Intelligent authoring shell based on Web services

Marko Rosić
Faculty of Natural Sciences, Mathematics and Education
University of Split
N. Tesle 12, HR-21000 Split
Croatia
marko.rosic@pmfst.hr

Vlado Glavinić
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR-10000 Zagreb
Croatia
vlado.glavinic@fer.hr

Branko Žitko
Faculty of Natural Sciences, Mathematics and Education
University of Split
N. Tesle 12, HR-21000 Split
Croatia
branko.zitko@pmfst.hr

Abstract – **E-learning took place between information and communication technology on one side and education on the other. Intelligent authoring shells are kind of e-learning systems capable for generating intelligent tutoring systems. Web services, as standard for describing, publishing, discovering and binding application interfaces, could raise e-learning systems into a higher level enabling communication between learning systems regardless of used application platforms. xTEx-Sys is intelligent authoring shell completely oriented on Web services. This paper describes functional requirements and architecture of xTEx-Sys system.**

I. INTRODUCTION

Information and communication technology (ICT) became assembled part of educational system for assisting teacher in traditional lecture realization or substituting same lecture combining numerous methods and manners of realizing learning and teaching process. In respect, progress of ICT, especially in multimedia, computer networking and software engineering enables appearance of new system generations for computer-based learning and teaching.

E-learning took place between ICT on one side and education on the other. American Society for Training and Development defines e-learning as instructional content or learning experience delivered or enabled by electronic technology [1]. Numerous learning strategies and technologies for learning support such as CD-ROM, computer based lecture, videoconferencing, learning content distributed via satellite and virtual knowledge networks are officially included in e-learning. In other words, e-learning is not only Web-based lecture or distinct learning, it incorporates many situations of individual information interchanges and knowledge concurrence.

Development of e-learning systems came to the level where interoperability has to be assured. Interoperability makes development and maintenance of such systems less expensive and also ensures services acquisition. Moreover, there are two conditions imposed to e-learning systems [2]:

- just-in-time and on demand deliverance of information,
- reuse of learning material.

An important subclass of e-learning systems is represented by intelligent tutoring systems (ITS) that base their work on the simulation of the "real" teacher in the learning and teaching process and an intelligent authoring shells (IAS) that help creating ITSs.

Web services as standard for describing, publishing, discovering and binding application interfaces could raise e-learning systems into a higher level enabling communication between learning systems regardless of used application platforms. Generally, Web services facilitate application to application communication making heterogeneous Web-enabled learning systems cooperative.

Architectural requirements and content aggregation model customization that have influence in construction of our new IAS are discussed in the following. New system merges technical and functional benefits of previous versions of our intelligent hypermedia authoring shells. Its service-oriented architecture provides functionalities to other platform independent system. Such interoperability can take content developed in one location with one set of tools and use them in another location with a different set of tools.

II. BACKGROUND

Intelligent authoring shells are kind of e-learning systems capable for generating ITS. They can adjust learning and teaching contents and the way of domain knowledge perception, considering student's learning capabilities. Generated ITS architecture is modular, including modules for teacher, learner, expert and communication module that enables interaction among previously mentioned modules. According to this architecture we have generated intelligent hypermedia authoring shell called Tutor-Expert system (TEx-Sys) [3]. TEx-Sys structure is based on cybernetic model of learning and teaching process.

Whole model of TEx-Sys has been developed through three technological generations: (i) on-site realization (1992-2001, on-site TEx-Sys), (ii) Web-oriented realization based on dynamic Web document generation technology (1999-2003, Distributed Tutor-Expert System (DTEx-Sys) [4]) and (iii) performance-based on Web services (2003- , eXtended Tutor-Expert System (xTEx-Sys)). In this part of the paper are presented essential characteristics of first two system generations that came out of TEx-Sys model.

The first version of TEx-Sys was actively used in the learning and teaching process since 1997. TEx-Sys implements data storage using database with no support for network-based application usage. According this shell's weakness, complete system is designed as standalone application, without connections and data interchange with the environment. Modules are single executable files without intercommunication and relay on Windows operational system application programming interface (API).

DTEx.-Sys, as second IAS version, took advantage of Web technology and was implemented according to 3-tier application architecture (see Fig. 1).

