

Bridging O-Telos and XML with XML Schema: the Authoring Environment for KBS Adaptive Hyperbook

Changtao Qu

Institute of Computer Engineering
University of Hannover

Appelstr. 4, D-30167, Hannover, Germany

E-mail: qu@kbs.uni-hannover.de

Wolfgang Nejdl

Institute of Computer Engineering
University of Hannover

Appelstr. 4, D-30167, Hannover, Germany

E-mail: nejdl@kbs.uni-hannover.de

Abstract

KBS Adaptive Hyperbook is a framework designed for modeling, organizing, and maintaining distributed hypermedia resources on the Web with the purpose of supporting Web-based distance education. Since the Hyperbook system is implemented based on a sophisticated meta modeling language (O-Telos), its authoring depends to a great extent on the Hyperbook developers instead of the lecturers who have rich teaching experience but generally have no knowledge of O-Telos and meta modeling. This has become a notable obstacle to enrich the content of existing Hyperbooks and further construct new Hyperbooks which are applicable to different specialties. In this paper we present a novel approach which can achieve the Schema-level transformation between O-Telos Schema and W3C standard XML Schema. Through transforming Hyperbook data models, which are originally represented in O-Telos, into XML schemas, we can provide a standalone Hyperbook authoring environment which is able to directly utilize some XML-Schema-aware XML visual editors as front-end GUI, and enable the lecturers to easily accomplish Hyperbook authoring process.

Keywords: *Hypermedia, Meta modeling, O-Telos, eXtensible Markup Language, XML Schema*

1. Introduction

KBS Adaptive Hyperbook (Hyperbook for short) is the major achievement of our project “Virtual Campus Hannover – Hildesheim – Osnabrueck” [12] [18]. It is designed as a framework for modeling, organizing, and maintaining distributed hypermedia resources on the Web

with the purpose of supporting Web-based distance education. In addition, it provides also adaptation facilities based on user knowledge about the delivered content. As a typical application scenario, Hyperbook models courses according to the semantic relationships of course content and guides students through the courses according to their different learning purposes and knowledge levels [12]. Since the summer semester 1999, a specific Hyperbook has been used for the teaching of a joint CS1 course “Introduction to Java Programming” (Java-Hyperbook for short) at three German universities and one university in Italy [13]. While getting lots of positive feedback from lecturers and students, we have also become aware of some shortcomings of Hyperbook during the teaching practice.

One notable shortcoming of Hyperbook is its lack of a suitable authoring environment.

Teaching and learning is naturally an interactive and iterative process. Consequently, the content and structure of Hyperbook need to be permanently revised and adjusted during the teaching and learning process. As a typical example, the lecturers continually need to integrate the latest learning materials into Hyperbook in order to reflect technology development, or modify a part of Hyperbook content according to the feedback from students. Because at present the “original” authoring environment for Hyperbook is directly based on the syntax of O-Telos and takes a text editor as the front-end user interface, the lecturers who have generally no knowledge of O-Telos have encountered big problems while accomplishing the Hyperbook authoring process. Moreover, through using a generic conceptual model to represent courses and teaching materials, Hyperbook is really designed to be applicable to different specialties besides computer science. Without a user-friendly authoring environment this goal is impossible to achieve when lecturers of other

specialties become the leading authors of the Hyperbook system.

Inspired by the rapid spreading of the XML (eXtensible Markup Language)[5] standard, and particularly by some of its new extensions such as XSLT (eXtensible Stylesheet Language Transformations)[7] and XML Schema [3][9][19], as well as its strong industry support such as commonly used XML parsers, XSLT processors [2], and XML-Schema-aware XML visual editors [1], we decided to implement the Hyperbook authoring environment utilizing XML and XML tools. Simply put, in order to accomplish Hyperbook authoring, Hyperbook data models are firstly transformed from O-Telos schemas into XML schemas. Then, taking advantage of some XML-Schema-aware XML visual editors as front-end GUI (Graphical User Interface), the lecturers can accomplish Hyperbook authoring, namely, instantiate the Hyperbook data models, restricted and guided by XML schemas which represent the equivalent Hyperbook data models in XML visual editors. Finally, after the authoring process is completed, the instances of Hyperbook data models are transformed back from XML into O-Telos using XSLT and further stored in the Hyperbook system.

