МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2005
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2005
Proceedings of the Thirty Fourth Spring Conference of
the Union of Bulgarian Mathematicians
Borovets, April 6–9, 2005

ON TWO ADAPTIVE SYSTEMS FOR DOCUMENT MANAGEMENT*

Vanyo G. Peychev, Ivo I. Damyanov

In this article we observe two document/contents management systems. We discuss the role of XML in development of loose-coupled multi-tier applications. A simple approach involving XML is presented.

Introduction. Software development always starts from a set of incomplete, imprecise, and sometimes self-contradicting requirements. Adaptability, flexibility and maintainability are important features that ensure a fruitful life of software growth [1]. Very often Small and Medium Enterprises (SMEs) need fast rather sophisticated solutions with small budget for development and maintenance. In this article two case studies for development of adaptive document/contents management systems are discussed.

Multi-tier architectures have grown to be standard solutions in various application domains in the last couple of years. The classic solution separates applications into back-end services (database the layer), business services (application layer) and user interface (presentation layer) [5]. This architecture promotes reuse and simplifies maintenance. At the same time designing multi-tier applications without proper separation, emphasizes the problems with adding support for new data.

With the introduction of XML [2] and its subsequent adoption by the major players of the software industry, data can live its own life.

According to W3C (World Wide Web Consortium), XML is a universal format for structured documents and data on the Web. XML is a text format and makes use of tags and attributes [4]. eXtensible Stylesheet Language (XSL) transformation can be used with the XML document to present the data in web browser. XML and XSL allow Web developers to separate data and presentation. XML is ideally suited for the next generation of Internet applications.

A lot of work has been done in the database community on mapping of XML data into and out of relational database systems, specifically, the query processing over such data. XML documents can be exploited to build adaptive systems. In an adaptive system, careful design though is given to isolate objects from one another and isolate data from the behavior.

Adaptive web document system. The first case study provided in the paper demonstrates simple web-based document management system that holds the data manipulated by the objects in a parallel universe. Results of this case study were used in the development of a real document management system deployed to local SME.

*Key words: document management, loose coupling, web application and XML

Adaptive Programming as a name was introduced around 1991. By definition a program is called adaptive if it changes its behavior according to its context [3]. Applications need to be able to communicate with and adapt to one another transparently. Moreover, compound applications (especially distributed ones) need to be designed to anticipate changes – changes in their components and in the way these components interface with other applications. How can we achieve adaptiveness? A generic mechanism to achieve adaptiveness is to use collaborating views, which are loosely coupled. Adaptive software, as currently implemented, is specified by complementary, collaborating views, each one addressing a different concern of the application (structure or behavior).

Multi-tier architecture physically separates the presentation layer or GUI, business processing and database logic. The reasons for developing multi-tier applications are: scalability, application partitioning and enhanced performance, improved reliability, reusability and integration, multi-client support.

There is a great demand to deploy business applications on the web. This demand is somehow stimulated and made possible by the services the web provides and at the same time drives the web's development. Web-based applications are applications that rely on the web as the application infrastructure to perform their functionality and have significant complexity in logic processing. They rely on web browsers and web protocols to provide the user interface in the form of web pages, delivered and connected to the rest of the application. Families of technologies such as CGI, Server side scripting, Client side scripting and series of revisions of HTTP have been developed, and innovation has been conducted to enhance the web for its new role – an application platform. Thus the web-based applications provide interface granularity which is much harder to be achieved in classical UI applications.

When designing adaptive, client/server applications, it is important to design the application as extensible and dynamic as possible. This is important for the separation of data from behavior, scalability, adaptability and maintainability.

Let us state the requirements for our document/contents management system: The main purpose of the system will be to manage documents with different complexity and size. (A) Adaptive document management system should manage different types of documents with non-editable and editable contents. (B) Some of the editable content is coming from nomenclature. (C) Editable contents of the documents should be stored in a manner to enable aggregation and better search. (D) Client (for editing document contents) should be web-based. (E) Non-editable (and set of editable) contents can be changed over time as well, since already stored documents should still be rendered as they were made. (F) Adding management of new documents should not reflect on server side code. (G) System maintenance should not require extra knowledge.

Design and implementation highlights. We will not emphasize the proprietary extensions of neither SQL servers nor Server side scripting. We assume that the used SQL server has facilities to return records set as XML document and has ability to store row that was passed as XML document. Such facilities are available in most popular SQL Servers. If they are missing we can use some adapters to provide these features to the application.