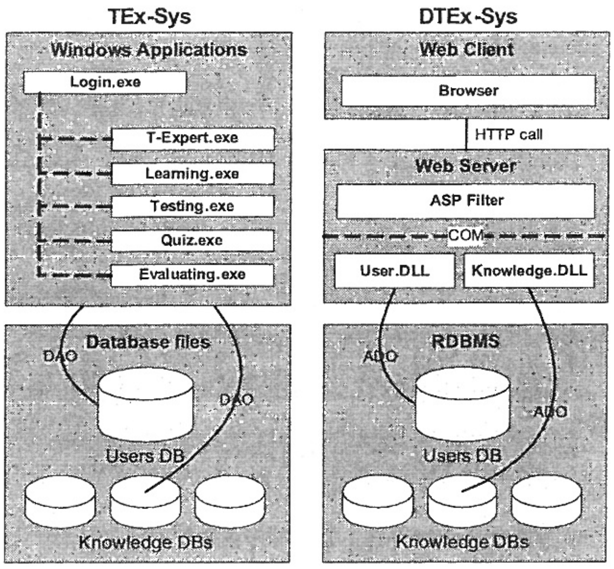

Fig. 1. Architectural differences of TEx-Sys and DTEx-Sys

Knowledge bases created using T-Expert module of TEx-Sys are converted into DTEx-Sys knowledge databases supervised by Relational Database Management System (RDBMS). Connectivity between RDBMS and application logic is realized on ActiveX Data Objects (ADO) language-neutral object model. ADO, as a programming interface for accessing data in a database, enables us to use predefined object's properties and method for easy data managing. DTEx-Sys application objects are enclosed into two dynamic link libraries (DLL) and deployed on Web server. User.dll file is used for administration of system users and is connected to DTEx-Sys user database. Second DLL file has methods for reading elements of stored knowledge bases. This application layer organization enables implementation of two user interface kinds. First one is Windows operational system dependent interface which has to be installed before employment. The second one is DTEx-Sys user interface, programmed using Active Server Pages (ASP) technology for dynamic Web page generation on Web server, and hereby installation of thin client is very likely not necessary because only Web browser is needed.

DTEx-Sys brakes classroom barrier and can be accessed anytime and anywhere. It does not support creating knowledge bases, which is the main functionality of TEx-Sys authoring shell, due to complexity of creating user-friendly Web page interface for knowledge authorization. Knowledge bases were still built using T-Expert module of TEx-Sys and manually deployed in DTEx-Sys RDBMS.

Differences between our two first versions of ITS authoring shells are shown at Fig. 1. Considering stand-alone TEx-Sys system, pre-installation of few components foregoes its main installation. Accessing knowledge databases requires setting Data Access Objects (DAO) engine. After deployment of DAO, executable files demand installing runtime engine and all ActiveX components utilized in user interface. Setting new versions of ActiveX components continually rewrites old component versions on computer system. Often new component versions had alternative method definitions that would not work with original TEx-Sys executable file. Considering deployment of DTEx-Sys, it is necessary to install RDBMS and Web Server on one host or two hosts eventually. Client side requires only Web browser. For developing knowledge bases we still have to install TEx-Sys. Realizing open architecture not only at user interface level, but at application level, our new system will provide functionalities to other distinct e-learning systems and organizations. System will also have ability to deliver learning content to other systems built upon different set of tools and platform.

Within TEx-Sys and DTEx-Sys knowledge is represented by semantic networks with frames [5]. Primary elements of domain knowledge are nodes and links. Nodes with frames and structural attributes are used to represent domain knowledge objects and they are semantically connected using links.

However, importance of new e-learning standards, such as Sharable Content Object Referent Model (SCORM) [6], necessitates us to customize and upgrade courseware of a new system, in order to include knowledge representation using semantic networks with frames.

In such system, not only learning content for distinct systems and organizations will be open by Web services, but also a provision of functionality such as managing users, creating knowledge's and authoring courses. These requirements define inception phase for the generation of IAS called xTEx-Sys. Our system is taking advantages of new technology which promotes construction of Web oriented system, is an IAS designed as a Web service. Students and teachers work in virtual classroom; students learn individually and teachers prepare courses for students. xTEx-Sys architecture incorporates advanced technologies to gain interoperability and reusability towards other educational systems.

