A new Model for Document Management in e-Government Systems Based on Hierarchical Process Folders

Raphael Kunis, Gudula Rünger and Michael Schwind
Chemnitz University of Technology, Chemnitz, Germany
krap@informatik.tu-chemnitz.de
ruenger@informatik.tu-chemnitz.de
schwi@informatik.tu-chemnitz.de

Abstract: Document management plays a decisive role in modern e-government applications. As today's authorities have to face the challenge of increasing the efficiency and quality while decreasing the duration of their government processes a flexible, adaptable document management system is needed for large e-government applications. In this paper we introduce a new approach for a document management model that helps to face this challenge. The model is based on two new document management concepts that extend common document management facilities: hierarchical process folders and document security levels. A hierarchical process folder mainly consists of files that belong to a government process and include all documents processed during process execution. The folder grows during execution and contains all versions of changed, existing, and added documents. The process folders can be used in a single authority software system as well as in distributed e-government software systems. More precisely, this means that the model of hierarchical process folders can be deployed to exchange process folders in whole or in part between authorities to support the execution of distributed hierarchical government processes. We give an example how the application to single authorities and distributed systems is possible by describing the implementation within our distributed e-government software system. The application of security levels to documents allows the encryption of documents based on security relevant properties, e. g. user privileges for intra authority security and network classification for inter authority communication. The benefits of our model are at first a centralized data management for all documents of a single or a hierarchical government process. Secondly, a traceable history of all data within government processes, which is very important for the archival storage of the electronic government processes, is provided. Thirdly, the security levels allow a secure intra authority document accessing system and inter authority document communication system.

Keywords: Electronic government applications, document management systems, hierarchical government processes, interoperability, document processing, e-Government security

1. Introduction

An important objective in the modernization process of authorities is the application of e-government solutions. These solutions help to support the communication process between citizens and authorities on the one hand and increase the efficiency of internal government processes on the other hand. In this paper we deal with the internal process execution. The execution of large government processes involving many employees depends highly on the treatment and transfer of documents. To reach the goal of increasing the efficiency by reducing delay time a flexible document management system (DMS) is needed. However most existing document management systems are not designed to fulfil the special requirements of e-government applications. Within the scope of our research project, we faced the challenge of embedding a *DMS* into the overall software system architecture and adapt this *DMS* to the special needs of authorities. These needs include an easy integration of the *DMS* into the existing information technology (IT) infrastructure, the auditable safe archival storage of the documents and an easy adaptation to the utilized software system used for the execution of hierarchical government processes.

In this paper we introduce a new approach for a document management model that fits the needs mentioned above. The model is based on two new concepts that extend common document management facilities: hierarchical process folders and document security levels (*slevel*). Hierarchical process folders deal with the storage of all versions of the documents of a process in a bundled manner. This is a necessary and advantageous approach enabling the inspection of all processed documents at each point of the process execution. The hierarchical aspect is reflected in separate process folders of all processes in a process hierarchy. Merge points at the start and

completion times of subprocesses guarantee a fluent cooperation between processes within a single authority and between authorities.

Security is a crucial point in the field of distributed process execution. Integrity, confidentiality, authenticity and traceability have to be guaranteed for each transferred document. The proposed solution of application of *slevel* to documents allows the encryption of documents based on security relevant properties, e. g. user privileges for intra authority security and network classification for inter authority communication. There are other European and German projects in the field of e-government. The "Document Management and Electronic Archiving in Electronic Courses of Business" (*DOMEA*) concept (KBsT 2005) is the basis of the German Government for achieving the aim of reaching the paperless office. DOMEA introduces particularly the concepts and criteria that should lead to paperless offices in the administrations. The three-pieced modular structure of DOMEA consists of documents, records and files. In contrast to our solution the issues of hierarchical process execution and security in the distributed process execution is not addressed.

Within the "standards and architectures for e-government" (*SAGA*) (KBsT 2005), defined by the German Federal Ministry of the Interior, regulations on standards, procedures, methods and also products for the modern IT-evolution are provided. Standards are divided into different categories (mandatory, recommended, under observation and rejected). SAGA also proposes applications like the "Government Site Builder". This application is a content management system whose *DMS* component offers versioning on change, write locks and the possibility to use meta-data for documents. This functionality is similar to that of our *DMS* implementation. However, the "Government Site Builder" was not designed for the work with cross-authority process execution.