Our simplified approach is to use one table for the document editable contents where each column is either reused between different document types or unique for specific doc-

ument. In another database table we have stored XSL/XSLT documents. They are used to "add" non-editable contents of actual document and editable capabilities (i.e. HTML form elements as <input>, <select>, <textarea>, etc.) if is needed. Requirement (B) leads us to another table that contains all the nomenclature in hierarchical manner (row records with additional info for parent/child relations) (Figure 1).

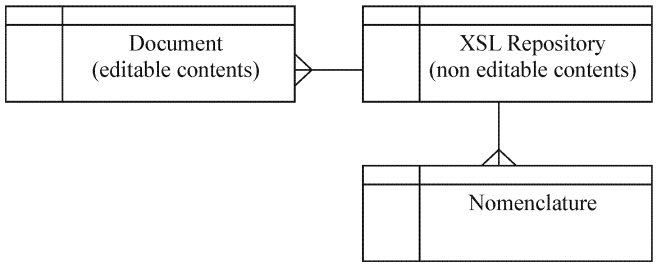

Figure 1

Processes of showing/editing document start with sending to the web tier a request from the client with specified document ID. The business tier processes the authorization credentials and requests from data provider XML with document editable contents, XSLT document with non-editable contents and instructions for proper rendering (for editing or review). Additionally, a nomenclature is requested (if necessary) and attached to the final XML document (Figure 2).

XML Engine processes the returned document according to transformation schema (in XSLT) and produces valid XHTML file.

After editing the document the client posts back to the web tier the document contents. Web tier loop through form parameters collection and compose XML document that is processed by database server and stored into document table. Adding new documents reflects on creating new XSLT and storing in XSL repository. If needed, some extra columns using web interface the "wide table" is altered and the necessary columns are added.

The proposed solution has enough flexibility to cover wide spectrum of documents. The only disadvantage is inability to store lists. This is one of the directions for further research to be done.

Relational model for document representation. Another way to store and present a document structure and content is to use the relational data model. Storing a document structure in RDBMS is not a trivial task. The usage of a relational data model involves the addition of semantic knowledge of the document structure, which is stored into a DB objects. The aim is for every document to be added information about the way of its visualization on the client browser. This information is stored when the document is created (Figure 3).

The document is considered as an ordered sequence of blocks. Each block contains information for its place in the view of the whole document. The visual elements are contained in blocks.

