
      <link uri="">JavaSpace</link>
      <link uri="">Lookup Service</link>
      <link uri="">Discover</link>
      <link uri="">Join</link>
      <link uri="">Lookup</link>
      <link uri="">Service Object</link>
      <link uri="">Template</link>
    </keywords>
    <openquestions>
      <openquestion>
        <question>
          <loID>snjini060101</loID>
        </question>
        <tentativeanswers>
          <tentativeanswer>
            <loID>snjini060102</loID>
          </tentativeanswer>
        </tentativeanswers>
        <helpfulresources>
          <helpfulresource>
            <link uri="http://www.artima.com/jini/jiniology/serviceui.html">How to Add a UI to a Jini
Service.</link>
          </helpfulresource>
          <helpfulresource>
            <link uri="">Look at the API</link>
          </helpfulresource>
        </helpfulresources>
      </openquestion>
    </openquestions>
  </metadata>
</component>

```

Figure 5.5: IDEAL Learning Desktop displaying a section in the same component shown in Figure 5.4

(a) The first section of a component has the full content displayed.

(b) The second section of the same component is displayed less than the first section.

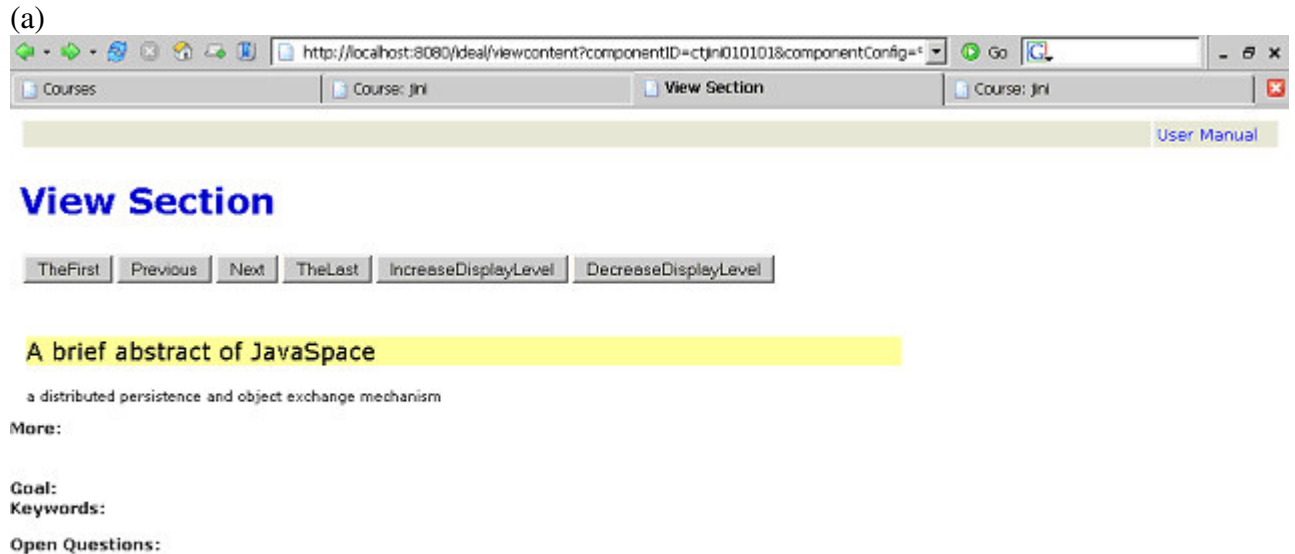
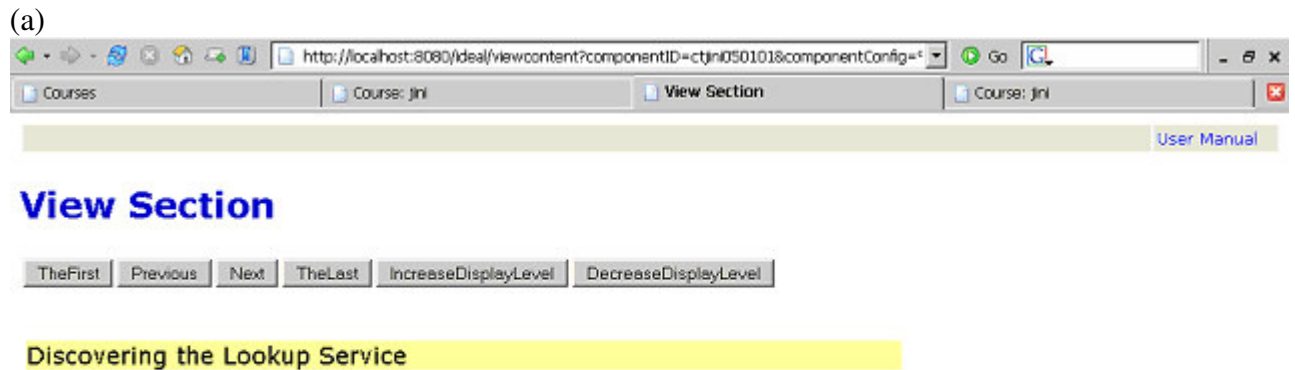


Figure 5.6: IDEAL Learning Desktop displaying a section
(a) When the display level of the section is 1
(b) When the display level of the section is 2
(c) When the display level of the section is 3
(d) When the display level of the section is 4



(c)

http://localhost:8080/ideal/viewcontent?componentID=ctjini050101&componentConfig=

Courses Course: Jini View Section Course: Jini

[User Manual](#)

View Section

TheFirst Previous Next TheLast IncreaseDisplayLevel DecreaseDisplayLevel

Discovering the Lookup Service

A local instance of the class `LookupDiscovery` multicasts a request on the local network for any lookup services to identify themselves. Get proxy of lookup service.

More:

The basic operations of discovering the lookup service are implemented by a Jini technology infrastructure software class (`LookupDiscovery`). An instance of this class acts as a mediator between devices and services on one hand and the lookup service on the other. In this example the printer first registers itself with a local instance of this class. This instance then multicasts a request on the local network for any lookup services to identify themselves. The instance listens for replies and, if there are any, passes to the printer an array of objects that are proxies for the discovered lookup services.