Throughout the European Union the "Model Requirements for the Management of Electronic" Records (*MoReq*) (Office for Official Publications of the European Communities 2002) project defines the basic requirements for document management. The provided requirements list is very detailed and based on the ISO 15489 standard (ISO 2001). Beside requirements for *DMS*s MoReq defines criteria for document functions, e. g. workflows, email and electronic signatures. The specification solely specifies the requirements and provides no implemented solution as we do. This short summary shows that other projects also deal with document processing in e-government. However none of the presented efforts reaches the possibilities provided by us, especially in the case of distributed processing. The benefits of our model are at first a centralized data management for all documents of a single or hierarchical government process. Secondly, a traceable history of all data within government processes, which is very important for the archival storage of the electronic government processes, is provided. Thirdly, the *slevel* allow a secure intra authority document accessing system and inter authority document communication system. Because we also provide an implementation of our model we can show that the application to authorities is guaranteed. The paper is structured as follows: Section 2 gives an overview of *DMS*s. In Section 3 the concept of hierarchical process folders is introduced. Section 4 deals with the concept of security levels. In Section 5 we discuss the cooperation of the models in the distributed process execution while Section 6 concludes the paper.

2. Document management systems

*DMS*s are used for the creation, capturing, modification, storage and the propagation of electronic documents. Many *DMS*s exist that are distinct in their base as well as extra functionality.

2.1 Importance of open source software in document management systems

It is generally agreed that software in the sector of public administration is in use over a very long time period. Thus this software has to compete with changes of the underlying hard- and software. If *DMS*s utilize closed source software, there is a high risk that support is discontinued because of company acquisition, insolvency or strategic decisions. The support problem can be solved using software with public interfaces. It has to be possible to migrate to different software implementing the public interfaces. Migration to new software is in general expensive or even impossible if large data sets grown over years are considered. There are open standards for *DMS*s, e. g. ODMA (DMWare 1997), DMA (DMWare 2002) and WebDAV/DeltaV (Goland 1999). Using such standards it is possible to exchange the *DMS* because they use public interfaces. An advantage using open

source software is, that this software has a large community which develops/adapts the software components over time. Even if the software support is discontinued developers can adapt the software because they have access to the source code. Open source software also has the advantage to be free of charge whereas proprietary software is in general very expensive and support requires additional payments.

2.2 Document management system protocols

The protocols DMA, ODMA and WebDAV have proved of value in *DMS* software. Both DMA and ODMA are interfaces depending on OLE and DCOM and are therefore only useable on Windows platforms. Through this restriction, we use WebDAV as an interface between our system and the *DMS*. WebDAV is an extension to the Hypertext Transfer Protocol 1.1 (HTTP/1.1) (Network Working Group 1999). It adds additional header data and methods to the protocol which enhance the support for storing and managing data. These enhancements are *Collections*, *Locking*, *Properties* and *Copy/Move*. The original WebDAV standard had no support for versioning files and collections, but there is an enhancement called WebDAV/DeltaV (Network Working Group 2002). WebDAV/DeltaV enables the storing of different versions of data which can be merged together (checkin/checkout). A simple mechanism for versioning data is the DeltaV auto versioning. With auto versioning every modification of data or meta data results in a new version. This functionality is sufficient for our software system to store different revisions of documents. WebDAV uses authentication and authorization methods provided by HTTP/1.1. Other inherited properties from HTTP/1.1 are the support for encrypted communication, use of proxies and caching of data. Because of the popularity of HTTP, WebDAV is easily usable within the infrastructure of large authorities. Especially there are no problems using WebDAV with protection systems like firewalls.

2.3 Evaluation of open source document management systems

The next paragraph compares two different *DMS*s in terms of their usage for our software system. The open source *DMS*s are Jakarta Slide and Subversion with its WebDAV module for the Apache web server.

2.3.1 Jakarta slide

Jakarta Slide is a low level content management framework, which can be used by developers to implement their own *DMS*. It is open source software developed under an Apache license. Jakarta Slide has good support for WebDAV and its enhancements like DeltaV and DASL (DAV Searching and Locating). It provides a Client-API that enables the easy integration into own projects and it is very flexible in its storage system integration. Normal file systems or database systems can be used. The WebDAV/DeltaV interface to Jakarta Slide is implemented as an Apache Tomcat-Servlet-Container and provides easy integration of the Lightweight Directory Access Protocol (LDAP) authorization system we use in our software system. Jakarta Slide is integrated into the Apache Jakarta project and therefore long time supported is guaranteed.

