ARCADE – Web-Based Authoring and Delivery Platform for Distance Education

Boyan Bontchev, Trayan Iliev

Department of Information Technologies, Faculty of Mathematics and Informatics
Sofia University, 1164 Sofia, Bulgaria
Email: {bbontchev, t_iliev}@fmi.uni-sofia.bg

Abstract. ARCADE is a distance learning platform based on international standards and specifications in the areas of e-learning, object-oriented development, reusable knowledge and information exchange. The Unified Software Development Process and UML were consistently applied in the project. ARCADE was developed in two stages – the courseware delivery features were implemented first, and then, in a second version, they were integrated with a courseware-authoring environment. The authoring environment provides instructors with abilities for flexible multimedia content creation, maintenance, reuse, and view personalization.

1 Introduction

During the last decades we became witnesses of important changes in worldwide business environment – the economies globalization and their transformation from industrial to knowledge and information based [15]. The global economy gave rise to global competition between business enterprises, especially in the well-developed, traditional industries. This became true even for academic and other educational organizations. In order to satisfy the demand of highly skilled knowledge workers and to be compatible in the global market, the educational institutions should provide new *flexible learning* opportunities.

According to Van den Brande [21]: "Flexible learning is enabling learners to learn when they want (frequency, time, duration), how they want (modes of learning, and what they want (that is learners can define what constitutes learning for them)". Terms like *tele-learning*, *e-learning*, *e-training*, *online and computer-based learning* describe the powerful learning opportunities emerging from combination of new pedagogical paradigms with state of the art information and communication technologies [7, 8]. Different types of *Course Support Systems* (CSS) [19] were developed to provide means for practical realization of these flexible learning opportunities – *Learning Management Systems* (LMS), courseware authoring tools, systems for computer based testing, repositories of reusable learning content, etc. The effective realization of the distance education process requires all these tools to be able to work together, exchanging data with each other, and supporting similar conventions and user interface styles.

In order to achieve learning content reusability and interoperability between different CSS several standardization initiatives and organizations were established, including Dublin Core Metadata Initiative (DCMI) [10], IMS Global Learning Consortium [13], ARIADNE Foundation for the European Knowledge Pool [4], Advanced Distributed Learning Initiative (ADL) [1], and IEEE Learning Technology Standards Committee (LTSC) [12]. The increasing number of organizations concerned with e-learning standardization efforts is a good evidence about the importance of establishing common standards in areas such as learning content metadata annotation and packaging, student assessment and modeling, content sequencing and presentation, etc. This will create a new market for reusable courseware materials – components (*learning objects)* that organizations providing education can purchase and integrate according to their unique educational needs.

Among the number of e-learning standards and specifications available, two were chosen for pilot implementation in ARCADE distance education platform – the IEEE LTSC *Learning Object Metadata* (LOM) standard, and the IMS *Question and Test Interoperability* (QTI) specification. They cover two central processes for distance learning – presenting learners with reusable courseware content, and receiving feedback about students' progress.

2 The ARCADE Project

ARCADE (Architecture for Reusable Courseware Authoring and Delivery) is a project developed by a team of researchers and students at the Department of Information Technologies, Faculty of Mathematics and Informatics of Sofia University "St. Kliment Ohridski". The goal of the project is to develop an integrated software platform for authoring and delivery of Internet-based distance courses covering the university needs. [3]

Several guiding design principles were applied in order to achieve this goal:

- Using object-oriented paradigm for software development on multiple levels – applying state of the art standards (Unified Modeling Language – UML) and development methodologies (Unified Software Development Process – USDP), as well as designing a distance education environment based on object manipulation and management.
- Extensive use of existing *e-learning standards and specifications* (IMS, IEEE LTSC, ADL SCORM) in order to provide interoperability and possibility for information exchange with other CSS based on the same standards.
- Building an *open* system by using XML import/export to reliably exchange data with different external software applications and sources of information.
- Designing a *modular* architecture: different project modules are made as independent as possible by using well-defined interfaces that allow binding components in a configuration most closely satisfying the educational needs of the client organization.
- Designing an *open system architecture* that is *easy to extend* with new functionality – by consistently applying UML and USDP the project is based on solid grounds for continuous future extension.

- Platform independent application development based on reliable implementation technologies – by choosing the industry standard Java™ and Java Server Pages (JSP™) as main implementation technologies for the project.

