A Component Based Software Architecture for E-Government Applications

Daniel Beer[1], Raphael Kunis[1], Gudula Rünger
Chemnitz University of Technology, Department of Computer Science
09107 Chemnitz, Germany
E-mail: *{dbeer,krap,ruenger}@informatik.tu-chemnitz.de*

Abstract

The raising need for e-government applications leads to many new approaches in this sector. To fulfill the requirement for a flexible government-to-government (G2G) software system being adaptable for the usage in many sectors of e-government applications we introduce the reference architecture for e-government (RAfEG) in this paper. The key features of the system are flexibility, security, adaptability and interoperability between authorities. The efficient usage of heterogeneous systems and heterogeneous hardware platforms, respectively, allows the execution of large interactive applications in e-government. Because security is a critical issue in e-government applications our solution uses different types of authentication and authorization methods and also supports secure communication between the interoperating heterogeneous systems. Due to the fact that the electronically supported execution of government procedures is the main aspect of the RAfEG system, an approach where these procedures are modeled as workflows and executed by an underlying workflow management system (WfMS) is the solution we present in this paper. Although many e-government applications exist at present, the RAfEG system is a new approach because it is able to cope with a wide range of internal official procedures and also highly adaptable to new procedures within e-government.

1. Introduction

The goal of our work is the implementation of a software architecture for the support of the electronic mapping of long running authority internal official procedures. These official procedures are associated with a lot of paperwork and in many cases more than one employee is involved. Additionally many official procedures are not restricted to a single authority, leading to the need for a distributed execution of these. This requires a system that is able to share tasks of such procedures and also task associated documents between authorities.

In this paper, we present a new software architecture based on workflows to fulfill the requirement for a software system that is adaptable to a wide range of official procedures within authorities. By the new approach of integrating workflow processing into e-government systems the efficiency of e-government applications can be increased. Every official procedure that is executed by an authority consists of several tasks that have to be executed by employees of this authority or external authorities and institutions. This is exactly what the term workflow stands for, the electronically supported execution of business processes. So the preferred solution is to model all procedures as workflows and implement a system that can cope with these instead of hard coding special procedures. This approach makes it possible to use the reference architecture for e-government (RAfEG) in all authorities executing procedures that can be modeled as workflows.

The main objectives of RAfEG include the provision of a system that can cope with security standards that are necessary in authorities and mechanisms to allow a distributed execution of workflows involving different workflow management systems (WfMSs). Due to the fact that security is of profound importance in security critical environments the system is flexible in choosing the security mechanisms. The security component can be exchanged to adapt the system to use existing and established mechanisms. This means that the system is able to allow an adjustment to existing security restrictions.

The goals of the RAfEG project are both, the design of a suitable e-government solution based on workflows and the design of a software system according to modern standards for open source software. Specifically, the RAfEG software system offers a component based software architecture and the usage of open source components and formats. This approach enables authorities to exchange all main software components in order to adapt the RAfEG system to their specific needs and requirements. There are

[1] supported by “Bundesministerium für Bildung und Forschung” (BMBF), the German Ministry for Education and Research, project RAfEG (reference architecture for e-government)

other e-government solutions but most of them are commercial products which use proprietary formats. One example is the eGov-Suite by Fabasoft [4] in the ELAK project [1] in Austria. The basic functionality is similar to the RAfEG system, however there are differences concerning flexibility, since the internal design is not visible. In contrast, the component based design of RAfEG allows access to specific components. Thus, the system is adaptable to specific software situations of authorities and allows a stepwise implementation of the system. It also enables the integration of custom components to fit extra requirements, e. g. high security needs.

Our implementation is tested on basis of official approval of building plans within the regional board of Leipzig. An official approval of a building plan consists of several steps modeled within a workflow. These steps include the incoming and preliminary survey of a building plan, the distribution of the plan to all involved authorities and external institutions, the check of the plan by these authorities and institutions and the adaption of the plan with returned suggestions for changes.