In the following is elaborated how e-learning standards in organizing learning and teaching content had influenced the generation of our IAS based on Web services.

III. CONTENT AGGREGATION MODEL

Interoperability and reusability requests imposed to e-learning systems can be realized by standardizing architecture and content material included in e-learning systems. Herein, SCORM went inmost. SCORM helps to define the model that references a set of technical frameworks, specifications and guidelines designed mostly for Web-based learning content and systems. SCORM Content Aggregation Model (CAM) [7] describes composition of content based on reusable learning object called Sharable Content Object (SCO). Indivisible parts of SCO are assets which are electronically representation of media, such as text, images, sound and other pieces of data that can be displayed on Web client. Other SCORM organization units use SCOs to aggregate the content for learner (see Fig. 2) [8]. At the top level there is single root aggregation that can include other aggregations and SCOs. Other aggregation units are used to group other aggregations and/or SCOs, realizing content's tree structure.

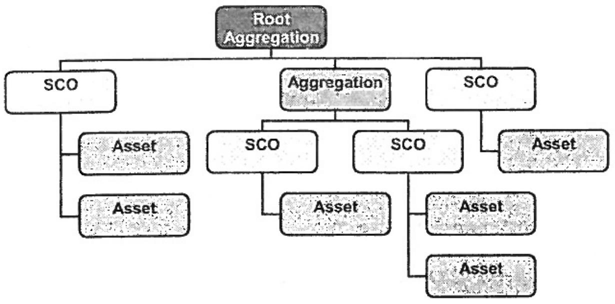

Fig. 2. Example of SCORM content aggregation model

Courseware in our earlier version of system is not totally guided. Students still have option to choose any nominated domain knowledge for learning and teaching, as well as for testing. Freedom in navigation throughout learning and teaching is moderately controlled using Courseware Agent or Learning Agent (CArLA) [9]. According to SCORM specifications for content aggregation model [7], along with sequencing and navigation [10], content is built in advance, so student is from top to bottom guided in learning and teaching and for testing. In the future SCORM will monitor semantic Web development for proper enhancement of knowledge representation [11]. We took precedence of knowledge representation using semantic networks with frames and some nodes from the network we define as an asset. Documents and media files attached to the node are used to refine description of knowledge object. In such way we still have export capacity to SCORM content model. Considering this new way of asset representation in SCORM model, we manage not with documents and media files, but with elements of semantic network with frames. That is the major difference of our IAS content aggregation model (see Fig. 3).

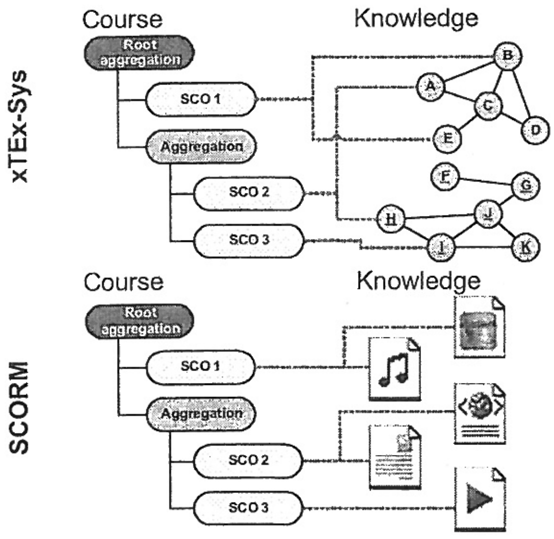

Fig. 3. Differences between assets in xTEx-Sys and SCORM course aggregation model

An overall process of xTEx-Sys generation is briefly described in the following. Analysis of xTEx-Sys construction is accompanied by description of tools and programming technologies involved.

IV. XTEX-SYS SYSTEM ARCHITECTURE

xTEx-Sys is completely oriented on Web services where appropriate one is assigned to every actor of the system. There are four main actors of the system:

- learner – learns beforehand arranged course,
- teacher – arranges learning and teaching course based on domain knowledge,
- expert – collects and formalizes domain knowledge,
- administrator – system resource manager, monitors system usage and stability.