The project has been developed in two stages. The first stage (ARCADE 1.0) was focused on providing web-based courses with a common delivery framework incorporating user and system management, course and curriculum management, communication facilities, and student assessment tools. These modules are automating the work of four different types of users – *students*, *instructors*, *course administrators* (managing the curriculum and courses), and *system administrators* (managing the system in a secure and stable state). The delivery framework is based on the metaphor for Web-based virtual classroom described in [17]: "A Web-based classroom is an environment created on the World Wide Web in which students and educators can perform learning related tasks. A Web-based classroom is not simply a mechanism for distributing information to students; it also performs tasks related to communication, student assessment, and class management".

The main goal of the second stage (ARCADE 2.0) was to design and implement an open, platform independent set of tools, automating the process of courseware authoring, and providing opportunities for XML import/export of reusable *learning objects*, that can be dynamically combined to form a curriculum, tailored particularly to unique educational needs of the learners with different profiles.

The before mentioned design principles were consistently applied during both stages of the project.

3 Development Methodology

As described by Jacobson, Booch and Rumbaugh in [14] the Unified Software Development Process (USDP) is a "generic process framework that can be specified for a very large class of software systems, for different application areas, different types of organizations, different competence levels, and different project sizes". The Unified Process is *component-based*, and uses *Unified Modeling Language* (UML) for preparing all blueprints of the software system. UML is an industry standardized "graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [18]. The Object Management Group (OMG) – an organization that "promotes the theory and practice of object-oriented technology in software development", adopted it as a standard in 1997.

The four key ideas behind the Unified Software Development Process are [14]:

- Use-case driven – based on *use-cases* describing "the manner in which the system provides a discrete value to an actor" [5], grouped together in a *use-case model*.
- Architecture-centric – based on functionality described in the use-case model it "casts the system in a form" that "embodies the most significant static and dynamic aspects of the system" [14].
- Iterative – the system is envisioned as an evolving entity that passes through several cycles of refinement.

- Incremental – the functionality is added in small portions, thus providing a smooth transition towards the complete realization of the system.

According to [14] the software lifecycle goes through four main phases: *inception*, *elaboration*, *construction*, and *transition*. During the inception phase a vision for the software product is established. It is followed by a complete specification of functional and nonfunctional requirements (use-case model) and detailed system architecture in elaboration phase. The construction phase "adds flesh" to the architecture skeleton specified – the software product is built using chosen implementation technologies. During the transition phase the emphasis is placed on maintenance and support activities like correcting defects and training users of the system.

There are five main *workflows* that take place during each project iteration – *requirements specification*, *analysis*, *design*, *implementation*, and *testing*. The main purpose of the *requirements specification* workflow is to develop a use-case model of the software system that describes the *actors* – "coherent sets of roles that users of an entity can play", and the *use-cases* – "coherent units of functionality provided by a system" [18]. During the *analysis* workflow the functional and nonfunctional requirements are transformed in overall system architecture. There is a bidirectional relationship between the use-case and the analysis models – the specification of requirements leads to architecture forming, and presenting the architecture leads to better understanding and re-formulation of requirements. The *design* workflow links the abstract analysis architecture with concrete implementation technologies – it provides a design model, a detailed blueprint for the system implementation. The real construction and coding of different modules is the main purpose of the *implementation* workflow. During the *testing* workflow all implemented classes, subsystems, and the whole software system are tested according to a *test plan* (unit testing, integration testing, alpha and beta testing). *Test cases* are derived from existing use-cases. They are applied according to predefined *test procedures*, and the whole process can be automated using *test components*.

For each workflow there are defined sets of *workers*, *activities*, and *artifacts*. The *workers* are developer roles of specialists taking part in the realization of certain *artifacts* (documents, modules, etc.) in the scope of different *activities* defined for the workflow.

The USDP seems the most appropriate candidate for ARCADE system development (compared with other candidates like Extreme Programming - XP, and Dynamic Systems Development Method – DSDM) [3, 24] because it supports production of high quality, UML-based documentation inevitable for an open project with many participating developers. It is providing an opportunity for parallel development of different modules by concurrent teams of researchers and students, because of the well-specified workflow models with their activities, workers and artifacts. It takes into account the context of the project development – the educational value and additional motivation for the students that comes from teamwork and practical application of such "state of the art" object-oriented development methodology. Though it is not ideal – implementing it in a full scope requires a lot of resources. That's why – as proposed by the authors of the methodology – the process should be tailored to the individual project requirements.

4 ARCADE 1.0 Platform

4.1 Types of Users

According to the use-case model of ARCADE there are five main types of users (actors) specified – *Student*, *Instructor*, *System Administrator*, *Course Administrator*, and *Course Author* (added in ARCADE 2.0). Fig. 1 shows an example diagram presenting actors and part of the main use-cases in ARCADE 1.0.