(d)



View Section

[TheFirst](#) [Previous](#) [Next](#) [TheLast](#) [IncreaseDisplayLevel](#) [DecreaseDisplayLevel](#)

Discovering the Lookup Service

A local instance of the class `LookupDiscovery` multicasts a request on the local network for any lookup services to identify themselves. Get proxy of lookup service.

More:

The basic operations of discovering the lookup service are implemented by a Jini technology infrastructure software class (`LookupDiscovery`). An instance of this class acts as a mediator between devices and services on one hand and the lookup service on the other. In this example the printer first registers itself with a local instance of this class. This instance then multicasts a request on the local network for any lookup services to identify themselves. The instance listens for replies and, if there are any, passes to the printer an array of objects that are proxies for the discovered lookup services.

Goal:

Keywords:

Open Questions:

5.2 View by Patterns

In addition to rendering the course map according to author/instructor's configuration, the system can also generate a course map according to some patterns. According to a certain pattern, components in the cloud can be selected, and certain content types of components can be grouped into several standard sessions. The session's title and goal are predefined for this pattern by the system. The component's display mode will also be configured automatically according to the pattern.

In our system, we have the following standard sessions that correspond to certain goals of learning:

“shallow knowledge,” “core knowledge,” “practical knowledge,” and “distributed knowledge.”

Components of content types “resource” and “example” are grouped into “shallow knowledge” session, which gets the learner to “know something about this topic before serious study.” Components of content type “core” are grouped into session “core knowledge,” which lets the learner “study the theory and learn the logics in this topic.” Components of content type “practice” are grouped into session “practical knowledge,” which lets the learner “have hands-on practice and get the certainty of this topic.” Components of content type “remote” are grouped into session “distributed knowledge,” which lets the learner “do some distributed learning to learn about the underlying knowledge from other related topics.”