2.3.2 Subversion

Subversion is a version-management-system like CVS (concurrent versioning system) developed under an open source licence. It can be combined with WebDAV/DeltaV as interface protocol. By utilizing an Apache web server with subversion module WebDAV can be used as an interface protocol. The Apache web server also offers the ability to integrate LDAP based user management. As in Jakarta Slide the storage of data is very flexible. File systems or database systems can be used. Subversion offers support for storing just the modifications between different versions for binary and text data. Subversion has an increasing popularity over the last few years and is used to manage large software repositories in open source projects. Therefore subversion will be supported and actively enhanced over a long period of time.

2.3.3 Selection of the system that fits the needs in e-government

Large administrations store a large amount of data, e. g. PDF-forms and GIS (geographical information system) data. These data are often changed during process execution resulting in many different data versions. Because of its efficient handling of versioned data we chose

subversion as low level *DMS* component of our system. But our software design and the use of open interfaces also enables the easy adoption to Jakarta Slide.

3. The concept of hierarchical process folders

Our software system supports authorities by executing their government processes electronically. The processes are modelled as hierarchical workflows and executed by an underlying workflow management system. A workflow is a government process that is modelled in a form that allows a computer system to execute it. A workflow that is in execution is in the following called a process. The following definitions are based on the specifications by the workflow management coalition (Hollingsworth 1995, Norin 2002). Workflows consist of activities that belong to actions that are processed by the computer system (automated activity) or a user (manual activity). An automated activity can be a workflow itself. By this model hierarchical workflows can be described by defining the individual parts of the workflows as independent workflows and combining them by using automated activities. Because we design our government processes as workflows following this specification we are able to map large processes consisting of many small subprocesses to workflows that our system can execute.

3.1 Hierarchical processes

In contrast to small processes consisting of only a few activities we deal with large processes that are hierarchically designed. This hierarchical design is necessary because parts of the processes are executed on external systems. The hierarchical design enables a flexible modelling approach. Large processes only need to define the interfaces for their smaller components. As an example we give the process of the *official approval of building plans* in **Figure 1**. In our context the terms parent process and child process are used to indicate the relationship between two processes in a hierarchical order. The overall process is called the parent process and any of the subprocesses is called child process. Parent processes may pass values to the children. The mechanism how documents are transferred from a parent process to a child process and backwards is described later. In the Figure the parent process of all other processes is the *official approval of a building plan* (P1*)*. This process has one subprocess called *execute the official approval of a building plan*. This process is child of P1 and also parent of five processes (*handle claims*, ...). This structure shows the hierarchcal structure of the government processes we deal with and will be used for the examples in the rest of the paper.

3.2 Implementation of the document management system

We use a combination of an Apache web server and a subversion repository as motivated in Section 2.3.3. The core components of the *DMS* can be seen in Figure **2**. The system can be accessed via HTTP/1.1. The web server utilizes a subversion module to communicate with the subversion repository. If a user connects via the apache server to the repository at first his rights are checked against the LDAP server. The LDAP server stores the user information and rights. If access is granted the user can read the specified document. If a user requests to write a document to the repository the access check is done in a similar way. Any write operation of a document results in a new version of the document. This version includes all changes the user did and information on the user. If automated activity operations are performed on a document (e. g. a document was written back after changing on another system) a special system user is utilized and handled in the same way as normal users.

3.3 The configuration of the repository

The structure of the repository consists of three primary directories. The configuration of the repositiory can be seen in Figure 3. In the "*template directory"* templates for documents, graphical process trees and webforms of the loaded workflows are stored. A graphical process tree is utilized to show the actual process execution location. Webforms are needed to output the activity variables and documents a user can change in the graphical user interface. The "*process folder directory*" is subdivided into directories representing the process folders of the actual executed processes. In the "*archive directory"* the parts of the process folders that have to be archived for later reference are stored after the completion of a process.

