

A Flexible System for Request Processing in Government Institutions

Miroslav Zarić, Milan Segedinac, Goran Sladić, Zora Konjović

University of Novi Sad, Faculty of Technical Sciences,
Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
E-mail: miroslavzaric@uns.ac.rs, milansegedinac@uns.ac.rs, sladicg@uns.ac.rs,
ftn_zora@uns.ac.rs

Abstract: This paper presents a solution for an electronic document handling system for government institutions. The proposed solution introduces a new system aimed at handling various administrative requests with minimal disruption to standard end user habits as well as minimal requirements in terms of end user training. During the development and testing phase, chosen software architecture has proved itself as robust and adaptable to requested changes. As the end user submits a request from a familiar office suite environment, usual problems, such as refusal to engage with new software and complaints about a steep learning curve are avoided. The approval of requests is executed through the standard web application. The system relies on well-defined user roles and an established workflow. The proposed solution can easily be adopted for handling various types of requests, as long as their processing fits into the deployed workflow. This system is designed with specific intention to be a low-barrier entry electronic document management system for existing administrative workers. Use of a standard office application suite for document submission allows existing users easy transition. This eliminates the need for user training and consequently reduces the disruption to the normal workflow.

Keywords: document management; workflow management; e-government; active documents

1 Introduction

E-government aims at enhancing the efficiency of governmental administrative processes, improving the quality of their services and reducing operational costs through the application of ICT (Information and Communication Technology) technologies. In this paper we present a system for handling administrative requests in a digital form (e-requests).

One of the trends in the development of e-government systems is to put emphasis on electronic document management (EDM) systems. EDM systems deal with the management of documents [1]. The document may be a part of a particular

business process, in the sense that it requires access to the document by individual staff undertaking separate activities according to a particular sequence guided by some procedural rules [2]. Workflow Management Coalition defines workflow as the computerized facilitation or automation of a business process, in whole or in part [2]. In computer science literature, business process and workflow terms are often used interchangeably; in this paper we regard them as synonyms.

Workflow management system (WfMS) is a system that completely defines, manages and executes workflows through the software in which the order of execution is driven by a computer representation of the workflow logic [2]. One of the main advantages of WfMSs is moving the focus from the automation of single process activity, to the overall management and improvement of the business processes. EDM systems are usually implemented using the document-oriented WfMS. Due to their advantages, government agencies tend to implement their services on document-oriented workflow platforms. The problem of document processing has long been recognized as a critical aspect in the enterprise productivity [3, 4, 5]. In decentralized working environments, where many people affect the contents of documents, efficient collaboration on document editing is a key feature, and a collaborative environment must take care of collisions that can arise from simultaneous access to the documents. This problem has been the focus of many research papers [6, 7, 8, 9, 10, 11, 12].

We propose an integrated system with the aforementioned characteristics through a combination of different, readily available, technologies. A standard word processing office application is used for the creation and direct submission of requests to the processing web application. Therefore, users continue to use the same application they have used for creating the printed request forms. The web application serves as a central module and as a standard access point for users involved in request processing and approval. A workflow engine is embedded in the web application. Access to the system is controlled by an authentication and authorization modules. Internal document format is XML-based, allowing for future expansions. We have observed custom business processes in different government institutions, and a case study described in this paper is developed for the local provincial government.

The rest of the paper is structured as follows. Section 2 reviews the related work. Section 3 presents basic motivations and requirements. Process description for chosen case-study is presented in Section 4. The application architecture overview is given in Section 5. At the end of the paper, a conclusion is given with some outlines for further development.

2 Related Work

This section covers two topics: research on document-based workflow systems and research on the application of office documents enriched with software components - *active documents*.

In [13], authors propose a methodology for developing hypermedia information systems on the basis of document-based workflow. The proposed methodology focuses on corporate systems that require the capability of handling complex business functions. It adopts a document-based perspective consistently through all phases. The work reported in [14, 15] addresses the problem of creating an information infrastructure and services for distributed and virtual organizations, and, particularly, the integration of two key enabling technologies, namely workflow and document management. Paper [5] proposes an XML document centric workflow management system that exploits the advantages of the XML documents, while having the full functionality of workflow management system to execute other activities. Paper [16] proposes a framework for document-driven workflow systems that requires no explicit control flow and the execution of the process is driven by input documents. The solution presented in [17] extends web-service based workflow engines with human interaction via email. A document-based dynamic workflow system, that is particularly suitable for agile business processes in which required tasks and their sequence flow may need to be determined dynamically, is proposed in [18]. Experiences in using Java Business Process Management (jBPM) for document flow are given in [19].

Any document workflow system will face the challenge of collaborative editing of document content. As stated in [12]: "Collaborative editing enables a group of people to edit documents collaboratively over a computer network." The purpose of collaboration is to achieve a common goal. Most group editing tools are using the copy/modify/merge paradigm, supported by three methods executed on a shared repository storing multi-versioned objects: checkout, commit and update. In collaborative editing two major principles of reconciling different versions of a document have become predominant: state-based merging approaches often used in versioning systems such as CVS [20], and Subversion [21], and operation based approach [22]. In recent years, since XML has become a de-facto standard format for representing structured data, special attention is given to collaborative editing of XML documents. Usage of XML documents presents a challenge and an opportunity, since most conventional operation based approaches (algorithms) such as dOPT [6], GOT [23], GOTO [24], SOCT2 [25] and similar [26, 27], view documents as linear structure. An approach to collaboration over hierarchical documents, treeOPT algorithm as well as asyncTreeOPT (for asynchronous editing) is described in [12]. In [28] collaborative editing of XML documents (in peer-to-peer environments) has been further discussed.

Term *active document* designates a document that, besides the data and structure, contains some software components (e.g., macros or scripts) [29]. Since it

combines data with processing components, an active document can embed processes such as data retrieval, data acquisition, transactions, workflows or documents archiving [30]. An overview of several case studies in which active documents are used follows.