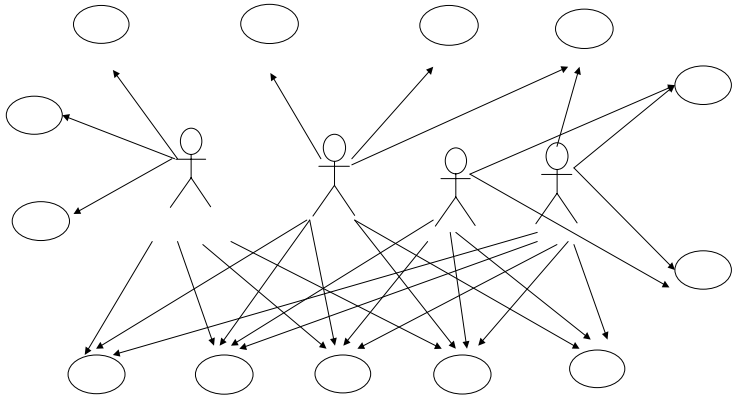

Fig. 1. Main functions provided by ARCADE 1.0 for different types of users

Each actor corresponds to a *default user group* in the implemented system with given access to specific functions, and with different responsibilities. A user can participate in more than one user group, accessing all the functions defined for these groups. An example showing *Course Administrator* functions in the left panel is presented in fig. 2. A brief description of the responsibilities assigned to these default user groups is given below:

- *Student* – uses learning resources to gain knowledge and skills, may participate in several registered courses, can access the course materials and the course environment, and may evaluate his/her performance. The students are allowed to form individual programs choosing from available elective courses (restrictions may be added on the minimal and maximal number of courses chosen) according to their personal preferences and interests. They are organized in student groups, for the purpose of curriculum management.
- *Instructor* – supports *Students* during the learning process. Tasks performed by the instructor include creation of a *course instance* (course customization according to specific educational needs of students with a given profile) from a given course, and defining and tuning course instance environment. The instructors may set up the evaluation procedures - create and add tests, define assignments and other assessment objects, that are not included in the initial course description, review the assignments results, and form the final grades for the students.

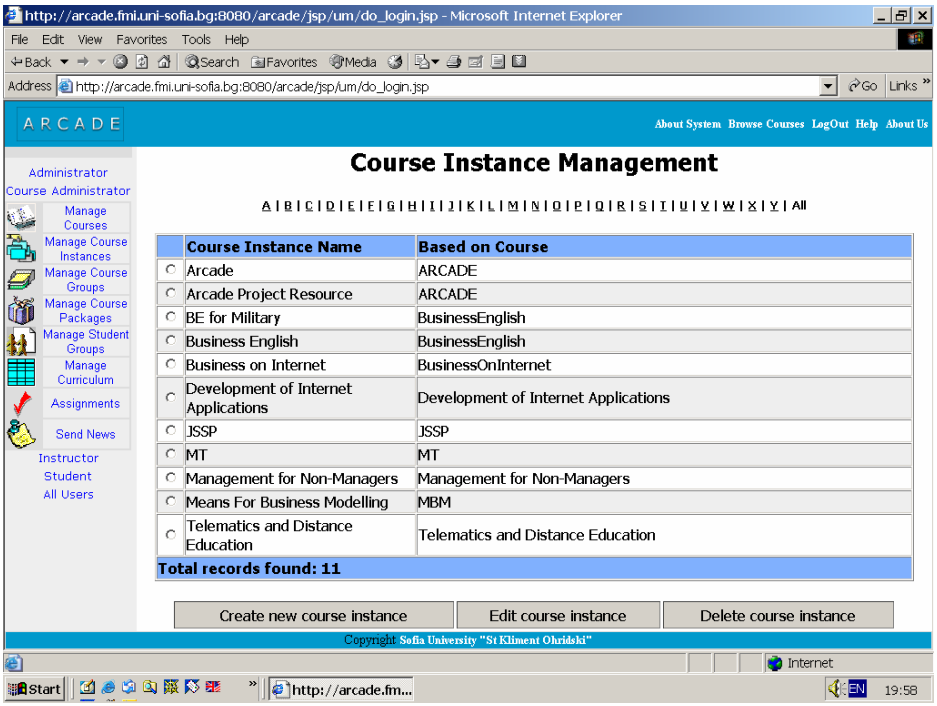

Fig. 2. Course Instance Management screen in ARCADE 1.0

- *Course Administrator* – manages courses, course instances and the curriculum, enrolls the students and the groups of students in a course instance, and assigns instructors to student groups. Different course instances may share one course as a source for learning resources. The course administrator defines different properties of a course instance - if it is elective or mandatory, freely available or paid. The *curriculum* is a cross table between course instances, student groups and instructors.
- *System Administrator* – takes full control over the system, manages system resources and settings, creates user accounts and user groups, and assigns user permissions. In addition to default user groups mentioned above, she/he can create additional user groups with different permissions in order to make the user management more flexible. The users may participate in more then one group. The system administrators also monitor event logs and maintain the system in secure and stable state.
- *Course Author* – creates and modifies the courses according to pre-defined models, supplies and manages learning object meta-data records according to LOM standard, describes the course structure and provides course materials by uploading resources. This user role was added in the second version of ARCADE and its functionality is described in more detail in section 5.