The implementation of the Hyperbook authoring environment through this approach has proven to be advantageous in two facets. On the one hand, as some XML-Schema-aware XML visual editors can “understand” the Hyperbook conceptual model, we are able to hide the lecturers from most of Hyperbook modeling details and present them with structure-based editors which can to a certain degree guide the Hyperbook authoring process. On the other hand, since some commercial XML visual editors can be directly used as the front-end GUI of the authoring environment, we can easily achieve a rapid application due to greatly reduced development work. Actually, this approach has efficiently extended the functionality of “original” Hyperbook system.

2. General Design

Hyperbook adopts a popular modeling methodology, namely, taking a general Meta Model as the basis to represent its Domain Model. Two kernel classes in the Hyperbook Meta Model are “Concept” and “Relation”. “Concept” defines the basic unit and primitive structural element of Hyperbook, while “Relation” defines relationships which can exist between “Concepts”. Because the detailed discussion about Hyperbook data models is beyond the scope of this paper, we introduce here only these two classes which are closely related to our following discussion. For more description of Hyperbook

data models please refer to our previous publications [11][12][18].

With regard to the Hyperbook authoring, the lecturers primarily need to interact with Hyperbook Domain Model. The principal task of the lecturers is to possibly construct a Domain Model and then instantiate this Domain Model with concrete teaching materials. Although constructing the Domain Model is an important task and the lecturers usually need to provide some design ideas or requirements concerning about domain knowledge in this process, this task is not the main body of Hyperbook authoring and is usually completed by Hyperbook developers. Therefore our authoring environment basically focuses on instantiating the Hyperbook Domain Model during the authoring process.

In figure 1 we illustrate the general infrastructure of Hyperbook and Hyperbook authoring environment.

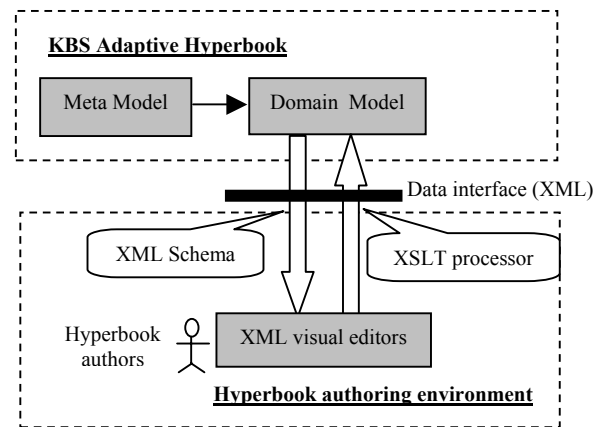


Figure 1. General infrastructure of Hyperbook and Hyperbook authoring environment

The principal design idea of the Hyperbook authoring environment is to provide it as a standalone module of the Hyperbook system through taking XML as the standard data interface. Firstly, Hyperbook data models originally represented in O-Telos are transformed into the corresponding XML schemas. These XML schemas are then used in the Hyperbook authoring environment and serve as the input of some XML-Schema-aware XML visual editors, which act here as the front-end GUI of the Hyperbook authoring environment. These XML visual editors can on the one hand graphically display all Hyperbook data models to help the lecturers understand the Hyperbook structure, on the other hand, they can also help the lecturers create a better authoring experience because these XML visual editors themselves can “understand” Hyperbook data models and are able to intelligently and constantly show “hints and tips” to guide the lecturers to instantiate the Hyperbook Domain Model without “violating the rules”. After the authoring process

is completed, all instances of the Hyperbook Domain Model are transformed back from XML into O-Telos using XSLT and are further stored in the Hyperbook system.

3. Bridging O-Telos and XML

3.1. Data model of XML

As an object-oriented meta modeling language, O-Telos provides a powerful mechanism to define data types and structures as well as to check whether data conforms to those definitions (so-called O-Telos Schema). Similarly, XML also provides a commonly used counterpart mechanism: DTD (Document Type Definition). Although DTDs are quite comprehensible and easy to use, it is impossible to represent all relationships defined in Hyperbook data models, including the commonly used inheritance relationship, with DTDs due to their weak representation capability. Of course, it is also possible to use some extension mechanisms to strengthen the representation capability of DTDs [10][15], however, such an approach will unavoidably damage the extensibility and interoperability of the Hyperbook authoring environment, especially its interoperability with some XML visual editors.