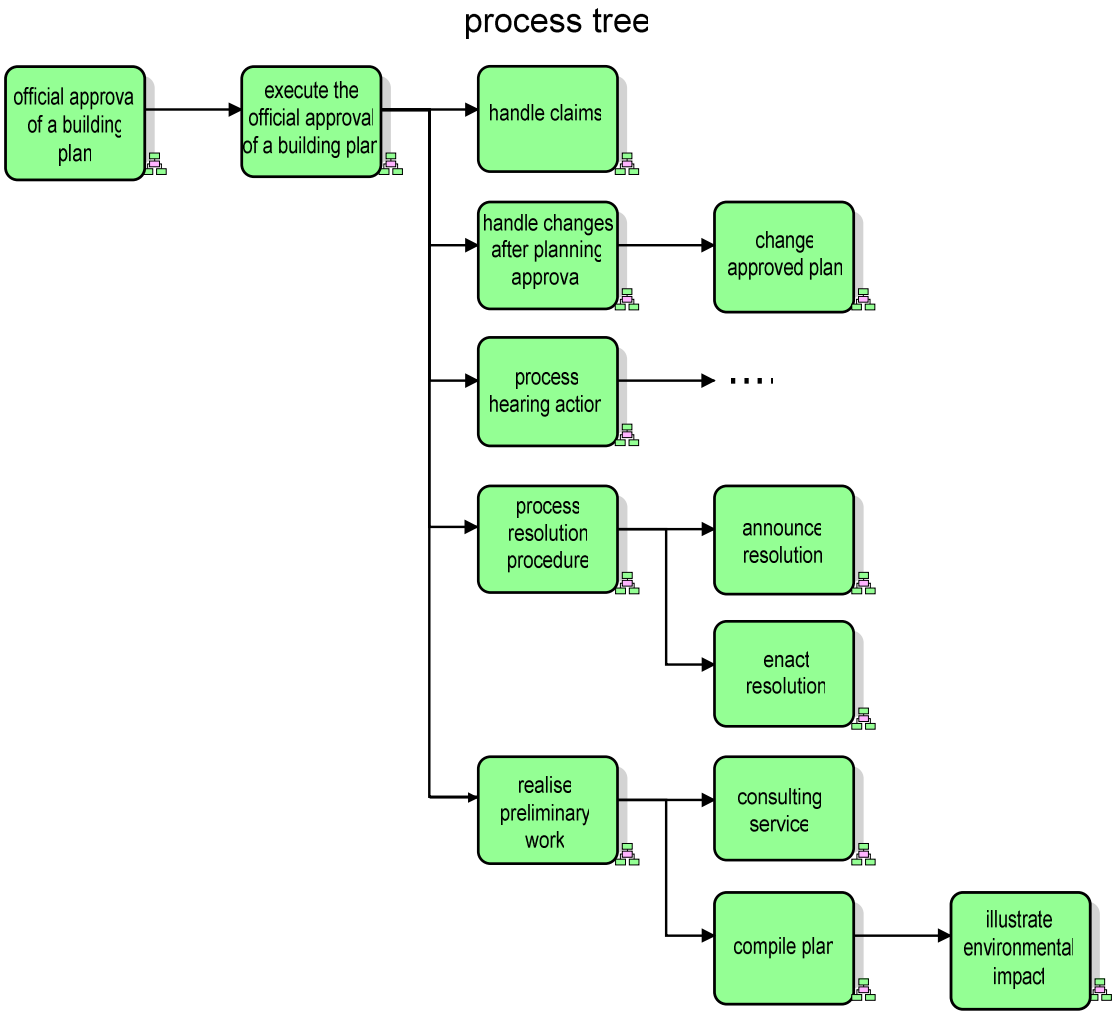

Figure 1: Parts of our example procedure

The official approval of building plans. The green rectangles show the overall process (upper left corner) and some of the subprocesses. The arrows are used to indicate the parent – child relationships, the source of the arrow is the parent and the target is the child process.