In the United States Patent and Trademark Office (USPTO) report [31], the usage of an active Word document named *Electronic Filing System – Application Body eXtensible Markup Language, EFS-ABX*, was proposed for the creation of patent application specification in XML format. In the report the following benefits of using active Word document were identified: 1) ease of use, 2) simplified image management, and 3) simplified client side workflow. In the report [32], National Institute for Health and Clinical Excellence (NICE) has proposed active Microsoft (MS) Word 2007 documents for the creation of guidance documents in XML format. The following advantages of active MS Word 2007 documents have been identified: 1) ease of the manipulation with XML files, 2) many people have the appropriate Microsoft Office skills, 3) there is a degree of backward compatibility to MS Word 2003. Paper [33] describes an active Word document that creates an environment for editing the material in the publishing industry. The paper puts emphasis on the advantages of using XML. The research presented in the paper [34] is an example of applying active Word documents in medical practices. The paper states that the most important benefit of using Word for record keeping in medical practice is the fact that most of the doctors are already familiar with Word, so there is no need for additional training.

3 Basic Requirements and Motivation

Our case-study system is developed to improve internal e-government services for the provincial government of the Autonomous Province of Vojvodina, in the Republic of Serbia. In such an institution, there is a substantial number of documents in circulation, and their prompt handling and tracking becomes a challenging task. Furthermore, permanent storage of large quantities of paper documents, often in multiple copies, is a growing problem. These documents are commonly created using a word processing editor (commonly some office suite, usually MS Office), printed and then handed over to an employee in charge of handling specific requests. This is a starting point of a business process in which documents proceed along the chain of employees, each of them making some kind of a decision about the document and/or modification of the content. Handling requests in such a manner has not changed much for a substantial amount of time. Although this type of request processing system is common, easy to understand, and well accepted, in a digital era and e-government environment it has its shortcomings: the overall process is slow, printing a large amount of documents just for internal use is a huge waste of resources, and, as mentioned before, the storage space requirement becomes unacceptable. However, even though the

documents are already electronic at its origin, paper documents are, in some cases, still required by applicable laws or internal regulations, and they cannot be, in the near future, completely avoided and removed from the business process.

In order to alleviate some of the problems, the provincial administration was seeking an appropriate solution for electronic handling of internal requests. Such a system has to provide the same capabilities as the paper-based system, but should accelerate the whole process, automatically notify relevant participants, and make it easier for them to perform their duties. Although there are well established solutions for electronic document management, and some of them are already deployed in the provincial government, there were some special requirements that lead to the development of a specialized system. The current business model for request handling, as well as their special demands and requirements were obtained through a series of interviews with employees and managers with the following findings:

- MS Office is already in use as the standard office suite, and employees have experience with using it;
- A set of standardized request forms exists (as Word documents);
- There is an established workflow for processing internal requests;
- Regular users should experience minimal change during transition to the new system;
- The system should be able to adapt to any possible future changes in the workflow;
- The system should alert the relevant participants if a document is waiting for their attention;
- The system should allow for easy addition of new form templates;
- The system should be accessible from the local intranet.

Based on these premises and requirements, a number of different issues had to be resolved, such as document representation, appropriate document storage, appropriate way of representing the business process, and the appropriate notification of the users. After considering different options, we have come to the following conclusions:

- XML, with its inherent support for structured data, has been chosen as the format for document representation - such a choice is further supported by availability of the software for XML manipulation; In accordance with this, a native XML database will be used as the document storage module;
- It would be beneficial for the system to use MS Word to fill-in and submit electronic requests. Thus, end users would continue to use a familiar environment for filing requests. To have a modifiable process workflow, a workflow (process) engine should be used;
- Since the application core should be accessible over the intranet - it is created as a web application, and
- Apart from MS Word, other parts of this integrated system should use open-source and free software to minimize the cost of service.

A document management system, namely Alfresco, is already implemented in the provincial government institutions. However, this system is preferably used for specific purposes, usually by higher level administrative employees. On the other hand, the request processing system is intended for widespread use, as almost any administrative employee is allowed to generate a request. Alfresco is a general purpose document management system, and it has a lot of advanced functionalities, but it also takes some training for all the users to get used to it. A large base of users require additional administration in the system, with carefully crafted user groups and permissions. Since most of these functionalities are advanced, and rarely needed by ordinary users only interested in submitting the request, simplicity was the most important issue.

Although the proposed system inherits most of the features commonly present in any document management system, its main advantage is its overall simplicity. For ordinary users, creating a new request is reduced to filling out a Word document template.

All requests are transformed to a single document type – XML, simplifying later operations on the document content. Any later interaction with the system is done through a web interface. As an additional advantage, the web pages used for presenting the content to a user, are created on-the-fly by converting the request itself (in XML representation) into HTML format. Therefore, only the skeleton HTML is manually coded, while any request conforming to the defined XML schema will be properly transformed for viewing and/or editing. These two features enable easy integration of new request types to the system.

4 Process Description

Figure 1 displays the process graph in standard Business Process Modeling Notation (BPMN) notation. This graph represents a blueprint for process execution and is used to control the program execution (instead of hard coding the business logic into the application). The main concepts are nodes, transitions and tokens. Tokens are used to track the process execution. There are different types of nodes, and a *task node* is used to describe a point in the process where human intervention is required. The process execution progresses by moving the main (root) token from the start node to the end node along the possible paths defined by the graph.

One process instance (one execution of the process graph) is started when an end user submits an eRequest through the Word active document template. Users involved in the process can play different roles: *Originator, Supervisors, Director of Administration, Authorized Deputy Directors, Responsible Chief Officers, and Executors*.