Further details on the reference architecture can be found in Section 2. Section 3 deals with issues concerning the interoperability between arbitrary authorities when executing procedures with distributed responsibilities. In Section 4 details on security mechanisms of our implementation are given and Section 5 concludes the paper.

2. The reference architecture for e-government

In this section details on the layout of our system architecture are given. The architecture can be divided into three main parts: the RAfEG core system, the frontend system and the backend system. These parts as well as their components and the deployed protocols are shown in Figure 1.

The RAfEG core system includes methods to gain a secure system through authentication and authorization, to log information of the other three RAfEG core system components, to mediate requests between the presentation component and the workflow management system component and to allow unified access to the document management server. The frontend system deals with interaction by users. This interaction includes the use by employees to work with the system, the use by administrators to monitor and control the system and also by workflow modelers to integrate new procedures modeled as workflows into the RAfEG core system. Furthermore, some nonelectronic communication methods need to interoperate with the system through the frontend system. The backend system contains third party components needed by our system, e. g. the underlying database needed by the WfMS as well as by the RAfEG kernel component.

The RAfEG system is based on several technologies that were chosen for the following reasons:

- the software should be executable on all established operating systems,
- the software should be easy to expand and adapt,
- distribution of system components on heterogeneous platforms as well as the distributed execution of official procedures should be possible,
- the security features should allow a highly secure system by using up-to-date security software solutions and protocols,
- nonelectronic communication should be supported,
- various output formats like HTML pages, PDF documents and WML (Wireless Markup Language) pages should be supported and new output formats should be easy to add,
- the third party software should be free and easily exchangeable and

To reach these goals the following technologies are utilized directly within our system and as third party modules in the backend system. Java is used as programming language, because many operating systems are able to execute Java bytecode. To allow the system to be flexible by exchanging components and to allow a distribution of the RAfEG core components to different servers, we have decided to implement them using the Enterprise JavaBeans technology [3, 11] and to use an application server (AS) to provide them. JBoss AS [7] has been the choice because it is mature, free, open source and Java based. The database management system used by the core system and the WfMS has to be Java Database Connectivity (JDBC) based to support a wide range of relational database systems. To access the document management system the WebDAV (Web-based Distributed Authoring and Versioning) [19] protocol is utilized. We have decided to assemble our own system that can be accessed via this protocol, because existing free document management solutions did not fit our needs. This needs are versioning of documents, access via the WebDAV protocol and authentication, authorization support. Our solution is based on an Apache web server with LDAP (Lightweight Directory Access Protocol) [17] and Subversion (SVN) support.

Due to the fact that the output formats should be easy to expand our decision has been to use an XML based interim representation that is generated by the components after requests, e. g. worklist data by a request to the workflow management system component. This representation contains no layout information. The presentation component of the RAfEG core system transforms this data XML documents to the final output. The underlying WfMS should correspond to the workflow management standards defined and