For example, we have a bird view pattern in our system. Bird view pattern is used when the learner wants to have a bird view of the whole course. Regardless of the learner’s skill level, the learner often wants to have a bird view of the course. So the components related to a topic are grouped into four sessions as shown in Figure 5.7. Only the abstract content of the components will be displayed, regardless of the difficulty level (ranging from “novice” to “expert”).

Figure 5.7: IDEAL Learning Desktop displaying a “bird view” of the course

Session 1: Shallow Knowledge
Goal: Get to know something about this topic before serious study.

IoID	IoTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select
ctjini050101	One simple JINI exampl	abstract	novice	expert	<input type="checkbox"/>
ctjini050201	One complex but super	abstract	novice	expert	<input type="checkbox"/>
ctjini040101	JINI resource	abstract	novice	expert	<input type="checkbox"/>

Session 2: Core Knowledge
Goal: Study the theory. Learn the logics in this topic.

IoID	IoTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select
ctjini010101	JavaSpace	abstract	novice	expert	<input type="checkbox"/>

Session 3: Practical Knowledge
Goal: Hands-on practice. Get the certainty of this topic.

IoID	IoTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select
ctjini070101	API of JINI	abstract	novice	expert	<input type="checkbox"/>
ctjini070201	Sample Code of JINI	abstract	novice	expert	<input type="checkbox"/>
ctjini070301	Organization of JINI cod	abstract	novice	expert	<input type="checkbox"/>

Session 4: Distributed Knowledge
Goal: Do some distributed computing to learn about the underlying knowledge from other related topics.

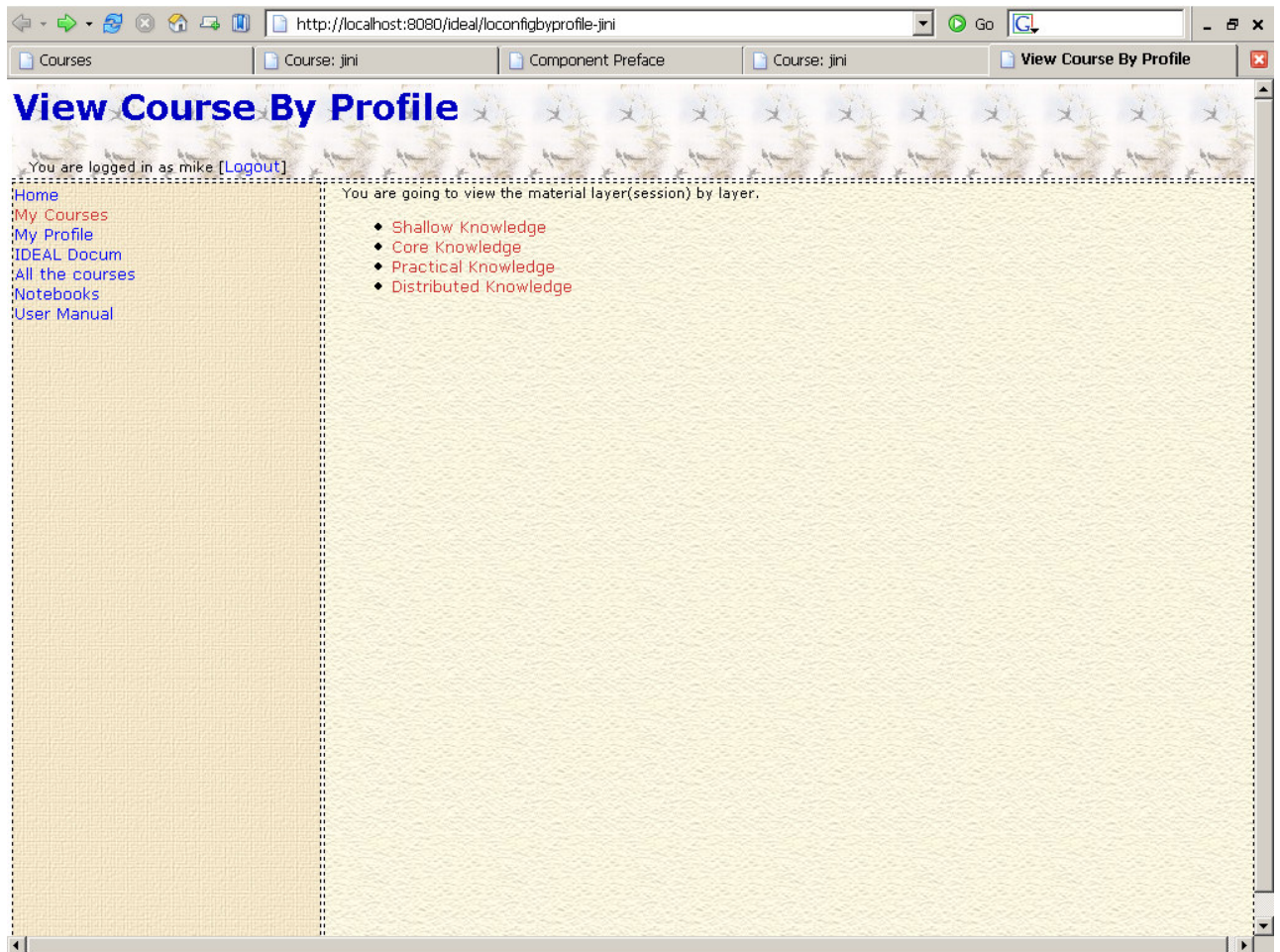
IoID	IoTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select
------	---------	-----------	---------------	---------------	--------

In addition to the bird view pattern, we have novice pattern and expert pattern in our system. The novice pattern only selects components of low difficulty level and only displays the low difficulty level content of the component. The expert pattern only selects components of high difficulty level and only displays the high difficulty level content of the component.

5.3 View by Profile

The system can also generate course material session by session according to the learner's skill level. Each learner has a learner profile stored in the database, and a skill level of a corresponding component is stored in the profile. The student can jump into any session as shown in Figure 5.8 according to her current learning needs, and her current skill level will be used in generating this session at runtime.