According to Rational Unified Process (RUP) methodology, system has four general sets of use cases, and additional use case for shared functionalities. Those use cases designates xTEx-Sys Web services (see Fig. 4).

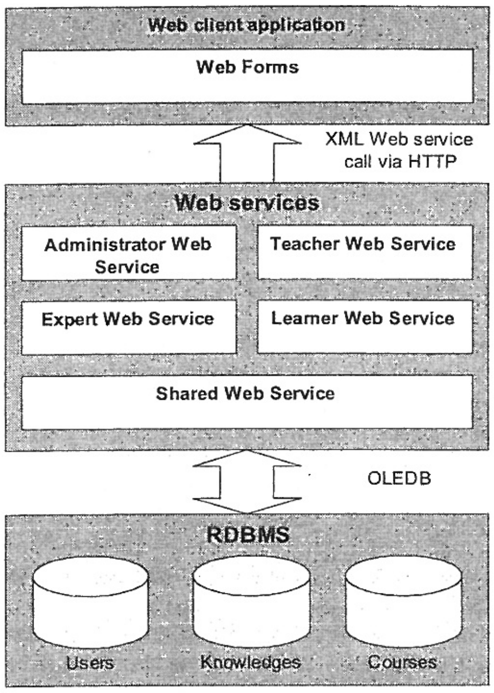

Fig. 4. xTEx-Sys overall architecture

System architecture is 3-tiered where service oriented technology connects parts of the system. Distributing nodes of xTEx-Sys can be dispersed on maximum 8 distinct hosts. Every Web service and database can be deployed separately. Host consisting database must have installed RDBMS insomuch that host providing Web service can access data and by Web server consign them to Web client. There are three major layers of the system architecture:

- data layer – consisting three databases semantically connected using XML schema,
- Web service – realizing application layer relaying on actors functionality, and
- user interface – including capability for constructing courseware and domain knowledge.

In following we describe layers of system architecture.

A. Data Layer

Previously known knowledge and users' databases together with new courses database are encapsulated within Microsoft SQL Server 2000 RDBMS. Users' profile information is grouped and written into users' database and serve for collecting data of system functions usage. Additionally, learner progress in learning and teaching is tracked and stored. All system users share xTEx-Sys communication module and records of such communication are stored in users' database. Unlike having multiple knowledge database files in previous versions of the systems, knowledge is now indexed and aggregated into single database. In DTEx-Sys and TEx-Sys knowledge is recognized by its database name. Now knowledge names are stored into table called sub-area, and those knowledge names are grouped using area table. Thus, knowledge is more searchable and navigable, and enables easier creation of courses stored in course database. All databases are enriched with stored procedures written for standard records manipulation, which means that commands for inserting, updating and deleting records are executed at RDBMS side. These databases are not semantically independent; records from database table are related to some table records in distinct database. Overlapping relations between tables from distinct databases is described by XML Schema definition language (XSD) [12,13]. XSD is used to define valid structure, valid data content, and relations of certain types of XML data, just as a database schema defines and validates the tables, columns, and data types that make up a database. Table with users' data is part of a users' database, while other tables are part of a knowledge database, as shown in Fig. 5.

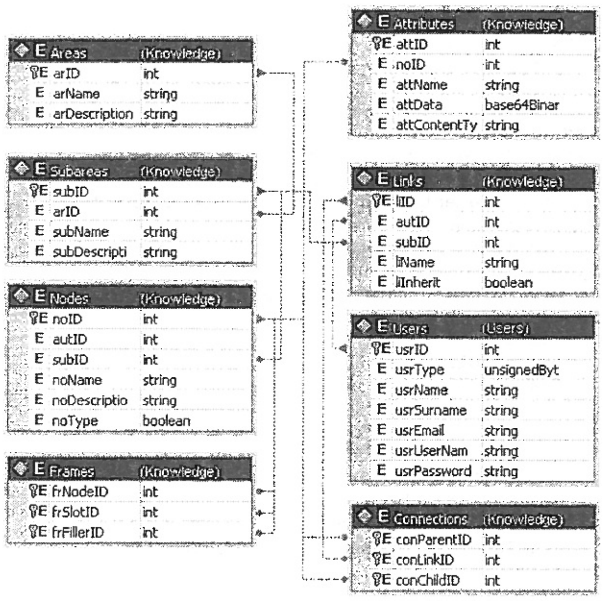

Fig. 5. Schema view of XML Schema for expert Web service data

XSD elements' definition enables assembling needful data content, independently of data storage place. XML Schema usage guarantees consistency among certain types of XML data that is shared among applications and organizations. Its ability to pass through firewall barrier guarding Internet hosts makes it ideal for Web services data transfer.