There are also some ARCADE functions available for all registered users of the system – abilities to modify the personal data, to use communication facilities, and to view the curriculum.

4.2 Architecture

As presented in fig. 3 there are four main packages developed during the first stage of the project:

- *Course and Curriculum Management* – implementing different functions for the instructors and the course administrators, such as managing courses and course instances, assigning students to student groups and creating curriculum. Some student functionality is also included in this package – searching for courses, forming individual program by choosing elective courses in addition to basic ones, and reading course materials.

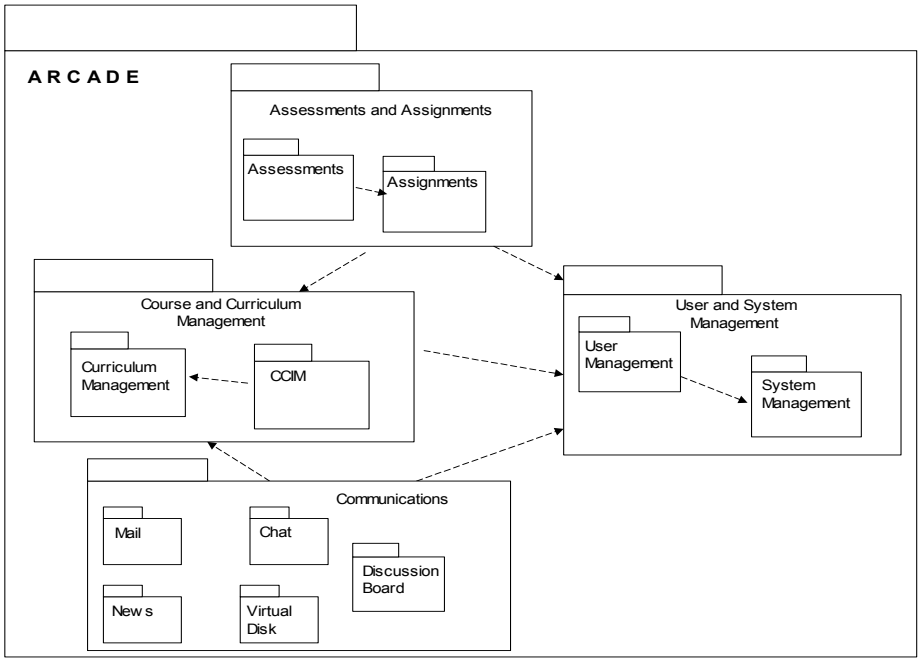

Fig. 3. Architecture of ARCADE 1.0 – main packages and their dependencies

- *User and System Management* – this package contains most of the functionality assigned to the system administrators including management of users and user groups, assigning user permissions for different objects in the system, managing system event logs and so on.
- *Communications* – five different communication alternatives are proposed for the first prototype: internal e-mail, virtual chat rooms, discussion boards, news, and shared/personal space (web-based virtual disks). All users of the system can use them in order to collaborate (while working on a group course projects, for example). For some of them (discussion boards, news) there are special privileges available for instructors and administrators but not for students (ability to publish news for example).

- *Assessments and Assignments* – the package is responsible for management of students' performance evaluation. It supports the construction, modification, and management of tests by instructors, performing the tests by students, evaluating results according to built-in instructions, generating feedback to instructors and students. It provides means for automated assignment and evaluation of individual and group course projects, and helps the instructor to form final grades based on different students' assessments [2, 23].

The last package is central for the system not only because it provides instructional feedback, thus enabling the instruction be adapted to individual students' needs as a basis for personalized and adaptive instruction, but because it was the pilot work-package where the new standard specification (IMS QTI) was applied in ARCADE.

The IMS Question and Test Interoperability Specification (QTI) "describes a basic structure for the representation of question (item) and test (assessment) data and their corresponding results reports" [13]. The *items* are combining interrogatory, rendering, and scoring information. The *sections* are collections of items and other sections grouped together in order to provide additional information (e.g. sequencing instructions). The *assessments* are the overall structure corresponding to a whole test. Grouped together these elements constitute the so-called *ASI structures* (Assessment, Section, Item) defined by the specification. The sections and items are stored in *object banks* in order to facilitate the reusability of the materials.