Figure 5.8: IDEAL Learning Desktop displaying a course by profile



For example, Figure 5.9 shows the “shallow knowledge” sessions for two different learners. Figure 5.9(a) shows how the “shallow knowledge” session is displayed for a

learner with high skill level in the related components, while Figure 5.9(b) shows the “shallow knowledge” session displayed for a learner with low skill level in the related components. Only difficulty level attributes are used in adapting content.

Figure 5.9: IDEAL Learning Desktop displaying a “shallow knowledge” session by profile
(a) for a high skill level learner
(b) for a low skill level learner

(a)

Course: jini

Course Map

Notice: you can further configure the course the way you want to view

Session:Shallow Knowledge
Goal: Get to know something about this topic before serious study.

loID	loTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select
ctjini040101	JINI resource	normal	advanced	expert	<input type="checkbox"/>

Add vertex Remove selected vertex

(b)

Course: jini

Course Map

Notice: you can further configure the course the way you want to view

Session:Shallow Knowledge
Goal: Get to know something about this topic before serious study.

loID	loTitle	MaxDetail	MinDifficulty	MaxDifficulty	Select				
ctjini050101	One simple JINI exampl	normal	beginner	intermediate	<input type="checkbox"/>	Append	MoveUp	MoveDown	View
ctjini040101	JINI resource	normal	novice	beginner	<input type="checkbox"/>	Append	MoveUp	MoveDown	View

Since we give learners the options to further configure the course map or change the display level of sections, we allow learners to be actively involved in shaping the learning environments, which is highly demanded in a constructivist learning environment (B. Bannan-Ritland, et al., 2002).

As we make the authoring of learning object quite easy, learners can also be involved in authoring of learning objects. Since knowledge is ill-structured and everyone holds her unique view of knowledge, it is important that the learner is able to construct her own knowledge. And since cooperation among authors becomes possible and relatively easy in our system, the work becomes easier for the author by reusing others' work.

Bannan-Ritland points out that learning objects designed with guidelines of constructivism still can provide electronic performance support (Bannan-Ritland, et al., 2002). In our system, the system helps rendering learning material with various patterns and help the learner adapt the material at run time. Such function is highly desired in active learning, but very difficult to achieve with traditional knowledge carriers.