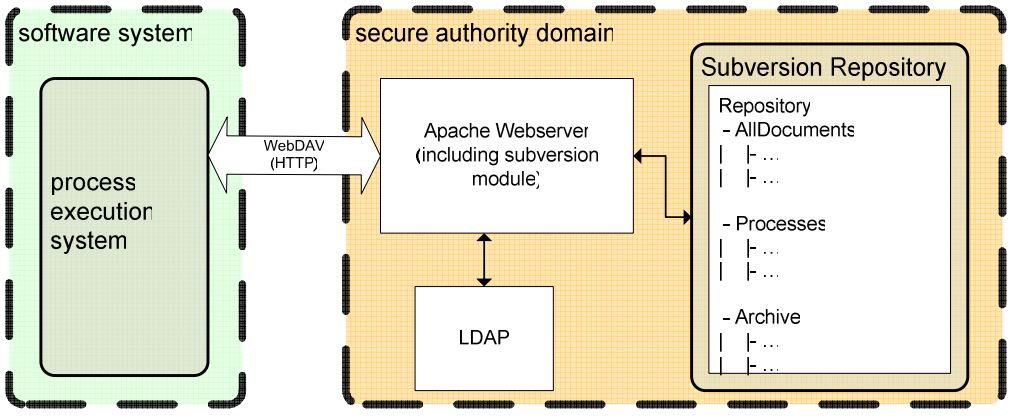

Figure 2: Overview of the implemented and utilized dms components.

The dms server consists of three components, an Apache web server with subversion module, a subversion system and an LDAP server.

Figure 3: Configuration of the repository. The three main directories "AllDocuments", "Archive" and "Processes" are shown.

3.3.1 The template directory "AllDocuments"

In this directory all input documents of the processes are stored. There are process independent and process bound documents. This directory contains at least the subdirectory "*docs*" where common documents are stored. Additionally there is a directory for any process package that is loaded into the system. A process package contains one or more workflow definitions of government processes that are logically connected. A hierarchical modelled process can be stored with all subprocesses in a single package. The advantage of this approach is the possibility to define global variables that can be accessed by all process parts. The package directories contain all documents that are relevant for the execution of the workflows contained in the corresponding package. In the Figure an example is given as package "*pfv*" that contains all workflows belonging to the official *approval of building plans*. The directories created for any package are:

- Directory "docs": This directory contains all predefined forms and documents of the workflows in the package. The documents in this directory can coincide with documents in the global "docs" directory. The document that should be used later is configured by an administrator when the package is loaded into the system.
- Directory "svg": This directory contains the graphical representation of the processes in the package. An example how this presentation looks like was given **Figure 1**. The representation is utilized to show the progress of the processes during their execution.
- Directory "webforms": The third directory contains the webforms of all manual activities in the workflows of the package needed for the graphical output and interaction of users with the system. A webform is the description of the modelled variables and their modification possibilities, e. g. read only, read and write, the documents belonging to the activity and in some cases also texts of law that belong to an activity.