Fortunately, there is at this time an advanced mechanism which can be used to define data types and structures of XML on a higher level compared to DTDs. This mechanism is XML Schema [3][9][19], a recent W3C Proposed Recommendation originally designed to remedy many of the problems and limitations of DTDs.

In general, XML Schema enhances the representation capabilities of DTDs in two ways. First, XML Schema is able to express more data types. Second, XML Schema is able to express more relationships between data types.

Actually, it is just the latter that makes great sense to our system design. XML Schema provides a mechanism to derive new data type definitions on the basis of old ones. Such sort of derivation can be either “extension”, which means the derived types add elements and/or attributes to those already defined in the base types, or “restriction”, which means the derived types add restrictions or limitations to the original data types. Taking the representation of inheritance relationship as an example, XML Schema is able to not only represent “usual” inheritance using “extension”, but also can represent “tailored” inheritance using “restriction”. As we will discuss in section 3.2., the main body of O-Telos data model is the semantics of object structure. Using XML Schema, we can represent the equivalent semantics of object structure of O-Telos. This makes it possible to realize the Schema-level transformation between the both.

3.2. Data model of O-Telos

O-Telos is based on the knowledge representation language Telos [17] with an especial capability to describe objects, classes, meta-classes, and meta-meta-classes within the same framework. This framework of the data model can be divided into two parts. The main body is the semantics of object structure. The other part is constructed from deduction rules, integrity constraints, and query classes which are used for knowledge inference and handling. The whole data model of O-Telos is based on the O-Telos Knowledge Base (KB), which is a finite set of interrelated propositions and can be represented as:

$$KB = \{P(\text{oid}, x, l, y) \mid \text{oid}, x, y \in \text{ID}, l \in \text{LABEL}\}$$

where oid has key property within the knowledge base, ID is a non-empty set of identifiers with a non-empty subset LABEL of names. The components oid, x, l, y are called identifier, source, label (or name), and destination. They can be simply read as “The object x has a relationship called l to the object y”.

There is a natural interpretation of a set of propositions as a semantic network in O-Telos. Here three patterns of propositions are distinguished and are named as follows [14][17]:

Individual: $P(\text{oid}, \text{oid}, l, \text{oid})$, which can be read as “oid is an object with name l”.

InstanceOf: $P(\text{oid}, x, \text{instanceof}, y)$, which can be read as “x is an instance of class y”.

IsA: $P(\text{oid}, x, \text{isa}, y)$, which can be read as “x is a specialization of y”.

In addition, O-Telos imposes some structural axioms on knowledge bases, e.g. referential integrity, correct instantiation, and inheritance. They are linked to pre-defined classes that are automatically part of each O-Telos KB [14][17].

As a matter of fact, O-Telos’ data model is strictly based on a well-axiomatized object-oriented framework. As far as the semantics of object structure is concerned, the commonly used relationship between data types is inheritance (IsA). In addition, another relationship (InstanceOf) is used to represent hierarchical instantiation between objects from meta-meta-class until token. Through defining appropriate rules, we can achieve Schema-level transformation between this part of data model of O-Telos and XML Schema. However, in comparison with the semantics of object structure, the definition of deduction rules, integrity constraints, and query classes in O-Telos’ data model involves more sophisticated syntax whose discussion is already beyond the scope of this paper. Actually, the Schema-level transformation of deduction rules, integrity constraints, and query classes between O-Telos and XML Schema is

very difficult or even impossible to realize using “usual” XML technologies.

3.3. From O-Telos to XML: XML Schema

Although the 100% transformation between O-Telos data model and the data model of XML is not possible, a scaled-down transformation of data model, namely, the Schema-level transformation of the object structure from O-Telos to XML Schema is already sufficient to the Hyperbook authoring environment. As we’ve mentioned in section 2, in the Hyperbook authoring environment the lecturers mainly need to interact with instances of Hyperbook Domain Model, therefore the class and relation declarations are already sufficient. This part of transformation can be realized through following rules.