Chapter Six--Implementation

Using the N-tier architecture, the IDEAL e-learning system is separated into data layer, logic layer and presentation layer as shown in Figure 6.1.

The data layer of the system is implemented using XML database Apache Xindice. Learning objects together with learner information and other information are stored in the XML database. We use Apache Axis to implement Web services, which wrap the database up to provide access to the XML documents. These Web services are named XMLData Web services. These services support the following interactions with the database:

- authenticate
- executeQuery
- getDocument
- getDocumentCount
- insertDocument
- updateDocument
- removeDocument
- listCollections
- dropCollection
- createCollection
- listResources
- executeUpdate

The logic layer of the system is represented by other Web services, which are built upon XMLData Web services to provide semantic interaction with the system. The LearnerInformation Web service provides services to interact with the learner profile. Open learning object (OLO) Web service provides services to interact with the learning material or learning objects.

In addition to inserting, retrieving, and updating learning objects, the OLO web service provides services to adapt learning objects to the learner.

XSL stylesheet is applied to the cloud XML document to generate course maps according to various patterns or to generate a session according to the learner's skill level. XSL stylesheet is also applied to adapt a raw component according to the display mode. Section is also adapted by applying XSL stylesheet against the raw section according to the display level, which is computed according to the component_vertexAttri of this section within a component.

IDEAL Learning Desktop, a multi-role portal, is the presentation layer of the system. It is implemented with Apache Cocoon, a Web development framework. Cocoon Forms is used to render learning objects. WSIF is used to conveniently invoke remote Web services.

Each of the three layers can be physically located in different servlet containers.

Below we will go through a more detailed description of the implementation.

Apache Xindice, a native xml database, is used to store, retrieve and update our xml documents. A servlet named IDEALXindiceServlet is written to replace XindiceServlet coming with Apache Xindice. IDEALXindiceServlet is loaded on the start up of servlet

container and it starts the xml database server. But unlike in XindiceServlet, no xmlrpc server is started in IDEALXindiceServlet, so no http request will be responded. Furthermore there is no servlet-mapping for IDEALXindiceServlet in the applications deployment descriptor, and IDEALXindiceServlet was written to ignore all http requests. The embedded Xindice can be only invoked by XMLData in the same container using XMLDB API. XMLData is exposed as soap web service by Apache Axis servlet. AXIS also provides WSDL description for the web service. Axis web service has handlers in both request and response flows. A custom authentication handler authenticates user from the request flow. A custom authorization handler deployed on both request and response flows authorizes operations against xml database. The authentication handler looks for username/password at the header and method parameters of soap request message to authenticate user. Once authenticated, a session will be established with the client. WSIF is used to conveniently invoke web service. WSIF allows invoking web service through their WSDL description, regardless of how the web service is implemented and accessed. A stub is generated at the client side that can be used to invoke web service conveniently.

Built upon XMLData web service, higher-level web services are implemented to manage learner information and learning objects. They invoke XMLData web service through WSIF and their services are exposed by Apache Axis as web services.

The web service to manage learning objects provides semantics for application developers to interact with learning objects. Below is a brief description of the key services implemented:

- to store, retrieve, update, delete raw section, component, cloud, course and parts of them;

- directory to list all courses in the system, and all courses of a learner. It is rendered in the format of xml string;
- to get the default configured course map, or get the learner configured course map if there is one. It is rendered in the format of xml string;
- to store learner configured course map into the xml database;
- to provide a cloud of a topic. Instead of simply getting a cloud xml document of the topic, it also gets the title and content type of the included components and plugged them into the cloud xml document in the output;
- to provide adapted component according to the context. The context is given in the format of xml string. The context can be provided as the session ID, Course ID the component is in, and the learner ID. In this case, the display mode will be obtained based on the context. The context can also be directly the display mode of the component. In either case, XSL stylesheet is applied to adapt a raw component according to the display mode. The Java methods providing these services are as below:
 - `public String GetComponentPrefaceFromContext(String username, String componentID, String sessionID, String closureID)throws Exception`
 - `public String GetComponentPrefaceFromConfig(String componentID, String componentConfig)throws Exception`
- to provide adapted sections given the context. The context can be the index of the section within a component, the component ID, the session ID, the course ID and the learner ID. The context can also be already adapted component

and the index of the section within the component. Or the context can be component ID, its display mode and the index of the section within the component. In whichever case, the section ID and the display level will be computed. The context can also be display level itself. In all cases, XSL stylesheet is applied to adapt a raw section according to the display level;

- to generate course map by pattern or to generate session by session according to learner's skill level; and
- other services.