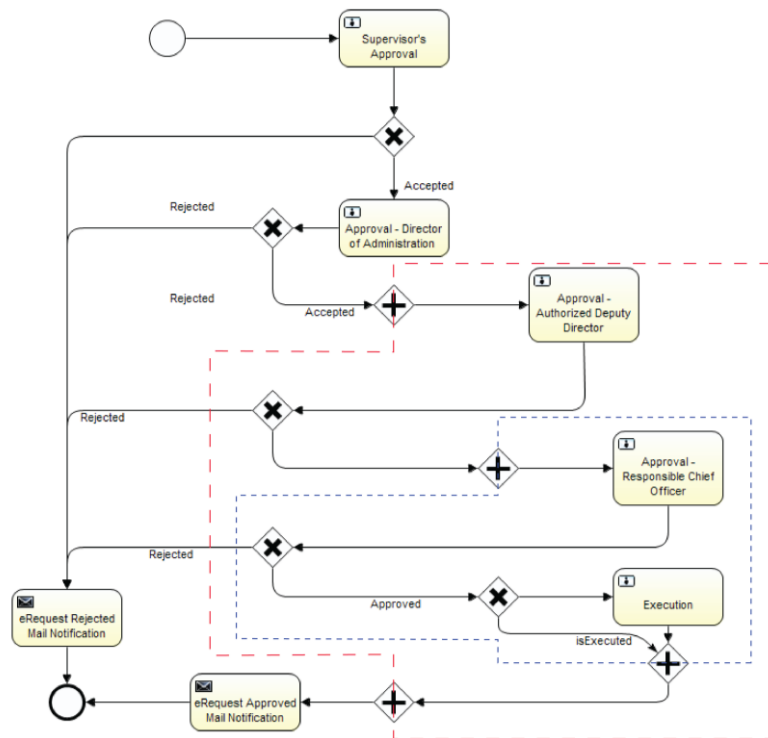


Figure 1
Process graph

The *Originator* prepares the request in MS Word by filling out a special template document, prepared as an *active document* that contains a macro enabling it to directly communicate with the server-side application. When the document is ready, the end user will submit the request directly from within the document by pressing the “Send” button embedded in the template. If the system accepts the request as legitimate (the user is registered in the system and entitled to submit the request), the task for its supervising officer is created (task *Supervisor's Approval* in the graph). During the task assignment phase, an automatic email message notifying *Supervisor* is generated (this is also automatically done for all other tasks). Only the supervisor relevant to the current user (in the same organizational unit) will receive the task. In any task, an actor can accept, decline or modify and accept the request. If *Supervisor* declines request, it is forwarded to the *eRequest Rejected Mail Notification* node, and appropriate notification is sent to the request *Originator*.

If the *Supervisor* approved the request, it is forwarded for further approval to the *Approval – Director of Administration* task. The *Director of Administration* reviews the task, and as before, if it is denied, the process enters the *eRequest RejectedMail Notification* node. If the request is to be approved, he/she has to

choose one or more subordinate users that the request will be passed on. If *Director of Administration* assigns request to more than one of her/his *Deputies*, an implicit “dynamic forking” (when more than one approval at the same level of hierarchy is needed, but the exact number of actors is determined only at runtime) is performed, and execution path (marked with larger dashed line) is multiplied and child tokens created. The main (root) token of the process will stay in fork node until all generated child tokens are collected at the join node. The join node of this fork is presented as the last parallel gateway join node before the email *eRequest Approved Mail Notification* node.

Similar decision-making and forking is performed at the *Approval – Authorized Deputy Director* task node, when the request is assigned to multiple *Responsible Chief Officers*. The rest of the process execution is straightforward: the *Responsible Chief Officer* reviews the request, decides if it's acceptable, can approve it and pass it to his/her subordinates for execution, or can mark this part of the request handling process to be executed by himself. Finally, tasks assigned to *Executors* can only be executed. In this phase, since all necessary approvals have been obtained, the request cannot be denied final execution.

The process will make the transition to the final *eRequest Approved Mail Notification* node only when all tasks are completed. If the request has been rejected at any phase all pending tasks will be canceled upon entry to the *eRequest Rejected Mail Notification* node. After sending a mail notification to *Originator* (either with final approval and success notice, or with rejection notice) the process will enter to its final *End* state and will be archived.

Additional requests were introduced when the system went into production – the task assignment model has to be extended to support not only actors and groups, but to allow for: a) replacement for a specific user, b) reassignment of already assigned group tasks.

Replacement for specific user was needed to accommodate situations where an actor may be occupied for prolonged periods of time, and they have an opportunity to re-delegate their tasks to a registered replacement, but keep ownership of the tasks. In comparison to standard group tasks, the task stays assigned to the originally intended actor, but upon completion, the record is made that the task was performed by a replacement on their behalf. This implementation required user administration and user handling model to be extended.

The second request came into focus when it became obvious that some tasks, required to fulfill the request, can take substantial time to accomplish, and that it is possible, although rare, that one person has indeed started the task execution, but is currently unable to complete it. In the standard assignment model, tasks that were visible to the group are changed to personal tasks, when the final assignment to one actor (from the group) is performed. Other members of the group can no longer access that task, unless the assigned actor (or an administrator) returns the task to the group. Although this solution is acceptable, since the assigned actor

sometimes is not in position to do it her/(him)self, so in order to speed up the overall process execution, as well as to lower the burden on system administrators, the system is extended to allow other members of the group to see such tasks. The task is displayed alongside information to whom it is currently assigned, and when its execution was started. Other members of the group are allowed to “capture” the task and thus perform reassignment.

5 Software Architecture

In this section we'll present the global architecture of the implemented system, document structure (active documents), and software architecture of the server-side application.