The IMS QTI specification was used as a basis for the development of the *Assessments* module of the ARCADE system. During the design the module was divided in three subsystems, implemented as three consequent *builds* (implementation iterations) [23]:

- *ASI structures management* – related to creation of ASI structures' elements, importing, searching and retrieving these elements to/from object banks.
- *Tests management* – provides ability to create *tests* – instances of *assignments* with set property values (date and group of students to be assigned)
- *Test performance* – allows students to perform given tests, the test results to be automatically evaluated according to a predefined set of rules (or to be evaluated manually by the instructor), and to receive feedback about their performance.

During the analysis, design and implementation workflows (according to USDP), there was developed and realized a layered system architecture allowing the ARCADE platform to be easily extended in the future with new modules. It is based on the industry standard three-tier server architecture separating different aspects of the software application development – presentation, business logic, and persistent data storage and retrieval [6]. The main implementation technologies applied were Java Server Pages (JSP™) for the presentation layer, Java™ for business logic implementation, and JDBC™ for accessing different databases. The chosen technologies provided the possibility for achieving platform independence (for the hardware platform, operating system, and relational database management system).

The second version of the system extended the ARCADE functionality in several directions. It added support for multiple languages, extended the number of RDBMS supported, and developed a multimedia authoring tool in order to close the courseware development lifecycle. This tool is described in detail in the following section.

5 The ARCADE Authoring Tool

5.1 Objectives

The ARCADE authoring tool aims to provide authors with the ability to separate content authoring and presentation at different levels. Using different styles of content templates by means of DTD or XSchema [16], authors may choose between strong separation *at infinitum* and introducing presentation properties inside XML content in order not to define XSL transformations for each content piece but to use only one XSL template for all the content. The tool contains and manipulates content templates with metadata definitions according to LOM specification making the produced courseware reusable for LOM-compliant e-learning environments. As far as it is designed for creation and delivery (both via Internet) of content formed as *linear story with hierarchical structure*, it can be used for maintenance of any Web sites presenting books, stories, magazines and similar linearly-hierarchical content. The content structure of each site should be defined by a separated template and may have powerful and extensible hierarchy without restrictions in terms of level's depth, hierarchy tree size, etc. Using multilevel hierarchies for content structure the authoring tool does not suffer granularity or paginating problems. It provides a *template-driven* content design process, where the chosen content template guides authors. One of the most important design issues in e-learning platforms is to provide effective support for the learning objects. By means of multilevel hierarchies for multimedia content organization, the ARCADE authoring tool deals flexibly with objects granularity at different aggregation levels in order to achieve maximal reusability of the content. The main types (levels) of 'building blocks' are:

- *Media element (learning tokens)* - text (plain text, tables and lists), images, animations, video or audio;
- *Page element (learning object)* - consists of one or more media elements;
- *Content module (lesson)* - consists of one or more learning elements grouped into a hierarchy structure;
- *Thematic site (study course)* - includes hierarchy of content modules and defines a thematic structure relevant for a specific study goal.

While the first names of these building blocks are generic and used when authoring general purpose content, the names in the brackets are their interpretation in the e-learning context. The e-learning support was improved by using the IEEE LTSC Learning Object Metadata (LOM) specification templates at page elements level. The authors may specify metadata about learning objects at different depth according to their needs and the purpose of objects [9, 11, 20].

As stated in the beginning in this section, the authoring tool provides means for clean separation of the course content and presentation. The following features were realized:

- flexible differentiation of views (presentations) of a given course content not only in the terms of different layouts but, as well, supporting different, personalized course instances (i.e., for different groups of course users);

- easy content management (with possibilities to add new items, copy and paste, move and delete, search and retrieve items) by storing the content in XML format;
- XML content can be exported in different presentation formats (such as HTML, WML, PDF, MS Word, etc.). By adding new exporter (render) modules the tool can easily support new output formats.