Template documents must be added manually because they contained in a process package. If a process package is imported or changed a mapping file is created. This mapping file maps generic names for documents that are specified in the workflow definitions of a package to existing documents stored in the template directory. An example mapping file is given in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
  <files>
    <map realname="form_claim1.pdf"
         refname ="form_claim1"/>
    <map realname="form_hearing.txt"
         refname ="form_hearing"/>
    <map realname="form_cp2006.pdf"
         refname ="form_concerned_persons"/>
    <map realname="p_announcement2.pdf"
         refname ="prototype_announcement"/>
  </files>
  <directories/>
</mapping>
```

Figure 4: Example of a mapping file that is used to map generic document names given in workflow definitions of a package to existing documents in the template directory.

3.3.2 The hierarchical process folder directory "Processes"

This directory contains the process folders of all running processes. Each process has its own process folder that is created at the starting time of the process and is named after the process identifier within the system. **Figure 3** shows three process folders "201_pfv_12", "202_pfv_1" and "203_pfv_16". Any process folder consists of two subdirectories. The subdirectory "*docs*" is used for storing the documents of the process. These are the documents that were copied at the starting time of the process and all other documents created during process execution. The subdirectory "*memos*" is used to store memos created by users of the system to notify other users about events that are connected to the further execution of the process. At the starting time of a process the document templates are copied from the template directory. The copy procedure is based on the mapping file. The case of hierarchical processes needs special treatment. In the case of starting a subprocess of another process the copying of documents from the template directory is replaced by the hierarchical search for documents in the parent process hierarchy. This is necessary because it is possible that in the subprocess documents form the parent process are needed and the previously in the parent process inserted data needs to be available. Therefore in the case of a subprocess the copy procedure searches for the documents of the subprocess in the process folder of the direct parent process. Direct means that this parent process is the one containing the subprocess. If documents are found in this process folder they are copied and the algorithm finishes. All documents that were not found are searched for in the process folder of the direct parents parent process folder if existing. If a document was not found it is copied from the template directory. When a process finishes its execution this procedure is done in the reverse direction. All documents of the process are copied back to the correct process folder. This copy is a new version of the document in the parents process folder.

3.3.3 The archive directory "Archive"

Data of all finished processes are transferred to the archive directory, because in the data have to be stored for subsequent inspection. Instead of using the process identifier as in the process folder now the reference number of the process is used as a directory name. In **Figure 3** two completed processes exist "A01_2006" and "A02_2006". The archive contains the document directory "docs" and the memo directory "memos".

4. The concept of security levels

Our software system enables the distributed execution of government processes. Most of the operations within an authority involve external institutions in the execution of their government processes. In the case of our example process these institutions are other authorities, agencies of official concerns or companies. When processes are executed in a distributed manner it is necessary to exchange information (e. g. documents) between the involved institutions. In most cases different networks connect them requiring different security arrangements. Because the security arrangements are needed to determine what level of security is necessary to transfer data over the networks we partition them in three levels. Level one means that a network is highly

trustable (e. g. the intranet of an authority), level two means medium trustable (e. g. a network between two authorities that is managed by a governmental agency) and level three means not trustable (e. g. the internet). According to this level and the level of the document that is introduced in the next paragraph the transfer security mechanism is chosen. Document security levels (*slevel*) are implemented by adding additional information to the documents stored in the process folders. Each document is annotated an *slevel*. This level lies between one (low security needs) and three (high security needs) and is assigned by an administrator when adding the document to the template directory. The values for the *slevel* depend on four security aspects: confidentiality, integrity, authenticity and traceability.

- Confidentiality: no one else except the sender and receiver of the document should be able to read it during the transportation.
- Integrity: protection of documents against unauthorized modification.
- Authenticity: covers the assurance that the sender sent the document and the receiver received the document.
- Traceability: it can be proofed who sent a document and who received it.

Table 1: Mapping of document security level times network security level to the utilized security mechanism for the security aspects.

security aspect	document security level	Security mechanism		