B. Web services

Web services refer to the technologies that allow connections among Web-based systems. Generally, service is the endpoint of a connection and has some type of underlying computer system that supports the offered connection [14]. The combination of an organization's internal and external services makes up service-oriented architecture. Web Service Description Language (WSDL) [15] is a format for describing a Web Services interface. It is a way to describe services and how they should be bound to specific network address. WSDL is composed of three parts (see Fig. 6):

- definitions
- operations
- service bindings

Definitions are globally expressed in XML and include both data type definitions and message definitions that use the data type definitions. These definitions are usually based upon some agreed XML vocabulary. The most likely industry-wide vocabulary will be used for data type and message definitions used between organizations. WSDL operations describe actions for messages supported by a Web service. They are grouped into port types. Port types define a set of operations supported by Web service. Service bindings connect port types to a port. A port is associating a network address with a port type. A collection of ports defines a service. This binding is commonly created using Simple Object Access Protocol (SOAP), also other forms can be used [16].

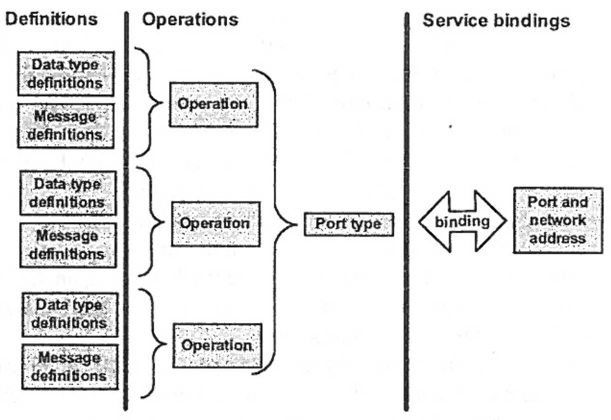

Fig. 6. Relationships among WSDL's basic parts

SOAP provides the "envelope" for sending Web services messages over the Internet and contains two parts:

1. An optional header providing information of authentication, data encoding, or how a recipient of a SOAP message should process the message.
2. The body that contains the message which can be defined using WSDL specification.

SOAP commonly uses HTTP, but other protocols such as SMTP may be used. A Web services emphasize .NET strategy. The .NET Framework is an integral Windows component for building and running the next generation of software applications and Web services. It is composed of common language runtime and a unified set of class libraries.

.NET framework is used for building and running all kinds of software, including Web-based applications, smart client applications, and XML Web services-components that facilitate integration by sharing data and functionality over a network through standard, platform-independent protocols such as XML (Extensible Markup Language), SOAP, and HTTP (see Fig. 7) [17].