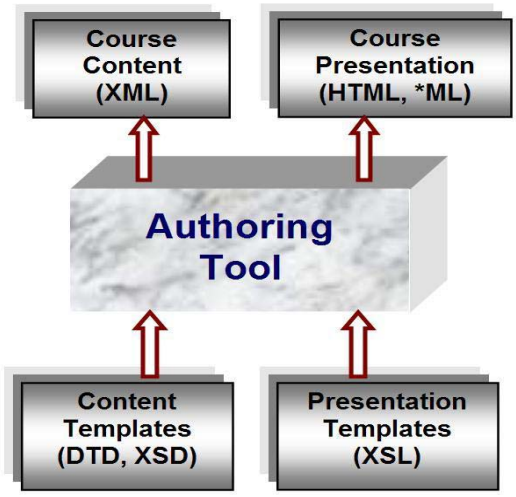

Fig. 4. Authoring tool functional workflow

The information workflow is presented in fig. 4. Content templates can be in DTD (Data Type Definition) or XSD (XSchema) format, while presentation templates are various XSL definitions transforming XML content in HTML, WML or any-ML (*ML). The content, content templates and presentation are accessible in different ways by the user roles described in section 4.1. In ARCADE 2.0 their functions are extended as follows:

- *Course Author* – can define both content and presentation templates and write the content.
- *Instructor* – may use the content templates to create personalized course instances targeted for given individuals or groups of students; can define additional content templates for extra personalization of content presentation.

Other design goals of the authoring tool with regard to common design principles (described in section 2), which have been addressed before the implementation phase of the project, are:

- *high-level personalization* - all the groups of building blocks can be personalized in terms of different content excerpts, and personalized look-and-feel for different audiences;
- *easy to learn user interface* – all the user roles can access hierarchically structured content by drill-up and drill-down navigation options, and as well, can move linearly through the structures using links to next/previous/first/last page;

- *flexible content reuse* – especially for the media elements (learning tokens), and the page elements (learning objects);
- *search and retrieval abilities* – content-based search and metadata search.

5.2 Authoring Tool Architecture

The authoring tool architecture is Web-based and follows the modular design presented in fig. 5. The content and presentation manager takes a central place in the system, as it is responsible for setting and updating relations between the content building blocks, for storing them in the database, and for submitting all needed information to the XML/XSL transformer [22].

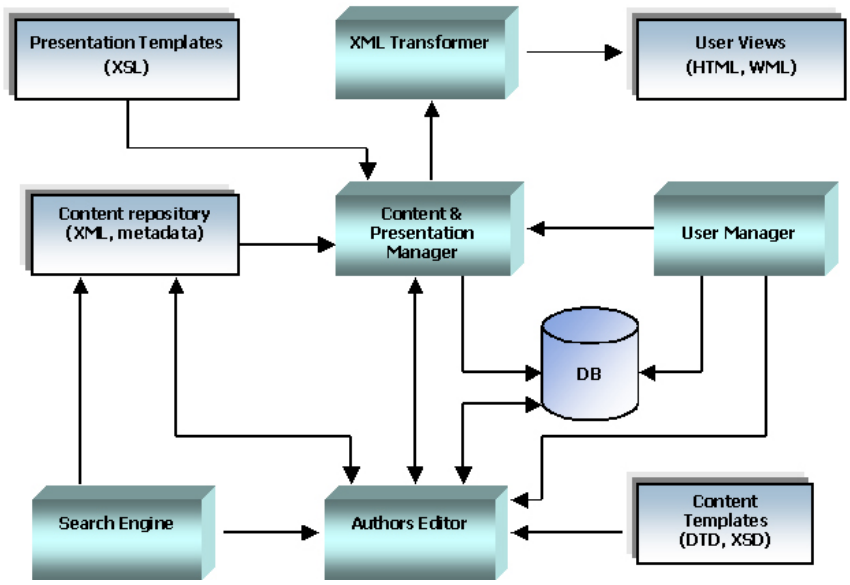

Fig. 5. Authoring tool basic architecture

The content building blocks relations are used to realize operations like deleting all media elements for given page in an effective way. The content manager contains a XML parser, which generates from any XML file a document tree according to DOM (Document Object Model) standard, and manipulates it. It is used by instructors to create course/site instances and to generate presentation by choosing a predefined XSL template set, and by students – to view some of the created instances based on their access rights.

The authoring tool editor is accessible through a user manager module only by authors for creating and updating courses. The authors can choose a preferred content template (DTD or XSD, i.e. XSchema) in order to create their course/site content. The content authoring process is *template-driven*, i.e. authors are guided in the creation of linear hierarchical structure, content pages and media elements, based on the chosen DTD/XSD. Thus, inside a hierarchical structure, authors may create/update/delete a

sub-hierarchy, and inside a page – a media element. There are two types of content templates – for content structure tree (i.e., courses and lessons) and for page elements (for example slides and LOM).

Page elements (or learning objects) are connected to the content tree by means of XLinks [16]. They can be located based on their content or metadata using a search engine (fig. 5).