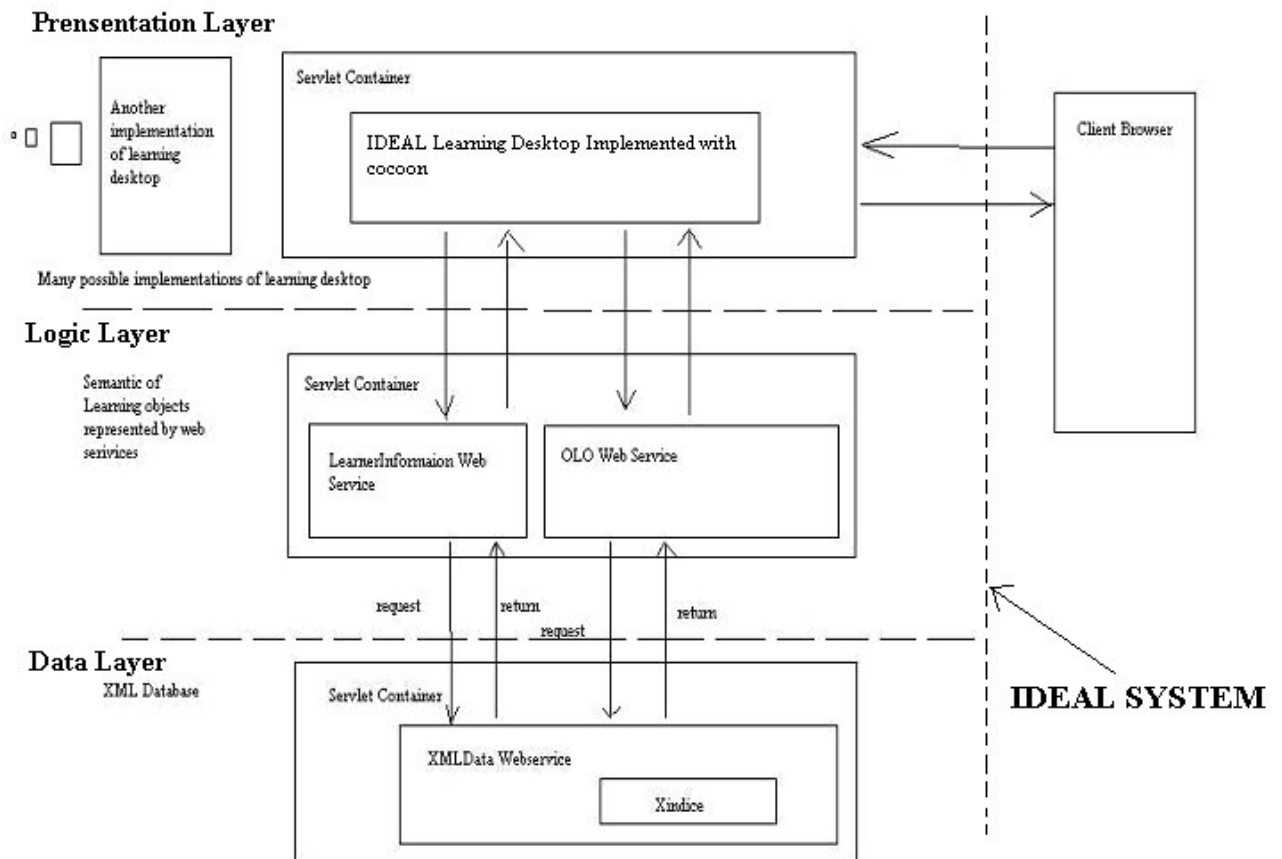
The web service to manage learning objects can be deployed either at the same container of XMLData web service, or at a different container.