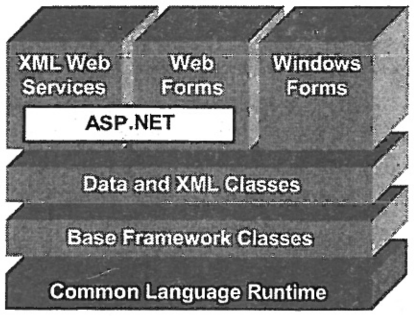

Fig. 7. Architecture of the .NET framework

ASP.NET is one of the centerpieces of the Microsoft .NET Framework and provides the infrastructure for developing dynamic .NET Web applications. ASP.NET is not only the successor to Microsoft Active Server Pages (ASP), it is a unified Web development platform that provides the services necessary for building Web applications. The ASP.NET page framework is a scalable programming model that is used on the server to dynamically generate Web pages. Building an XML Web service using ASP.NET automatically supports client's communication using the SOAP, HTTP-GET, and HTTP-POST protocols. Since HTTP-GET and HTTP-POST support passing messages in URL-encoded name-value pairs, the data type support for these two protocols is not as rich as that supported for SOAP. In SOAP, which passes data to and from the XML Web service using XML, it is able to define complex data types using XSD schemas, which support a richer set of data types. XML Web service generation with ASP.NET does not have to explicitly define complex data types they expect when using an XSD schema. Rather, it is built using a managed class. ASP.NET handles mapping class definitions to an XSD schema and mapping object instances to XML data in order to pass it back and forth across a network. XML Web service methods are a key part of the messaging infrastructure employed by XML Web services. Programming Web services using ASP.NET consists of defining Web methods. Methods of a class implementing an XML Web service do not automatically have the ability to be communicated over the Web, but with XML Web services created using ASP.NET, adding that capability is enabled.

As seen in Fig. 4, Web services are concentrated around actor's functions. Expert Web service consist Web methods for managing domain knowledge. It can manage areas and sub-areas by calling Web methods, as pictured on Fig. 8.

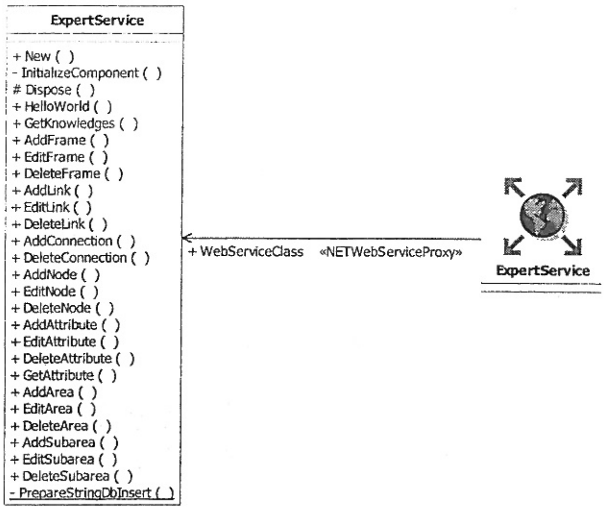

Fig. 8. Expert Service class

These Web methods enable managing domain knowledge. For example, calling *AddNode* method and passing node name as input parameter over the network will result with the addition of one node to the domain knowledge and returning value of successfulness of same action. *GetKnowledge* method has no input parameter, it passes all elements of domain knowledge using XML format for data interchange (see Fig. 9).

XML format consists of two parts; on the one hand the definition of meta-data which includes tables, columns, indexes and keys definitions with table relations and on the other appropriate formatted data. Such data format is interpretive to any Web-enabled system permitting independence of user interface in relation to application logic.

```
  <xs:element name="Nodes">
    <xs:complexType>
    <xs:sequence>
    <xs:element name="noID"
msdata:ReadOnly="true"
msdata:AutoIncrement="true" type="xs:int" />
  <xs:element name="autID" type="xs:int" />
    <xs:element name="subID" type="xs:int" />
    <xs:element name="noName" type="xs:string" />
    <xs:element name="noDescription"
type="xs:string" minOccurs="0" />
    <xs:element name="noType" type="xs:boolean"
/>
  </xs:sequence>
  </xs:complexType>
  </xs:element>

  <Nodes diffgr:id="Nodes3" msdata:rowOrder="0">
    <noID>29</noID>
    <autID>10</autID>
    <subID>1</subID>
    <noName>Animal</noName>
    <noDescription>class</noDescription>
  <noType>false</noType>
  </Nodes>
```

Fig. 9. XML schema and data view of node element

C. User interface

Web page elements are not grateful user interface elements. It is far easier to develop interface based on operational system graphic user interface (GUI) than designing usable Web page. The .NET technology took best practices in developing Web interfaces and facilitates creating custom Web components powered with events definition. Internal set of .NET Web components still can be used in Web page. In .NET, a Web page is represented by Web Form class, with all its functionality. Such Web Form class has other classes called Web components. Functionality of Web page is event driven across Web components, therefore designing user interface for Web application greatly approximates the development of standard GUI. For example, Web page from Fig. 10 illustrates components for editing connections between nodes.