In order to support various types of client applications, some new components should be added to the common middleware architecture. In fig. 6, HTTP requests are passed through an agent detector, which determines the browser type and reports it to a presentation manager. As in the common Java™ server architecture [6], the JSP™ (Java Server Pages) are invoked by a view handler to return XML data content. It is accepted by the presentation manager, which chooses the XSL needed to transform XML into presentation corresponding to the detected type of client.

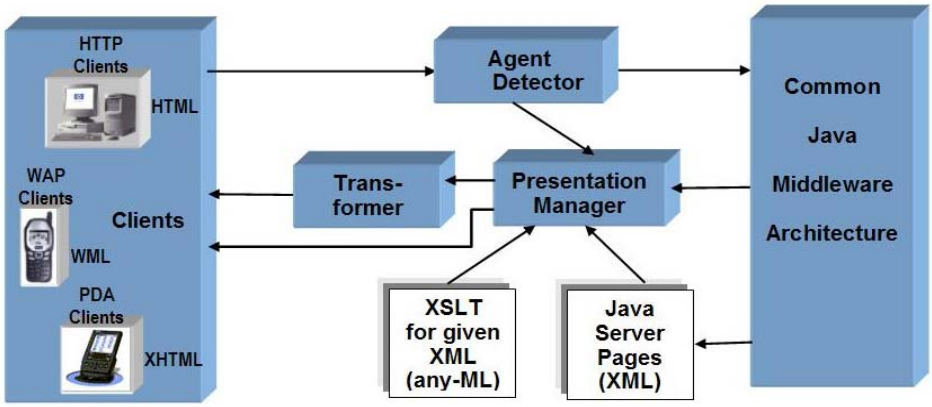

Fig. 6. XML/XSLT extensions of Java™ middleware

The XML/XSL transformation can be done on the client side for XML/XSL-compliant client devices. However, due to existing support problems with client-side XSL transformations, they are done on the server. When content pages (i.e., learning objects) are requested, they are generated dynamically (on-the-fly) as HTML/WML/any-ML views by the XML/XSL transformer.

The software implementation of the authoring tool uses Java™-based freeware technologies to create an open, portable, and easy for integration tool at reasonable cost. Initially was used an embedded database server (HyperSonic database) in order to facilitate remote installations, and after that the tool has been tested with MySQL™ and Oracle™ database servers. The business layer is implemented by JSP™ and Java Servlets™ following JSP™ model 2 [6, 20] and runs not only under any Java™ application server but as well under JSP™ engines such as Apache Tomcat and Resin™. The presentation layer uses HTML/CSS and JavaScript but can be implemented for any markup language by adding new XSL templates.

Figure 7 presents an editor screenshot of the integrated authoring tool. The left navigation frame contains unfolded tree structure of the course's content being under development. In the middle of the screen are located the page multimedia elements

(texts, images, etc.), their style values, and rights parameters as defined in the chosen DTD (these values will override the default values in the XSL template). The right frame provides control over the editing process. In contains the type of page elements that can be included in the current course page according to the DTD/XSD template.

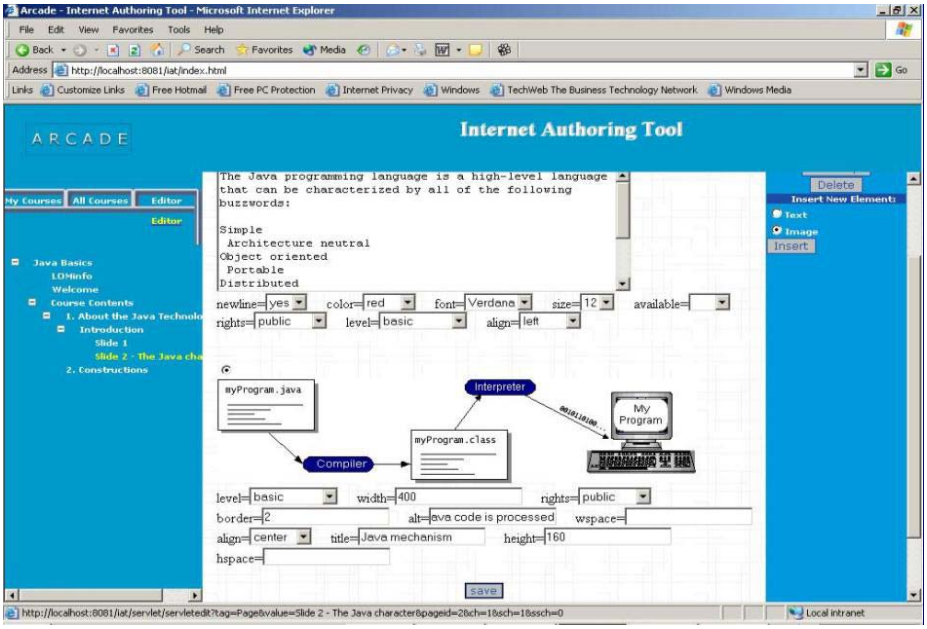

Fig. 7. View of the integrated authoring tool

6 Conclusions and Future Work

In this paper we have presented the ARCADE project. The working prototype of the ARCADE system is available at [25].