5.1 Global Architecture

The global architecture of the system is presented in Figure 2. Required authentication is performed by user's operating system, wherein authentication information is stored on the LDAP (Lightweight Directory Access Protocol) server. The employee submits their request to the e-request application by using *active documents* and MS Word. An embedded macro transforms the request to XML document. Upon receiving the document, an e-request application creates an instance of the process described in the previous section and starts its execution. Users involved in request processing access the application through the web interface, and the application server consulting LDAP server performs authentication.

The server-side of the system is implemented using the Java open source technologies. The e-request application code relies on the different open source libraries such as *JBoss jBPM* workflow engine and the *Xalan* XSLT processor. The access control is performed by the *COBAC* (*context-sensitive access control model for business processes*) [35, 36] implementation. The received requests are stored in the *eXist* native XML database, while access control policies and workflow data are kept in the *MySQL* database. We used *Apache Tomcat* as the application server.

The system is implemented in the environment of a local area network. Access from the outside world is guarded by applied network policies. Additionally, some measures are implemented on the server side in order to protect the system from involuntary errors or malicious attacks. Wireless network access to the local network, due to the sensitivity of governmental institutions, is restricted to registered computers and users.

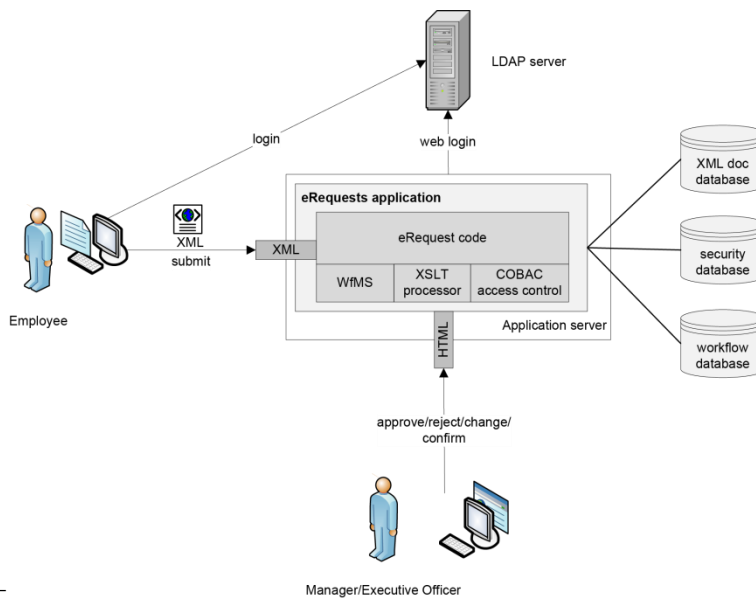


Figure 2

Global architecture

As macro-enabled Word documents (discussed in following section) are used to create requests, upon receiving the request on the server side, the XML is checked for conformance with the defined XML schema. In addition, the incoming XML string is checked for XSS-like (Cross Site Scripting) attacks. This step is necessary since XML is later transformed to HTML, and any embedded JavaScript could potentially be a threat. The SQL Injection attacks are not possible since request data is stored exclusively in the native XML database, and only process data, generated internally, is stored in the MySQL database.

Although potential for CSRF (Cross Site Request Forgery) in the restricted local network is relatively low, a proper protection is implemented on the server side. This type of attack is checked for when the request editing and/or approval phase is executed and HTTP requests are being processed.

Databases (both native XML and MySQL) are on separate servers, and access is allowed only from application server, with appropriate user credentials. Access from other networks and other computers in the local network is forbidden by applied network and server policy rules.

If outside access to the system is to be allowed in the future, it will be performed through a server in a DMZ, over an HTTPS connection. This server will be routing traffic to the application server in the local network.

5.2 Active Documents

The active documents are implemented as MS Word 2007 documents, with processing component implemented as VBA macros. Analysis of the preexisting request forms documents showed that their primary contents are: free text input fields, checkboxes, tables, and images. The active documents were designed to automate the document conversion to XML (in further text *XML requests*), according to the proposed XML Schema, and send it to the server. The overall schema is shown in Figure 3.

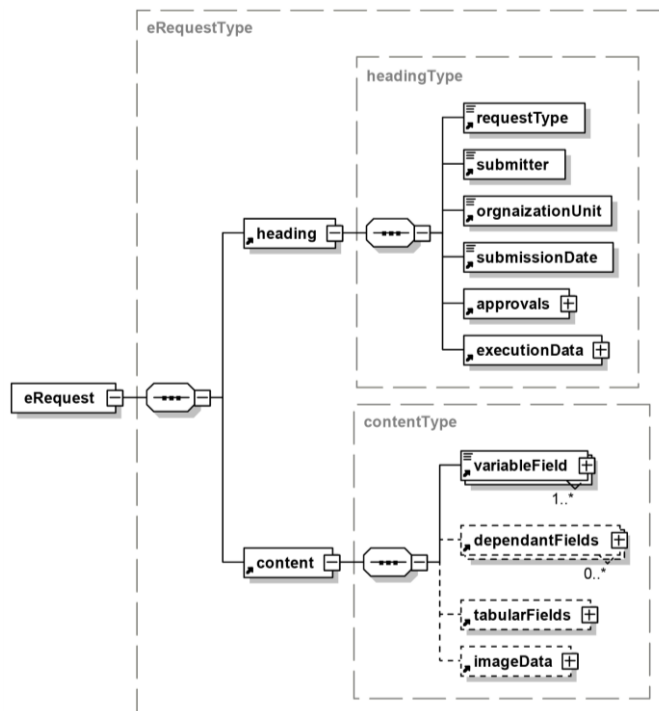


Figure 3

The overall XML Schema

Each XML request consists of two sections: *heading* and *content*. Heading contains the request submission date (*submissionDate*), the employee's data (*submitter*, *organizationUnit*), and document type (*requestType*). Other information (typed by the user into active document) is mapped to the subelements of the *content* element. When designing active documents to serve as eRequests templates, a possible solution was to use Rich Text Content Controls for free text input fields, and to map them to the XML Schema. However, such an approach tends to be inappropriate because of a vast number of forms, and the frequent need to modify them. Instead, we have simplified the documents by using two-column tables in active documents to represent text input fields. The first column contains

field names, while the second one contains field values (Figure 4). Such an approach allowed us to observe each table row as an ordered pair (*field_name*, *field_value*). Adding a new text input field to an existing form is then reduced to inserting a new row into the table. The left column is then locked to avoid accidental editing. Such an approach does not require macros to be altered for each new field.