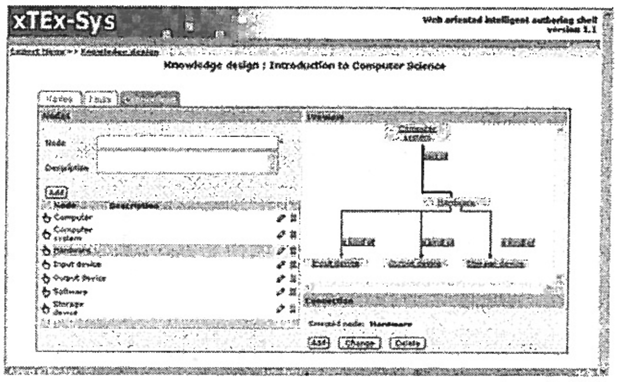

Fig. 10. Example of xTEx-Sys user interface

V. CONCLUSION

This paper describes functional requirements and architecture of xTEx-Sys system. Unlike systems based on SCORM model, xTEx-Sys includes courseware based on knowledge representation, not only on a set of electronic documents and media files. xTEx-Sys architecture is based on Web services, enabling publication of learning content as well as providing the ability of sharing between similar e-learning systems. Every component that satisfies specification of built services is able to access learning content via matching Web services. Such system realization increases interoperability with other e-learning systems, facilitates knowledge access and enables e-learning interaction based on quality user interface.

VI. ACKNOWLEDGMENTS

This work has been carried out within project 0036033 Semantic Web as Information Infrastructure Enabler, as well as technological project TP-02/0177-01 Web-based Intelligent Authoring Shell funded by the Ministry of Science and Technology of the Republic of Croatia.

VII. REFERENCES

[1] xxx, "A Vision of E-Learning for America's Workforce, Report of the Commission on Technology and Adult Learning", American Society for Training and Development, 2001, http://www.astd.org/

[2] xxx, "E-learning, A White Paper from IsoDynamic", IsoDynamic, 2001, http://www.isodynamic.com/

[3] S. Stankov: *Isomorphic Model of the System as the Basis of Teaching Control Principles in an Intelligent Tutoring System*, PhD Diss., Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Split, Croatia: 1997 (in Croatian).

[4] M. Rosić, *Establishing of Distance Education Systems within the Information Infrastructure*, M.Sc. Thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia: 2000 (in Croatian).

[5] S. Stankov, V. Glavinić and M.Rosić: "On Knowledge Representation in an Intelligent Tutoring System" in *Proceedings of 4th IEEE International Conference on Intelligent Engineering Systems – INES'2000*, Portoroz, Slovenia: 2000, pp 17-19.

[6] xxx, *SCORM 2004 Overview*, Advanced Distributed Learning, 2004, http://www.adlnet.org/

[7] xxx, *SCORM Content Aggregation Model, version 1.3*, Advanced Distributed Learning, 2004, http://www.adlnet.org/

[8] xxx, *SCORM Best Practices Guide for Content Developers*, 1st Edition, Carnegie Mellon Learning System Architecture Lab, 2003.

[9] S. Parović, S. Stankov and B. Žitko: "CArLA – Intelligent Agent as Support for Learning and Teaching Process" *CEEPUS SUMMER SCHOOL Split 2002 Jointly with Fifth Symposium on INTELLIGENT SYSTEMS*, Split, Croatia: 2002

[10] xxx, *SCORM Sequencing and Navigation, version 1.3*, Advanced Distributed Learning, 2004, http://www.adlnet.org/

[11] P. Dodds and J. D. Fletcher, Opportunities for New "Smart" Learning Environments Enabled by Next Generation Web Capabilities" in *Proceedings of Learning Objects 2003 Symposium – ED-MEDIA 2003*, 2003, pp 20-25.

[12] xxx, "Extensible Markup Language (XML)", 2003, http://www.w3.org/XML/

[13] xxx, "XML Schema Part 0: Primer", 2001, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[14] xxx, "Web Services Architecture: W3C Working Draft 8", 2003, http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[15] xxx, "Web Services Description Language (WSDL) 1.1", 2001, http://www.w3.org/TR/wsdl/

[16] xxx, "Simple Object Access Protocol (SOAP) 1.1", 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[17] T. L. Thai and H. Lam, *.NET Framework Essentials*, O'Reilly & Associates; 1st Edition, 2001.