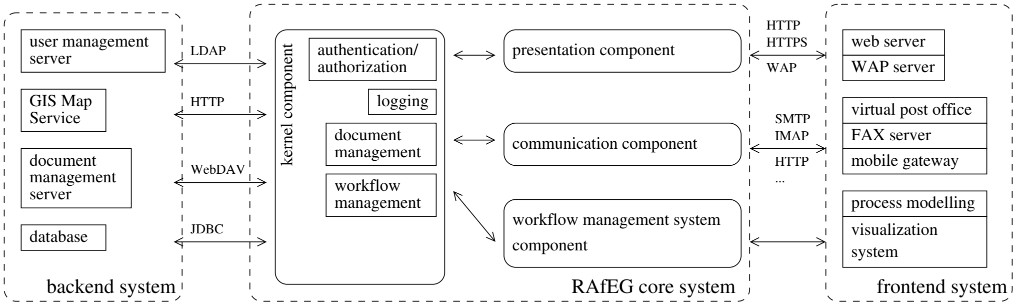


Figure 1. Structure of the reference architecture for e-government

proposed by the Workflow Management Coalition (WfMC) and the Object Management Group (OMG). Although many different standards are available in this sector we have chosen this solution because these standards are implemented in many of the available free WfMS and interoperability interfaces and methods are well specified. In particular, these standards are the Workflow Reference Model [6], the Interoperability Abstract Specification [8], Wf-XML [16] and XPDL [12]. Although free WfMSs use existing WfMC standards, the interoperability interfaces have not been implemented yet. So we implemented them ourselves. Details on the mechanisms implemented are given later in detail. Our implementation supports two free WfMSs at present. These are Enhydra Shark [15] developed by the Enhydra group and WfMOpen developed by the Danet GmbH [18]. As communication and especially secure and traceable communication are important, the solution implemented in the reference architecture is a combination of OSCI-Transport (Online Services Computer Interface) [13] and a self implemented interface for nonelectronic communication. OSCI-Transport fulfills the needs of communication in e-government applications and is proposed in SAGA [5] by the German Federal Government Co-ordination and Advisory Agency for IT in the Federal Administration.

2.1. Core system components

The core RAfEG system consists of four components: the kernel component, the presentation component, the communication component and the workflow management system component. The kernel component itself is composed of smaller subcomponents. Below, the separate components are discussed in detail.

2.1.1. Kernel component. The kernel component is responsible for allowing or refusing access from end users to the system, for coordinating access to the workflow management component for each user, for logging to allow a traceable history of all tasks and for the access to the document management server. Authentication and authorization is reached by the implementation of a security component. JBoss AS as well as other application servers support the definition of security permissions by implementing a special module that provides functions for checking username/password combinations. This module is used to authorize access to Enterpise JavaBeans. These Enterprise JavaBeans are the components of the core system that are remotely accessible. The logging ability of the kernel component is needed to allow the monitoring of all actions done by the user as well as the system and a failure analysis if problems occur during the execution of a procedure. The document management is especially important for the insertion of new forms, requests and texts of law. These are needed to allow an update of the system when procedures are updated or new ones are issued by the government. Furthermore, when a procedure is executed documents are edited, have to be read when a task is done or have to be inserted and checked when submitted by external requesters. By the use of electronic document processing the paperwork is profoundly reduced. To allow a rollback of a procedure it is also important that every version of a document is available. The kernel workflow component is also the connection point between the workflow management system component and the presentation and communication component.

2.1.2. Workflow management system component. In the reference architecture, official procedures are modeled as workflows as mentioned earlier. To support a wide range of WfMSs the workflow management system component is built as a standardized layer between the RAfEG kernel component and the underlying WfMS. The function of this component is mainly the provision of methods to access worklists, to integrate new procedures modeled as workflows and to update existing ones, to start workflows and to accept, execute and complete tasks by employees. Additionally, some features that are not implemented by workflow management systems are added to this component. Firstly, the mechanism for appending documents to workflows, because in official procedures most tasks include document processing. Secondly, the possibility of adding memos to tasks is also needed. This helps employees to pass on special execution notices for later processed tasks and also to read the notices of tasks processed earlier during the workflow execution. These memos are bound to the workflow they are created in. As this component is a layer around the actually utilized WfMS and the other RAfEG core system components, the WfMS can be easily replaced. This is especially important if the authority already has a specific WfMS in use and does not want to replace it.