Rule 1: InstanceOf: $P(oid,x,instanceof,y)$ “InstanceOf” represents hierarchical instantiation relationship between “x” and “y”. In XML Schema “y” is expressed as “complexType”. “x” is expressed as “element” with an attribute “name”, whose value is the LABEL of “x”, and an attribute “type”, which refers to “y”. If “x” is a type of class (meta-class, simple-class, etc.), it will be further expressed as “complexType”. If “x” is a type of token, it is “finally” expressed as “element”.

Rule 1:

```
<complexType name="y"> ..... </complexType>
<element name="x" type="y" minOccurs="..." maxOccurs="..." />
```

Rule 2: IsA: $P(oid,x,isa,y)$ “IsA” represents inheritance relationship between “x” and “y”. In XML Schema “y” is expressed as parent “complexType”. “x” is expressed as the derived “complexType” of “y” with appropriate extensions.

Rule 2:

```
<complexType name="y"> ..... </complexType>
<complexType name="x">
  <complexContent>
    <extension base="y"> </extension>
  </complexContent> .....
</complexType>
```

Besides above two rules, there is also a commonly used “combined” representation style in O-Telos which can be described as “x InstanceOf y IsA z”. Here “x” is not only the subclass of “z”, but also the instance of “y”. The corresponding representation of this sort of relationship in XML Schema can be realized using the combination of Rule 1 and Rule 2. We will present a detailed transformation example including the practical use of these rules in section 4.

3.4. From XML to O-Telos: XSLT

In comparison to transforming Hyperbook data models from O-Telos schemas into XML schemas, the transformation from XML back into O-Telos is a simple process. Although Hyperbook data models have been transformed from the O-Telos schemas into XML schemas, they are used only for restricting and guiding the instantiation process in the Hyperbook authoring environment and remain untouched during the authoring process. When we transform the instances of the Hyperbook Domain Model from XML back into O-Telos, a corresponding Schema-level transformation is not necessary. Here we employ a flexible transformation language: XSLT to realize transformation from XML to structured text, which is actually in the syntax of O-Telos.

4. Working Scenario

In this section we will describe a working scenario in the Hyperbook authoring environment based on a specific Hyperbook: Java-Hyperbook. The Domain Model of Java-Hyperbook is strictly based on the Hyperbook Meta Model. The domain classes such as “course group” and “course” are subclasses directly derived from pre-defined class “Concept”. Besides, there are also some classes which are used to define semantic relationships between different “Concepts” such as “Course_Group_include_Courses”, which is the subclass of pre-defined class “Relation” (also the instance of class “Index”) and represents the relationship between two “Concepts”: “course group” and “course”. In figure 2 we list several classes of Java-Hyperbook Domain Model which are originally expressed in O-Telos.

```
Class Concept with attribute
name: String; abstract: String; id: String end

Class Relation with attribute
source: Concept; drain: Concept end

Class Displayed_Relation with attribute
nameSourceDrain: String; nameDrainSource: String;
visualisation: Boolean; priority: Integer end

Class Index isA Displayed_Relation end
Class Course_Group isA Concept end
Class Course isA Concept end

Class Course_Group_include_Courses in Index isA
Relation with attribute
source: Course_group; drain: Course; nameSourceDrain
n1: "includes courses" nameDrainSource n2: "are
included by course group" priority p1: 80 visualisation
v1: TRUE end
```

Figure 2. Part of Java-Hyperbook Domain Model represented in O-Telos

According to the transformation rules defined in section 3.3., we can transform Hyperbook data models