Date:	
Time:	
Conference hall:	
Host:	
Occasion:	
Number of persons:	
Contact:	
Should be provided:	

Figure 4

Request form containing text input fields

Some forms, in addition to the text input fields, contain checkboxes. Here, a column is added to the table (Figure 5). In such a table, each row is observed as an ordered triple (*checkbox*, *field_name*, *field_value*).

<input type="checkbox"/>	Refreshments	
<input type="checkbox"/>	Lunch	
<input type="checkbox"/>	Security	
<input type="checkbox"/>	Transportation	
<input type="checkbox"/>	Parking	
<input type="checkbox"/>	Sound system	

Figure 5

Request form containing checkboxes

Some request forms contains tabular data (Figure 6). Data from tabular forms are mapped to *tabularFields* XML element, containing *rows* elements. The first *row* contains XML elements representing table columns definitions, while other table rows cells are mapped to the *variableField* elements in XML.

No.	Product name	Measuring unit	Quantity
1.	Printing paper	Package	5
2.	CD	Item	60
3.	DVD	Item	20
4.	Marker	Item	10
5.	Notebook	Item	2

Figure 6

Request form containing tabular input fields

With this approach to document processing, transformation of the request document content to XML document is achieved simply by iterating through the table rows. Additionally, some request forms can contain image elements (such as the request for printing an ID card). Therefore, it was important to support transport of image data from within the Word file to the server. To support this, Picture Content Control needs be used in the Word document, but also appropriate transformation of image binary data was needed to facilitate transport over HTTP. MS Word (2007 and higher) supports adding an optional *CustomXMLParts*. Properly bound to Picture Content Control element, it provides a Base64 representation of image binary data, which is suitable for insertion in the XML request. Upon the XML document creation, the macro sends it to the server.

5.3 Server-side Architecture

As mentioned before, Java has been used to develop server-side application. Figure 7 displays classes of the module receiving the requests sent from Word application. The central class of this module is *AcceptRequestAction* which accepts the sent XML documents. Different operations on XML documents are modeled with the *XMLUtil* class. The *LDAPService* class is used to access the employee's information (*User* class), while *COBACService* is used to perform access control as described in [35]. The *XMLDBRepository* class implements access to the native XML database. *AcceptRequestAction* uses the classes from the jBPM API to create and start a process instance, and to populate process instance with appropriate variables. The class *UserProcesses* conveys information about the processes that the user has created.

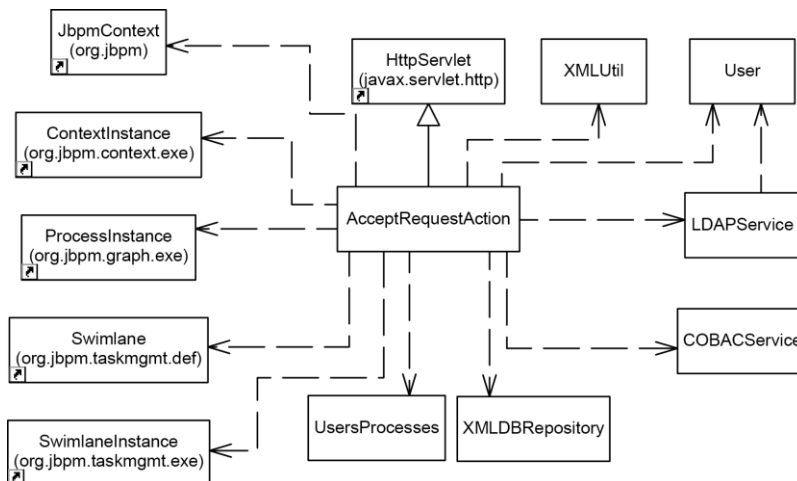


Figure 7

Classes for accepting request

Figure 8 displays classes involved in task processing, and classes involved in non-task related actions, such as searching and reviewing of requests. Since all tasks are accessed and handled through the web interface, the base class of the hierarchy – *TaskAction* – is extended from *HttpServlet* class. The class *TaskAction* contains some basic task-related functionality, and uses classes from the jBPM API to access information about the process instance. *TaskListAction*, using the jBPM API, generates the list of tasks for the current user. The *TaskViewAction* class allows the current user to access the assigned task, view the content of requests, and to make decision about it. The Loaded XML document (eRequest) is passed through an appropriate XSLT to generate the final web page. The *TaskEditAction* class, though similar to *TaskViewAction*, prepares additional variables and passes the XML document through different XSLT to generate fully editable content. *TaskProcessAction* is used to evaluate user's response to the presented task. It adds additional content to the request's XML document - the name of actor that performed the task, and his/her decision. The *ListMyRequestAction* class allows the current user to view a list of active requests he/she has submitted. This is important because content of the document is changing as document progress through workflow, and additional information about approvals and notifications/explanations are added. Additionally, the user can withdraw a request, canceling all pending process tasks, and finalizing jBPM process instance.