		network slevel = highly trustable	network slevel = medium trustable	network slevel = not trustable
confidentiality	Low (1)	none	none	encryption
	medium (2)	none	encryption	encryption
	high (3)	encryption	encryption	encryption
integrity	low (1)	none	none	advanced signature
	medium (2)	none	advanced signature	qualified signature
	high (3)	qualified signature	qualified signature	qualified signature
authenticity	low (1)	none	none	check for valid certificate
	medium (2)	none	check for valid certificate	challenge-response method
	high (3)	challenge-response method	challenge-response method	challenge-response method
traceability	low (1)	none	none	acknowledgement
	medium (2)	none	acknowledgement	signed acknowledgement
	high (3)	signed acknowledgement	signed acknowledgement	signed acknowledgement

Table 1 shows the mapping of document and network *slevel* to the needed security mechanism for transportation. No security mechanism means that the document is transferred without any security mechanism. If a document is encrypted, a cryptography algorithm is used before sending the document over the network. Possible encryption algorithms are Data Encryption Standard (DES) (FIPS 1999) and Advanced Encryption Standard (AES) (FIPS 2001) that belong to the class of symmetric algorithms and Rivest-Shamir-Adleman (RSA) (Rivest 1977) and Elgamal (ElGamal 1984) that belong to the class of asymmetric algorithms. Asymmetric algorithms (public key systems) can also be used for the digital signature of documents. Public key systems use two keys to encrypt data, a private and a public key. The private key is used to decrypt a message that was encrypted with the corresponding public key. A digital signature is created by using the private key of the sender and is directly dependent on the contents of a message. The verification of the message can be done using the public key of the sender. The terms advanced signature and qualified signature are defined in (German Federal Ministry of Justice 22. Mai 2001). The challenge-response method guarantees that the sender of a message is in possession of the private key that is assigned to the public key of the utilized certificate. A certificate is a digital signed document that defines the binding between a public key and a certain name/system. Certificates are created by a certification authority. Acknowledgement means that the receiver has to confirm that he received the data from the sender. If a signed acknowledgement is necessary, the receiver has to sign the acknowledgement with his private key. Our transport mechanism is

based on OSCI Transport (OSCI-Leitstelle 2002). OSCI-Transport enables the secure and traceable communication between two software systems.

5. Cooperation of the models in the distributed execution of government processes

The external execution of government processes primarily deals with the automated transfer of documents between the involved authorities. The documents have to be sent from the software system executing the parent process to the external authority. After the execution of the child process the documents have to be reintegrated into the parent process. Our system is not limited to the exchange of documents that exist in the parent process but also enables the integration of newly created documents. An example can be seen in Figure 5. This Figure shows two software systems that execute the government processes modelled as workflows. The example process is the *official approval of building plans* that was introduced earlier. Software system I (*S1*) is the system executing the parent process (*execute the official approval of a building plan, P*) while software system II (*S2*) executes the child process (*process resolution process, C*). This example is abstract because typically the transfer network is the same. However, this example aims to show the potentialities of our system.

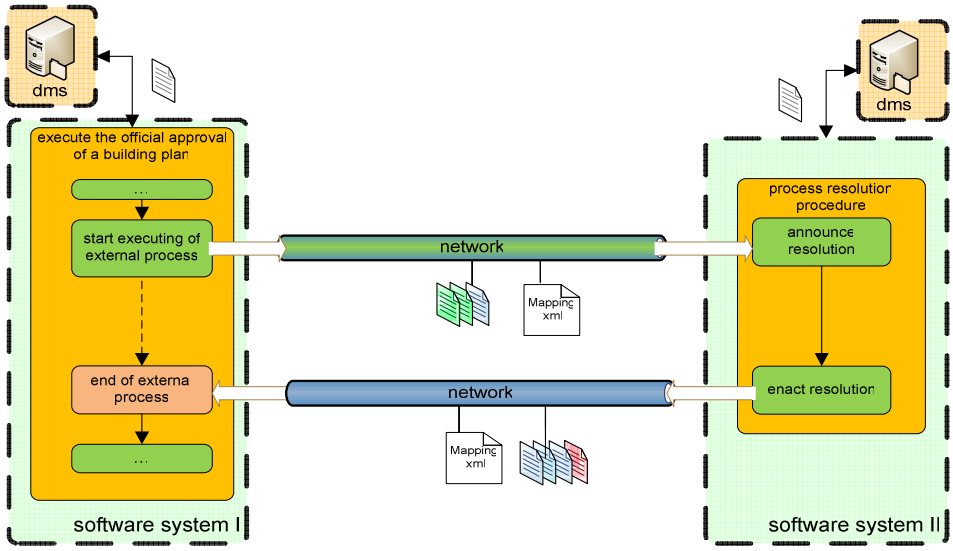

Figure 5: Example of the external execution.

The involved systems, the document management systems, the data that is transferred and the involved networks are shown. The green (upper) network (parent to child) is a highly trustable network meaning that the security level of the documents is the security level for the transfer. The blue (lower) network (child to parent) is medium trustable. Because a medium network indicates that the security level has to be at least medium the documents with a small security level have to be adopted for transfer.

The example starts with the execution of an activity in process *P* that includes an external subprocess *C*. The single steps are shown in Table 2. *S2* is waiting for requests to start the execution of process *C*. At first a request for the execution of *C* is sent from *S1* to *S2*. *S2* starts the process, creates its process folder and waits for the needed documents to be transferred. Afterwards *P* gathers information on the documents that have to be transferred to the system *S2* and the *slevel* of these documents. This information and the information of the green network in the Figure are utilized to determine the security mechanism needed for the documents. The determination is done with the data given in Table 1. Afterwards the documents are transferred. In **Figure 5** these are two documents with *slevel* one (green) and one with *slevel* two (blue). Because the medium is highly trustable, the security mechanism is mainly affected by the document *slevel*. In addition to the documents the needed mapping file parts are also transferred. S2 copies the received documents and the mapping file to the process folder of the earlier started process *C*. This process is executed and system *S1* waits for the completion of *C*. The documents can change