IDEAL learning desktop is implemented using Apache Cocoon. Built upon Cocoon, IDEAL learning desktop is an extensible web publication framework. Its metaphor is a portal, in which various users with different roles will be presented different menus. It is a comprehensive learning environment in which various roles of users (learner, instructor, administrator and etc.) conduct their work or learning.

Apache Cocoon is a web development framework built upon the concept of separation of concerns and component-based web development (using the idea of Inversion of Control or IoC). It separates the concerns of management, logic, content and style so they don't conflict with each other. Cocoon implements these concepts around the notion of "component pipelines", which usually include a Generator, Transformers and a Serializer. This makes it possible to use a Lego(tm)-like approach in building web solutions, assembly components into pipelines without requiring programming. Information can be inserted into the middle of the pipeline.

Custom generators are used to obtain content from various resources including by invoking web services. To implement a portal for various roles of user, we can define users' role and related actions in an xml file. We can use an action2menu stylesheet to select the actions and transform into menu information. We can insert the menus and header information (so-called navigation) into the content and transform the integrated page into html of certain styles by applying XSL stylesheets. Cocoon session transformer is used to insert session related element such as username.

Figure 6.1: Architecture of IDEAL Learning Object System



We use the cocoon authentication framework to authenticate the user. An authentication generator is written to be used in the cocoon authentication framework. Cocoon sitemap pipelines that need to be protected are put inside the authentication action. Once the authentication generator authenticates the user, it will set up the user session context (username, password, userrole, profile).

Thus we have a web publication framework, which separates different concerns and allows parallel development. The learning desktop implemented has a clean logic, and it is easy to make changes.

Learning object rendering in IDEAL learning desktop is implemented using Cocoon Forms, together with Control Flow (continuation-based page flow that hides the complexity of request/response processing and is cleanly separated from the view and data).

We use the flow script to provide the programming functionality so we can call the web service (the cocoon form control flow script requests the learning material on behalf of the learner) very flexibly and control the rendering of course very freely (such as calculate the next index or the previous index, get the component size and determine whether the end of the component has been reached, or if no sections are included in a component, clicking on the view content button only redirect to the same page).

We use the Cocoon Forms to provide an easy and clean implementation of the forms used to render various kinds of learning material (course map, cloud, session, component, section) to the learner.

Chapter Seven--Conclusion

Our design demonstrates a possibility of using constructivism learning theory to guide the design of learning objects so that constructivism and objectivism can be integrated together. Through our design, the cooperation among learning object authors becomes possible and easy, and thus learners can also actively participate in the construction of learning objects. Learning objects are rendered by some patterns and the learner can further configure the course and have the material adapted to her needs at run time. We provide a way to allow the learner to grasp the whole picture of the course in the fastest way by taking control of the learning and by the support of our system. Such ease to view the learning material iteratively in different ways will greatly assist learners to learn efficiently in the real constructivist learning environment. Such a direction should be the future of learning object research. The significance and essence of this direction can best be described in the following quote from Bannan-Ritland's paper:

“Placing the power of this technology in the learner's hands may indeed reveal the true potential of this technology for learning. As Scardamalia and her colleagues (1995) have so eloquently stated, ‘... it is not the computer that should be doing the diagnosing, the goal-setting and the planning, it is the student. The computer environment should not be providing the knowledge and intelligence to guide learning, it should be providing the facilitating structure and tools that enable students to make maximum use of their own intelligence and knowledge’ ” (Bannan-Ritland, et al., 2002).

Designing learning objects has been a very struggling but enjoyable experience for me. As there are a lot of development and specification of learning objects, I am not aware of any design and implementation that demonstrates a significantly improved learning experience and meets the expectation of learning objects. As the concept of learning object is certainly very attractive and consistent with many people's view of

knowledge and learning experience, to probe deeply into such a concept and to experiment with it provide me more experience and insights into many facets of this issue. Obviously, a lot of concerns regarding learning object, especially learning object as a representation of knowledge, should be very good research topics, yet few people have addressed these topics. Constructivism, as the latest learning theory, is rarely adopted in the design of learning objects. Realizing the insufficient research on learning objects in association with learning theory, IDEAL learning object design is more explorative in nature, trying to demonstrate the possibility of applying constructivism to learning object design. It is expected that with the increase of breadth and depth of learning material, our system's advantage will be more obvious in facilitating the learner to have a better control of the learning material and their pace and strategy of learning.