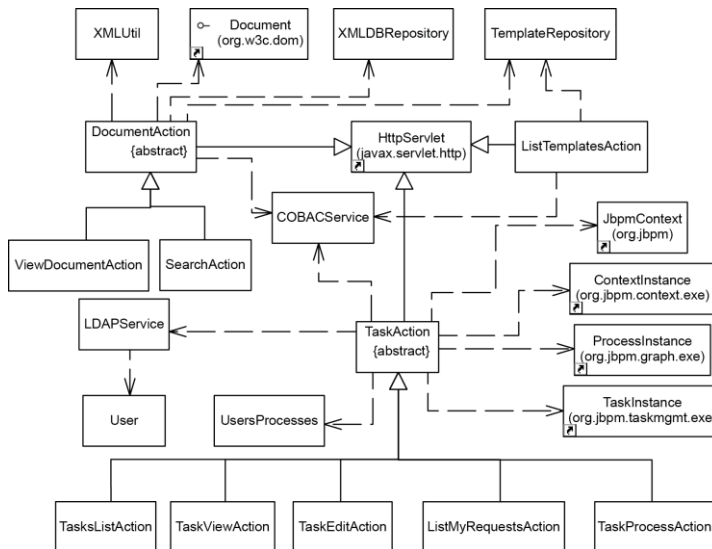


Figure 8

Classes for processing request

Actions that are not directly related to task execution are modeled through the abstract class *DocumentAction*, and its descendant classes *ViewDocumentAction* and *SearchAction*, used for document preview and for document search, respectively. A user can search document on different criteria: request type,

request status, and document originator. The Request type list is automatically generated by extracting header information from documents in repository – eliminating the need for maintenance of web application when new type of request is added to the system. The Request status list contains fixed options: active, finished – accepted, finished – rejected and withdrawn.

An additional class *ListTemplatesAction* produces a web page containing a list of macro-enabled Word documents representing eRequest template forms. The documents are stored in a dedicated folder on the server. This class scans directory and generates a list of download links on a web page.

5.3 XML Document Transformations


XML document transformation can be observed from two different standpoints: the change of XML documents content as a result of performed action, and the transformation of document content into a presentation form. The first one is a direct result of an action performed in the workflow system and the XML document content is edited. Critical operations on the XML document are node insertion, node deletion, and change of content and attribute value [28]. In our system, document editing is encapsulated in a process task-execution environment. While performing a task, users can confirm, reject or edit and confirm an eRequest. In either case a new node containing approval or rejection, with explanation and user id, will be inserted as child node of the *approval* node. In the case of editing, only certain users are allowed to change certain parts of the document in any given moment. As a result of this, most of the time, only one user will be allowed to perform her/his task over a certain part of the document. If this is the case, no conflicting (concurrent) editing will happen, and the document will be simply stored to the document repository. However, as mentioned earlier, there is a possibility of parallel tasks execution. In this case, an automatic document merger, based on document timestamp and element position in document, is performed.

Presentation layer of the web application is built by using Java Server Pages - *jsp* based dynamic pages and XSL transformations for displaying request details. Additionally, XSLT is used whenever original XML document needs to be transformed to presentation format i.e. XHTML. Hence, task processing and request document preview/modification is handled on pages generated through XSLT. Since request data is originally in XML format, this approach gives us flexibility, as only XSLT needs to be adapted if the XML format is changed in the future. Approach through XSL transformation files proved as a viable solution during development and testing phase when some changes to the originally envisioned structure of XML were requested. There are three different XSLT files used for transforming request data to three different states:

- Simple preview with no additional controls,

- Preview with controls for accepting/rejecting requests, and finally
- Transformation for editing of full content of request.

Created XSLT files convert XML documents to XHTML representations that are seamlessly integrated in overall application design. Additionally, image data is displayed in resulting XHTML by converting Base64 encoded data to image representation. Some request types contain images, allowed format is jpeg, and images are usually small so direct transformation was a feasible solution. Figure 9 displays the transformation results for editing mode. Special styles were developed to properly format the document for printing.



Republic of Serbia
Government of Autonomous Province of Vojvodina

eRequest Processing

Request for ID Card

Current User: XXXX YYYY

[Home](#) | [eRequest forms](#) | [My Requests](#) | [My Tasks](#) | [Search](#) | [Log Out](#)

eRequest No.: 121

Submitted by: Goran Sladić

Org. Unit: Provincial Secretariat for Economy


Date Submitted: 2012-04-01

Status: In Process

Implemented (executed) by:

Name:	John
Family Name:	Doe
Title:	Mr
Organization:	Provincial Secretariat for Economy
Function:	Visitor
ID Card no.:	
Year Issued:	2010
Place Issued:	Novi Sad
Language and script to be used for printing the ID card:	
Comment:	None

Sample image



Reason for editing original eRequest:

Save changes and forward request

Choose executive officer:

Mary I

George

John

Ted

Cancel

Already Approved By:

2012-04-01

Edward
head of the authority

Figure 9
XSLT result for editing mode

Conclusion

In this paper we have presented an approach for the handling of administrative requests as electronic documents (eRequests). Although different DMS systems are available, and one (Alfresco) is installed in the provincial government, special requirements guided the decision to develop a new system. One of the key benefits of the implemented system is low-entry barrier for end users, achieved by using their usual workplace software environment as a tool for creation and submission of electronic requests, avoiding common uploads to different platforms and the need to train the users to work in a new environment. We have preserved their existing document templates, and enhance them to active documents.

Unlike common web forms that usually need to be completely filled out in one session, or intermediate state stored on the server, our approach allows users to work on the document, store it, continue later, and submit it when completed. Stored document with filled-in data can be later used as a starting point for a new eRequest, without the need for special template archiving mechanism. Also this approach allows administrators to easily create a new eRequest form – usually by editing an existing one and uploading it to the designated directory on the server. Some parts of the document (such as the title block) are automatically used to register a new type of request without the need to maintain special registry on the server.