from O-Telos schemas into corresponding XML schemas. As an example, figure 3 lists part of classes which appear in figure 2 using XML Schema.

```

<?xml version="1.0"?>
<schema targetNamespace="http://.../Hyperbook"
xmlns:hyperbook="http://.../Hyperbook"
xmlns="http://www.w3.org/2000/10/XMLSchema">

<complexType name="Concept">
<sequence> <element name="name" type="string"/>
<element name="abstract" type="string" minOccurs="0"/>
<element name="id" type="string" maxOccurs="unbounded"/>
</sequence> </complexType>

<complexType name="Relation">
<sequence> <element name="source"
type="hyperbook:Concept" maxOccurs="unbounded"/>
<element name="drain" type="hyperbook:Concept"
maxOccurs="unbounded"/>
</sequence></complexType>

<complexType name="Displayed_Relation">
.....</complexType>

<complexType name="Index">.....</complexType>

<complexType name="Course_Group">
<complexContent>
<extension base="hyperbook:Concept">
<attribute name="token_name" type="string" use="required"/>
</extension>
</complexContent></complexType>

<complexType name="Course">
<complexContent>
<extension base="hyperbook:Concept">
<attribute name="token_name" type="string" use="required"/>
</extension>
</complexContent></complexType>

<complexType name="Course_Group_include_Courses">
<complexContent>
<extension base="hyperbook:Relation">
<sequence>
<element name="source" type="hyperbook:Course_Group"/>
<element name="drain" type="hyperbook:Course"
minOccurs="0" maxOccurs="unbounded"/>
<element name="Index" type="hyperbook:Index"/>
</sequence>
<attribute name="token_name" type="string" use="required"/>
</extension>
</complexContent></complexType>

```

Figure 3. Part of Java-Hyperbook Domain Model represented in XML Schema

After transformation, the generated XML schemas are taken as the input of an XML-Schema-aware XML visual editor Altova XML Spy 4.2 [1]. Because XML Spy can “understand” Hyperbook data models represented through XML Schema, it is not necessary for the lecturers to fully understand Hyperbook data models. On the basis of XML schemas, XML Spy can automatically generate the Java-Hyperbook structure, which consists of all essential

concepts and relations/attributes defined in the Domain Model, at the beginning of the authoring process. The remaining task for the lecturers is to instantiate the Domain Model utilizing a user-friendly GUI through filling pre-structured “tables”. Furthermore, XML Spy supports so-called intelligent editing, which can provide constant feedback about the generated instances and specify which elements or attributes can be inserted in what position. It can also constantly validate the element context the authors are working on and immediately show any errors in element structure. This function provides the authors with a much better authoring experience, as syntax and semantic errors are not possible anymore. Figure 4 shows the structure of Java-Hyperbook which is automatically generated by XML Spy at the beginning of the authoring process.

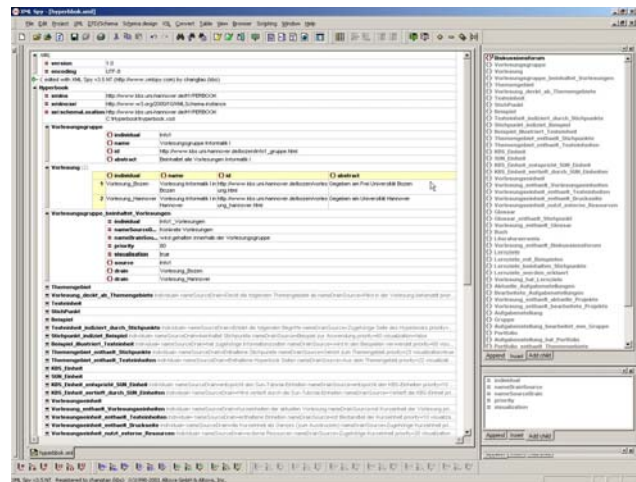


Figure 4. The Java-Hyperbook structure automatically generated by XML Spy

5. Discussion

In recent years a number of Adaptive Hypermedia Systems (AHS) have been successfully developed [4][6][8][16]. In comparison with the research on AHS and some general AHS frameworks, the authoring environments for AHSs have received fewer attention. Since AHSs usually need to adopt some sophisticated modeling methodologies and modeling languages to realize their adaptability, authoring AHS has always been a tedious task which in most cases has to be accomplished by technically skilled developers. In addition, due to the diversity of AHSs there are obviously no specialized authoring tools and/or languages which naturally fit into a general open AHS architecture.

XML and XML Schema provide us with a new possibility to bridge the gap between AHS and AHS authoring environments. Although XML and XML

Schema are not very suitable to be directly used to construct AHSs due to their weak representation capability in comparison to other modeling languages (e.g., O-Telos) while representing deduction rules or integration constraints, they are very suitable to be used to construct AHS authoring environments due to their simple syntax. As we've presented in this paper, in most cases the representation capability of XML Schema is already sufficient to provide a scaled-down data model for the construction of AHS authoring environments.