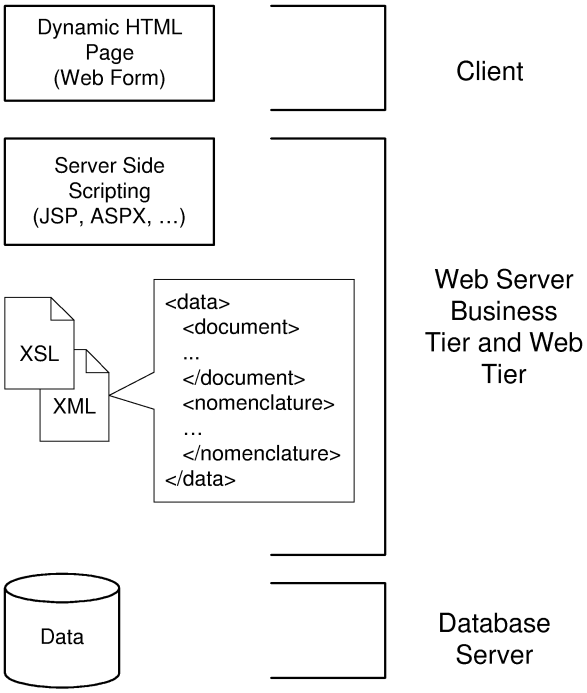

Figure 2

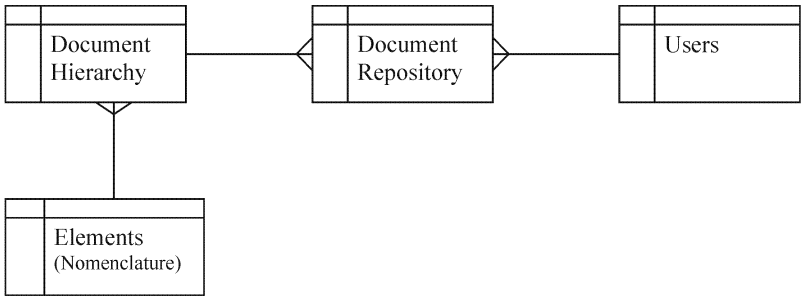

Figure 3

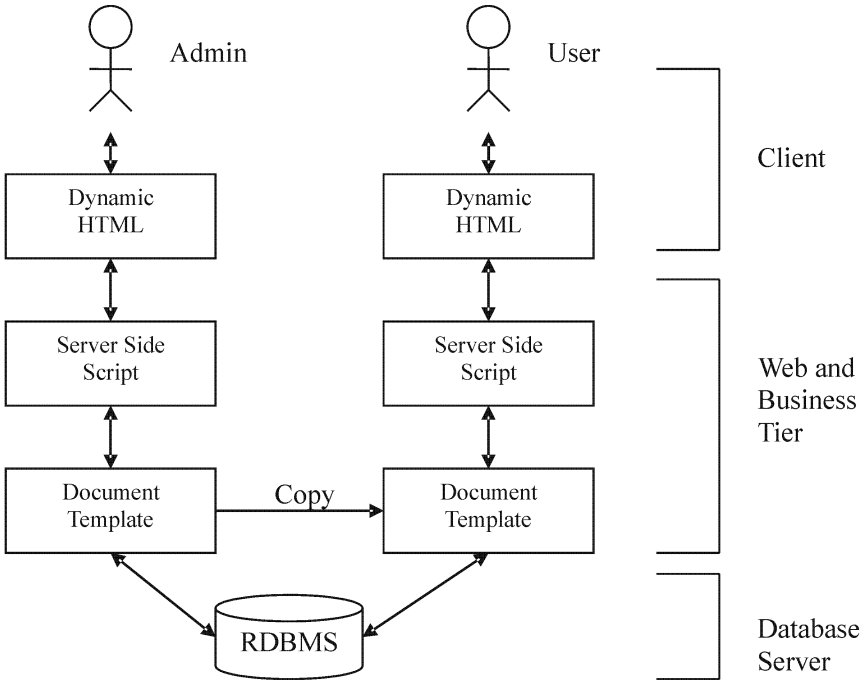

Figure 4

Concerning documents life cycle there are two important activities:

- Generate a document template (the document hierarchy is created);
- Manage a document content (from users).

The generation of a document template and its storing into the document repository is done only by an authorized user called a document administrator. The document template could be in two states: development and production. The administrator can change the state. After the document template is published in a production state it could be used by a user. The user can fill the document template with personal data. Only an authorized user can access the document content according to the rules:

- For every user all versions of all own documents are kept;
- For every type of a document all versions are kept;
- Only an administrator can see all documents.

In general the system has the following model:

The solution allows support for a wide set of document templates for many users. Tracking versions is very important for scalability of the solution. Some disadvantages are connected with the complexity of the interface for the creation of document templates.

Conclusions. In the article we present two solutions for a document repository and content maintenance. Both solutions are influenced by real requirements for document management in SME. Each solution has its own advantages and disadvantages. In our approach for developing a loosely coupled multi-tier and an adaptive web-based application we have achieved maximum flexibility through the use of XML. Both approaches were used in the development of real document management systems deployed to local SMEs. The implications of these approaches are important for designers of mission-critical, enterprise-scale systems.

REFERENCES

[1] L. Osborn. Information systems lessons learned. NSF Informatics Task Force, Educating the next generation of information specialists, Alexandria, Virginia, 1993. National Science Foundation, 40–41.
[2] World Wide Web Consortium. Extensible Markup Language (XML) 1.0. February 1998. (http://www.w3.org/TR/REC-xml).
[3] K. Lieberherr. Adaptive Object-Oriented Software. The Demeter Method. PWS Publishing Company, 1996.
[4] D. Hunter, K. Cagle, D. Gibbons, N. Ozu, J. Pinnock, P. Spenser. Beginning XML, Wrox Press, 2000.
[5] K. Renzel, W. Keller. Three Layer Architecture, Software Architectures and Design Patterns in Business Applications, TUM-I9746, 1997.

Vanyo Georgiev Peychev
Department of Computer Sciences
Faculty of Mathematics and Informatics
University of Sofia
5 James Baucher Str.
1164 Sofia, Bulgaria
e-mail: vanyo@fmi.uni-sofia.bg

Ivo Damyanov
Department of Computer Sciences
South-West University
66 Ivan Mihailov blvd.
2700 Blagoevgrad, Bulgaria
e-mail: damianov@aix.swu.bg

ДВЕ СИСТЕМИ ЗА УПРАВЛЕНИЕ НА ДОКУМЕНТИ

Ваньо Г. Пейчев, Иво Й. Дамянов

В доклада се разглеждат две системи за управление на документи/съдържание. Обсъдена е ролята на XML в създаването на непълно алгоритмизируеми приложения.