To control the document processing, a workflow engine has been used. This approach gives the system a flexibility regarding possible changes in the established workflow. XML is used throughout the system to represent eRequest documents, allowing for easy manipulation and transformation.

These approaches have proven to be valuable even during the development and testing phases, when changes to the workflow were requested, as well as changes to the existing eRequest templates and creation of new eRequest forms. Initially, ten existing word templates were converted to eRequest active documents, and during this period, some were changed or removed from the system, while only one session was needed to train administrators to create new templates on their own, so they later produced ten new templates autonomously. This number will be growing as more paper requests are required to be processed through this system. Transition to electronic documents significantly reduced resource consumption in request processing (prior to introduction of this system, usual number of paper copies was 4 to 5 - this is now reduced to 1, which is archived). Annually, this amounts to a reduction of a few thousand paper sheets that no longer need to be printed and archived, and with more requests handled through the system, the difference will be even larger. Request processing time is also significantly reduced, since embedded workflow engine generates the appropriate notifications and progresses to the next task immediately upon completion of the previous one.

Further development will include adjustment of the user interface for mobile device access. This also implies that a new security layer needs to be introduced to

enhance security of the application when accessed over public wireless networks. Additionally, access control will be strengthened by introducing document and document-fragment-level access rights. For this purpose we plan to use eXtensible XML Access Control Framework [36, 37, 38]. Furthermore, ontologies will be added to increase e-requests semantics and appropriate XML elements representing this knowledge will be introduced into XML document representation.

References

- [1] International Organization for Standardization (ISO). (2001): ISO IEC 82045-1: Document Management – Part 1: Principles and Methods, ISO, Geneva, Switzerland
- [2] Hollingsworth, David: The Workflow Reference Model, Workflow Management Coalition, Cohasset, MA, USA (1995)
- [3] Baresi, L., F. Casati, S. Castano, M. G. Fugini, I. Mirbel, and B. Pernici: WIDE Workflow Development Methodology: Proceedings of the International Joint Conference on Work activities Coordination and Collaboration, San Francisco, California, USA, February 22-25, 1999, New York, NY, USA: ACM Press (1999) pp. 19-28
- [4] Casati, Fabio, Maria Grazia Fugini, Isabelle Mirbel, and Barbara Pernici: WIRES: A Methodology for Developing Workflow Applications. Requirements Engineering, Vol. 7, No. 2 (2002) pp. 73-106
- [5] Krishnan, Rupa, Lalitha Munaga, and Kamalakar Karlapalem: XDoC-WFMS: A Framework for Document Centric Workflow Management System. In H. Arisawa, Y. Kambayashi, V. Kumar, H. Mayr, and I. Hunt (eds): Lecture Notes in Computer Science 2465, Berlin: Springer (2002) pp. 348-362
- [6] Ellis, C. A. and S. J. Gibbs: Concurrency Control in Groupware Systems. SIGMOD Record, Vol. 18, No. 2 (1989) pp. 399-407
- [7] Suleiman, Maher, Michele Cart, and Jean Ferrie: Concurrent Operations in a Distributed and Mobile Collaborative Environment, Proceedings of the International Conference on Data Engineering (ICDE'98) Orlando, FL, USA, February 23-27, 1998, Los Alamitos, CA, USA: IEEE Computer Society (1998) pp. 36-45
- [8] Cobena, Gregory, Serge Abiteboul and Amelie Marian: Detecting Changes in XML Documents, Proceedings of the 18th International Conference on Data Engineering (ICDE'02), February 26-March 1, 2002, San Jose, California, USA: IEEE Computer Society (2002) pp. 41-52

-
- [9] Ionescu, Mihail and Ivan Marsic: Tree-based Concurrency Control in Distributed Groupware. *Computer Supported Cooperative Work*, Vol. 12, No. 3 (2003) pp. 329-350
- [10] Sun, David, Steven Xia, Chengzheng Sun, and David Chen: Operational Transformation for Collaborative Word Processing: Proceedings of the 2004 ACM conference on Computer supported cooperative Work (CSCW'04) Chicago, Illinois, USA, November 6-10, 2004, New York, NY, USA: ACM Press (2004) pp. 437-446
- [11] Ignat, Claudia-Lavinia and Moira C. Norrie: Flexible Collaboration over XML Documents. In Y. Luo (ed): *Lecture Notes in Computer Science*, 4101, Cooperative Design, Visualization, and Engineering, Berlin: Springer (2006) pp. 267-274
- [12] Ignat, Claudia-Lavinia and Moira C. Norrie: Multi-level Editing of Hierarchical Documents, *Computer Supported Cooperative Work* Vol. 17, No. 5, 6 (2008) pp. 423-468
- [13] Lee, Heeseok, and Woojong Suh: A Workflow-based Methodology for Developing Hypermedia Information Systems. *Journal of Organizational Computing and Electronic Commerce*, Vol. 11, No. 2 (2001) pp. 77-106
- [14] Aversano, Lerina, Gerardo Canfora, Andrea De Lucia, and Pierpaolo Gallucci: Integrating Document and Workflow Management Systems: Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments, Stresa, Italy, September 5-7, 2001, Los Alamitos, CA, USA: IEEE Computer Society (2001) pp. 328-329
- [15] Aversano, Lerina, Gerardo Canfora, Andrea De Lucia, and Pierpaolo Gallucci: Integrating Document and Workflow Management Tools using XML and Web Technologies: a Case Study: Proceedings of the Sixth European Conference on Software Maintenance and Reengineering, Budapest, Hungary, March 11-13, 2002, Los Alamitos, CA, USA: IEEE Computer Society (2002) pp. 24-33
- [16] Wang, Jianrui, and Akhil Kumar: A Framework for Document-driven Workflow Systems. In W. van der Aalst, B. Benatallah, F. Casati, and F. Curbera (eds): *Business Process Management - Lecture Notes in Computer Science* 3649, Berlin: Springer (2005) pp. 285-301
- [17] Velez, Ivan P., and Bienvenido Velez: Lynx: An Open Architecture for Catalyzing the Deployment of Interactive Digital Government Workflow-Based Systems: Proceedings of the International Conference on Digital Government Research, San Diego, California, USA, May 21-24, 2006, New York, NY, USA: ACM Press (2006) pp. 309-318
- [18] Mohammad Rahaman, Yves Ashiqur Roudier, and Andreas Schaad (2009) *Document-Based Dynamic Workflows: Towards Flexible and Statefull*