6. Conclusion

Meta-data modeling is one of the popular approaches to constructing AHSs. While some modeling methodologies have greatly improved the extensibility, reusability, and maintainability of AHSs, they also have to a certain degree increased the complexity of the systems and made the authoring process of AHSs more complicated due to the introduction of some complex data models. Our approach presented in this paper focuses on simplifying the authoring process of AHSs through bridging a sophisticated meta modeling language O-Telos with XML. Taking advantage of XML, XML Schema and XSLT, we can construct a standalone authoring environment which has extended the functionality of our existing AHS without shaking the system foundation. Since O-Telos is a typical conceptual modeling language and also the modeling methodology adopted in the Hyperbook overlaps with other AHSs in several aspects, our approach might provide some beneficial inspirations to the design and implementation of the authoring environments for other AHSs.

7. References

- [1] Altova Corp., XML Editor: XML Spy 4.2, <http://www.xmlspy.com/>
- [2] Apache Software Foundation, Xalan 1.2.2 for Java, <http://xml.apache.org/xalan/index.html>
- [3] Biron, P. V., and Malhotra, A., XML Schema Part 2: Datatypes, <http://www.w3.org/TR/xmlschema-2/>, Oct. 2000.
- [4] Boyle, C., and Encarnacion, A. O., MetaDoc: An adaptive Hypertext reading system, User Modeling and User-adapted Interaction, Vol. 4, Kluwer Academic Publisher, 1994.
- [5] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., Extensible Markup Language (XML) 1.0 (2nd Edition), <http://www.w3.org/TR/2000/REC-xml-20001006>, Oct. 2000.
- [6] Brusilovsky, P., Schwarz, E., and Weber, G., ELM-ART: An intelligent tutoring system on World Wide Web, in Proc. of third Int. Conf. on Intelligent Tutoring System, Montreal, Canada, 1994.
- [7] Clark, J., XSL Transformations (XSLT) Version 1.0, <http://www.w3.org/TR/xslt>, Nov. 1999.
- [8] De Bra, P., Houben, G. J., and Wu, H., AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, in Proc. of ACM Hypertext and Hypermedia 1999, Darmstadt, Germany, 1999.
- [9] Fallside, D. C., XML Schema Part 0: Primer, <http://www.w3.org/TR/xmlschema-0/>, Oct. 2000.
- [10] Fan, W., and Simeon, J., Integrity Constrains for XML, in Proc. of the 19th ACM SIGMOD- SIGACT- SIGART symposium on Principles of database systems, Dallas, USA, 2000.
- [11] Fröhlich, P., Henze, N., and Nejd, W., Hyperbook Data Modeling, in Proc. of Int. Conf. on Electronic Publishing, Document Manipulation and Typography. Saint Malo, France, Apr. 1998.
- [12] Henze, N., Nejd, W., and Wolpers, M., Modeling Constructivist Teaching Functionality and Structure in the KBS Hyperbook System, in Proc. of Computer Supported Collaborative Learning Conference (CSCL'99), Stanford, USA, Dec. 1999.
- [13] Institute of Computer Engineering at University of Hannover, KBS Adaptive Hyperbook: "Introduction to Java Programming", <http://www.kbs.uni-hannover.de/Hyperbook>
- [14] Jarke, M., Gallersdörfer, R., Jeusfeld, M., Staudt, M., and Eherer, S., ConceptBase: A deductive object base for meta data management, Journal of Intell. Inf. Syst. 4, 2, March 1995.
- [15] Klarlund, N., Moller, A., and Schwartzbach, M. I., DSD: a schema language for XML, in Proc. of FMSP '00 on Formal methods in software practice. Portland, USA, 2000.
- [16] Kobsa, A., Mueller, D., and Nill, A., KN-AHS: An adaptive hypertext client of the user modeling system BGPMS, in Proc. of the 4th Int. Conf. on User Modeling, Hyannis, MA, USA, 1994.
- [17] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. C., Telos: Representing Knowledge about Information Systems, ACM Trans. Inf. Syst. 8, 4, Oct. 1990.
- [18] Nejd, W., and Wolpers, M., KBS Hyperbook - A Data-Driven Information System on the Web, in Proc. of WWW8, Toronto, Canada, May 1999.
- [19] Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N., XML Schema Part 1: Structures, <http://www.w3.org/TR/xmlschema-1/>, Oct. 2000.