As future work, we plan in next prototypes and versions of the system to implement import/export from/to XML for all types of persistent data objects. The plans include extending the coverage of international e-learning standards and specifications (IEEE LTSC, IMS, ADL SCORM) in order to increase the system interoperability with other course support systems. Support for different instructional methodologies can be added to ARCADE in order to provide more flexible opportunities for the instructors and students.

The management of user profiles is another direction of improvement. The profiles can be multidimensional including user's personal differences, abilities, cognitive and learning styles, and personality traits. Instructional designers and content providers will be able to create adaptive courseware materials based on these profiles in order to provide improved learning experience for the students.

References

1. Advanced Distributed Learning Initiative Website, available at: http://www.adlnet.org/
2. Aleksieva A., Petrov M., Bontchev B.: Managing of complexity of e-learning administration in ARCADE. *Proceedings ICL'2003 Conference*, Villach, Austria, (2003)
3. Aleksieva A., Vladinova L., Iliev T.: ARCADE – Software Architecture for Web-Based Education. *Proceedings Next Generation Network Technologies International Workshop*, Rousse, (2002)
4. ARIADNE Foundation for the European Knowledge Pool Website, available at http://www.ariadne-eu.org/
5. Arrington, CT: *Enterprise Java™ with UML™*. John Wiley & Sons, (2001) 22-25
6. Bontchev B., Ilieva S.: Middleware Service Support for Modern Application Presentations. *Proceedings 15th SAER Conference*, St. Constantine, Varna, (2001)
7. Collis B.: Tele-learning in a Digital World: *The Future of Distance Learning*. International Thomson Press (1996) p.9
8. Corporate e-learning: Exploring a New Frontier, WR Hambrecht + Co, (2000)
9. Jones D.: Webfuse: an Integrated, Eclectic Web Authoring Tool. *Proceedings EdMedia'99 Conference* (1999) 1799-1801
10. Dublin Core Metadata Initiative Website, available at: http://www.dublincore.org/
11. Farinetti L., Bota F., Rarau A.: An Authoring Tool for Building Flexible On-line Courses Using XML. *Proceedings XML Europe'2000 Conference*, Paris, France, (2000)
12. IEEE Learning Technology Standards Committee Website, available at: http://ltsc.ieee.org/
13. IMS Global Learning Consortium Website, available at: http://www.imsproject.org/
14. Jacobson I., Booch G., Roumbaugh J.: *The Unified Software Development Process*. Addison-Wesley, (1999)
15. Laudon K., Laudon J.: *Management Information Systems*. 6th edition, Prentice Hall, (2000), 5-7
16. Maruyama H., Tamura K., Uramoto N.: *XML and Java: Developing Web Applications*. Addison-Wesley, (1999)
17. McCormack C., Jones D.: *Building a Web-based Education System*. John Wiley, (1998) 1-28
18. OMG Unified Modeling Language Specification Version 1.4, (2001), available at http://www.omg.org/
19. Robson R.: Course Support Systems – the First Generation. *Proceedings ED-MEDIA'99 Pre-conference seminar "Systems for WWW-Based Course Support: Technical, Pedagogical and Institutional Options"*, Seattle, WA, (1999)
20. Suhl L., Kliewer N., Kassanke, S.: Using Metadata in Web-based Learning of ORMS. *Proceedings IFORS 2002 Conference*, Edinburgh, (2002)
21. Van den Brande L.: *Flexible and Distance Learning*. John Willey, (1993) 2
22. Vassileva D., Bontchev B.: Internet Authoring Tool for E-Learning Courseware. *Proceedings 7th WSEAS International Conerence on Computers (CSCC),* Corfu, Greece, (2003)
23. Vladinova L., Petrov M., Iliev T.: Assessing Students' Performance in Distance Education Courses. *Proceedings International Conference on Computer Systems and Technologies (CompSysTech),* Sofia, (2003)
24. Wampler B.: *The Essence of Object-Oriented Programming with Java™ and UML*. Addison-Wesley, (2002) 217-223
25. Working Prototype of the ARCADE System, available at http://www.e-arcade.net/