- Services: Proceedings of the World Conference on Services - II, Bangalore, India, September 21-25, 2009, Los Alamitos, CA, USA: IEEE Computer Society, pp. 87-94
- [19] Bing, Han, and Xia Dan-Mei: Research and Design of Document Flow Model Based on JBPM Workflow Engine: Proceedings of the International Forum on Computer Science-Technology and Applications, Chongqing, China, December 25-27, 2009, Los Alamitos, CA, USA: IEEE Computer Society (2009) pp. 336-339
- [20] Berliner, Brian: CVS II: Parallelizing Software Development. Proceedings of the Winter 1990 USENIX Conference, Washington, District of Columbia, USA, January 1990, Usenix Association (1990) pp. 341-352
- [21] Collins-Sussman, Ben, Brian W. Fitzpatrick, and C. Michael Pilato: Version Control with Subversion. Cambridge, Massachusetts, USA: O'Reilly (2004)
- [22] Lippe, Ernst and Norbert van Oosterom. Operation-Based Merging. Proceedings of the fifth ACM SIGSOFT symposium on Software development environments, Tyson's Corner, Virginia, USA, December 9-11, 1992, New York, NY, USA: ACM Press (1992) pp. 78-87
- [23] Sun, Chengzheng, Xiaohua Jia, and Yanchun Zhang, Yun Yang, and David Chen: Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. ACM Transactions on Computer-Human Interaction, Vol. 5, No. 1 (1998) pp. 63-108
- [24] Sun, Chengzheng and Clarence Ellis: Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements: Proceedings of ACM CSCW'98 Conference on Computer-supported Cooperative Work (CSCW'98) Seattle, WA, USA, November 14-18, 1998, New York, NY, USA: ACM Press (1998) pp. 59-68
- [25] Suleiman, Maher, Michele Cart, and Jean Ferrie: Serialization of Concurrent Operations in a Distributed Collaborative Environment: Proceedings of the international ACM SIGGROUP conference on Supporting group work (GROUP '97) Phoenix, Arizona, USA, November 16-19, 1997, New York, NY, USA: ACM Press (1997) pp. 435-445
- [26] Li, Du and Rui Li: Preserving Operation Effects Relation in Group Editors. Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04), Chicago, Illinois, USA, November 6 to November 10 2004. New York, NY, USA: ACM Press (2004) pp. 457-466
- [27] Li, Rui and Du Li: Commutativity-based Concurrency Control in Groupware. Proceedings of the IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing

- (CollaborateCom'05), San Jose, California, USA, December 19-22, 2005, Los Alamitos, CA, USA: IEEE Computer Society (2005) pp. 1-10
- [28] Ignat, Claudia-Lavinia and Gerald Oster: Peer-to-peer Collaboration over XML Documents. In Y. Luo (ed): Lecture Notes in Computer Science, 5220, Cooperative Design, Visualization, and Engineering, Berlin: Springer (2008) pp. 66-73
- [29] Assmann, Uwe. Architectural Styles for Active Documents. Science of Computer Programming, Vol. 56 (2005) pp. 79-98
- [30] Duhl, Joshua, and Susan Feldman: Industry Developments and Models, Active Documents: Changing How the Enterprise Works, IDC, Framingham, MA, USA (2003)
- [31] United States Patent and Trademark Office (USPTO) (2004) Office of the Chief Information Officer, Operational Information Technology Plan, FY 2005 – FY2006, USPTO, Department of Commerce, Washington, DC, USA
- [32] Leng, Gillian, Nicola Bent, and Ian Saunders: Electronic Guidance Access Project (EGAP) (Interim report V1.0) National Institute for Health and Clinical Excellence, London, UK (2008)
- [33] Teohari, Mihai, and Copcea Larisa: XML Authoring Tool: Proceedings of the 4th Conference on European Computing, Bucharest, Romania, April 20-22, 2010, New York, NY, USA: ACM Press (2010) pp. 143-147
- [34] Yang, Haijun: Design and Implementation of Electronic Medical Record Template Based on XML Schema: Proceedings of the Second WRI World Congress on Software Engineering, Wuhan, Hubei, China, December 19-20, 2010, Los Alamitos, CA, USA: IEEE Computer Society (2010) pp. 225-118
- [35] Gostojić, Stevan, Goran Sladić, Branko Milosavljević, and Zora Konjović: Context-Sensitive Access Control Model for Government Services, Journal of Organizational Computing and Electronic Commerce, Vol. 22, No. 2, (2012) pp. 184-213
- [36] Sladić, Goran, Branko Milosavljević, Zora Konjović, and Milan Vidaković: Access Control Framework for XML Document Collections. Computer Science and Information System (ComSIS) Vol. 8, No. 3 (2011) pp. 591-609
- [37] Sladić, Goran, Branko Milosavljević, Dušan Surla, and Zora Konjović: Flexible Access Control Framework for MARC Records. The Electronic Library, Vol. 30, No. 5 (2012) pp. 623-652
- [38] Sladić, Goran, Branko Milosavljević, and Zora Konjović: Context-Sensitive Access Control Model for Business Processes: Computer Science and Information System (ComSIS) Vol. 10, No. 3 (2013) pp. 939-972