REFERENCES

- Alexander C. (2003). Nature of Order, Center for Environmental Structure.
- Apache Xindice: <http://xml.apache.org/xindice/>.
- Apache AXIS: <http://ws.apache.org/axis/>.
- Apache Cocoon: <http://cocoon.apache.org/>.
- Apache WSIF: <http://ws.apache.org/wsif/>.
- Bannan-Ritland B., Dabbagh N., and Murphy K. (2002). Learning object systems as constructivist learning environments: related assumptions, theories and applications. Instructional Use of Learning Objects, Agency for Instructional Technology. Also available at <http://www.reusability.org/read/>.
- Barab S.A. and T. Duffy (2000). From practice fields to communities of practice. In D. Jonassen and S.M. Land. (Eds.), Theoretical Foundations of Learning Environments, pp. 25–56, Mahwah, NJ: Lawrence, Erlbaum Associates.
- Bereiter C. and Scardamalia M. (1989). Intentional learning as a goal of instruction. In L. B. Resnick's (Ed.), Knowing, learning, and instruction, pp. 361-391, Hillsdale, NJ: Lawrence Erlbaum Associates.
- Brown J.S., Collins A., and Duguid P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18: 32–42.
- Brown J.S. and Duguid P. (1993). Stolen knowledge. Educational Technology (Special Issue on Situated Learning), 10–14, Embodied, Situated agents, pp. ix, 288, Hillsdale, NJ: Lawrence.
- Henning P.H. (2004). Everyday cognition and situated learning. In Jonassen D.H. (Ed.), Handbook of Research on Educational Communications and Technology, pp. 143-168, Mahwah, New Jersey: Lawrence Erlbaum Associations.
- Interactive Computation: http://en.wikipedia.org/wiki/Interactive_computation.
- Jonassen D.H. (1993). Designing constructivist learning environments. In Duffy T.M., Lowyck J., and Jonassen D.H. (Eds), The Design of Constructivistic Learning Environments: Implications for Instructional Design and the Use of Technology, Heidelberg, FRG: Springer-Verlag.
- Jonassen, D.H., Marra R.M., and Palmer B. (2004). Epistemological development: an Implicit entailment of constructivist learning environments. In N.M. Seel and S. Dijkstra (Eds.), Curriculum, Plans, and Processes in Instructional Design, pp. 75-88, Mahwah, NJ: Lawrence Erlbaum Associates.
- LTSC (Learning Technology Standards Committee), Standards for Learning Object Metadata, IEEE, 2000. Available at <http://ltsc.ieee.org/doc/wg12/LOMv4.1.htm>.

- Orrill C. (2002). Learning objects to support inquiry-based, online learning. Instructional Use of Learning Objects, Agency for Instructional Technology. Also available at <http://www.reusability.org/read/>.
- Savery J.R. and Duffy T.M. (1995). Problem based learning: an instructional model and its constructivist framework. Educational Technology, 35(5): 31-37.
- Shang Y., Shi H., and Chen S. (2001). An intelligent distributed environment for active learning. ACM Journal of Educational Resources in Computing, 1(2):1-17.
- Shi H., Rodriguez O., Shang Y., and Chen S. (2002). Integrating adaptive and intelligent techniques into a web-based environment for active learning. In C. T. Leondes (Ed.), Intelligent Systems: Technology and Applications, Volume 4, Chapter 10, pp. 229-260, CRC Press, Boca Raton, FL.
- Wegner P. (1997). Why interaction is more powerful than algorithms. Communications of the ACM.
- Wiley D. (2002). Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy. Instructional Use of Learning Objects, Agency for Instructional Technology. Also available at <http://www.reusability.org/read/>.