during the processing of *C* and new documents can be added to the process folder of *C*. When the process is completed *S2* notifies *S1* and *S1* waits for the documents to be sent. Therefore the information on the documents as well as the *slevel* are gathered and the documents are sent to *S1*. In the example in the Figure we have to deal with a medium secure network indicating that the transfer *slevel* is at least medium. It can be seen that the earlier sent green documents have now the colour blue meaning medium transfer *slevel*. The blue document from the start of the process remains blue. Additionally a new document with an *slevel* of three (red) is sent back to process *P*. *P* stores the documents and the adopted mapping file to its process folder and proceeds with its execution.

Table 2: Steps done by the systems during start and end of an external child process.

Parent process executing system (S1)	Child process executing system (S2)
0. request the external execution of the child process	0a. create and initialise the child process 0b. create the hierarchical process folder
1: gather information on documents that have to be sent to the child system	Wait for documents
2: read the documents and their slevel from the document management system	
3: read information on the network used for the transfer of the documents	
4. calculate the slevel of any of the documents depending on steps 2 and 3 = transfer security level	
5. transfer the documents according to the calculated transfer security level	1. receive the documents
Wait for the child process to complete	2. store the documents in the hierarchical process folder of the child process
	3. execute the child process … 3.1 process finishes
	4. gather information on documents that have to be resent to the parent system
	5. read the documents and their slevel from the document management system
	6. read information on the network used for the transfer of the documents
	7. calculate the slevel of any of the documents depending on steps 5 and 6 = transfer security level
6. receive return data and the return documents	8. signal the completion of the child process and transfer return values and the documents according to the calculated transfer security level
7. store the documents in the process folder of the parent process	
8. continue execution	

6. Conclusion

Document management is very important in e-government for reaching the goal of paperless offices. The management of documents using our model of hierarchical process folders is tailored to the execution of government processes. Within a process folder all documents of a process are stored and every change of a document results in a new version. This enables a traceable version history of all process data. Therefore our model is comparable to a changing file containing all information of a government process with the addition that every version can be restored. The included archival storage mechanism facilitates later inspection of all completed processes. Our model of security levels provides security mechanism that are tailored to the processing of distributed government processes within and between authorities.

References

DMWare (1997) “Open Document Management API (ODMA) specification”, version 2.0., [online], http://odma.info/

DMWare (2002) “Document Management Alliance (DMA) specification 1.0-7”, [online], http://dmatech.info/

ElGamal, Taher A. (1984) “Cryptography and logarithms over finite fields”, Stanford University

FIPS (1999) “Data Encryption Standard”, Federal Information Processing Standards Publication, No. 46-3

FIPS (2001) “Announcing the Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication, No. 197

German Federal Ministry of Justice (2001): Gesetz über Rahmenbedingungen für elektronische Signaturen. SigG. 2001, [online], http://www.gesetze-im-internet.de/sigg_2001/index.html

Goland, Y. et al. (1999) "RFC 2518. HTTP Extensions for Distributed Authoring - WEBDAV", [online], ftp://ftp.rfc-editor.org/in-notes/rfc2518.txt

Hollingsworth, David (1995): The workflow reference model. Issue 1.1, Winchester: Workflow Management Coalition.

ISO (2001): Information and Documentation - Records management. Vol. 1. Part 2: Guidelines (ISO 15489-2).

KBsT - Federal Government Co-ordination and Advisory Agency (2005) “SAGA - Standards and Architectures for eGoverment-Applications”, version 2.1, Schriftenreihe der KBSt 82

KBsT - Federal Government Co-ordination and Advisory Agency (2005) “DOMEA concept. Document Management and Electronic Archiving in Electronic Courses of Business.” , Organisational Concept 2.0, Schriftenreihe der KBSt 74.

Mendling, Jan/Nüttgens, Markus (2006): "EPC markup language (EPML). An XML-based interchange format for event-driven process chains (EPC)". In: Information systems and e-business management 4, No. 3

Network Working Group (1999) “RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1.”, [online], http://www.faqs.org/rfcs/rfc2616.html

Network Working Group (2002) "RFC 3253. Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)", [online], ftp://ftp.rfc-editor.org/in-notes/rfc3253.txt

Norin, Roberta (2002): "Workflow Process Definition Interface - XML Process Definition Language". WfMC, [online], http://www.wfmc.org/standards/docs/TC-1025_xpdl_2_2005-10-03.pdf

Office for Official Publications of the European Communities (2002) ”Model requirements for the management of electronic records. MoReq specification”, Luxembourg

OSCI-Leitstelle (2002): "OSCI-Transport. Specification", [online], http://www1.osci.de/sixcms/media.php/13/osci-specification_1_2_english.pdf

Rivest, R./Shamir, A./Adleman, L. (1977) “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, MIT/LCS/TM-82.