2.1.3. Communication component. Communication should be secure, protocol independent and traceable. Secure means that it should allow an encrypted and userdependent transport of information within authorities as well as between authorities and other institutions. The criteria of protocol independence is needed, because to involve a large number of different authorities the RAfEG system must be able to deal with arbitrary equipment. Some offices may only support communication via HTTP(S), via email (simple mail transport protocol (SMTP)) and some may only support fax or even normal mail as they do not have an internet connection. In e-government the traceability of messages is an important factor that has to be considered. These three criteria lead us to the approach using the OSCI-Transport protocol which is proposed by the German government [5] to fulfill e-government communication standards between two RAfEG systems. By the use of this component encrypted messages can be sent between two RAfEG core systems and also between RAfEG systems and other governmental systems implementing this protocol. Additionally an own implementation of a mechanism to allow nonelectronic communication has been realized. This means in particular that a possibility for data extraction, the sending via nonelectronic communication methods and the import of returned data is possible.

2.1.4. Presentation component. The presentation component deals with the dynamic generation of the user interface (UI). When a new page is requested by the user interface the kernel component uses this component to create according output data in an own XML based format. This XML reply passes a processing chain. In the processing chain the data is at first parsed for errors, then transformed several times and finally serialized (transformed into the output format, e.g. HTML, WML or PDF).

The following example outlines this procedure. At first the workflow component creates an XML representation of the data that has to be displayed. This XML format without layout information is transformed via XSLT into a new XML document including layout information. After that internationalization is done using an i18n transformer and finally the XML is serialized into the specific output format, e. g. a web page. The Apache Cocoon framework [2] is the solution we have chosen for realizing that processing chain because it is easy to integrate into JBoss AS, it is freely usable and it supports the outlined mechanisms for converting XML into different output formats. Figure 2 shows the output of a worklist as web page in a web browser.

2.2. Frontend components

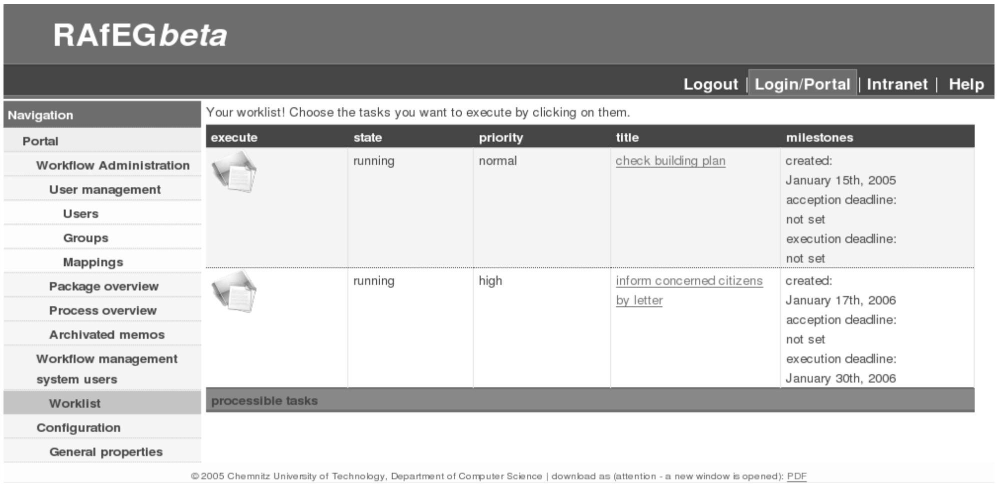
The main frontend component is the visualization system of the user interface. To allow a widespread usability of the system a web-based UI was preferred, because a web browser is available for many operating systems. The pages presented to the user are dynamically generated by the presentation component when requested by a web browser. It also enables users to administrate the system, provided that they have the appropriate authorization.

Another important component is the process modeling tool. A widely accepted modeling tool is the ARIS Toolset. With the help of ARIS, Event-Driven Process Chains (EPCs) which are developed by Scheer [14] can be designed in a concise way. For the purpose of editing the EPCs with additional tools an intermediary format is generated first, i. e. EPC Markup Language (EPML) proposed by Mendling and Nüttgens [10]. The transformation of the ARIS XML export (ARIS Markup Language – AML) into EPML is done via the AML2EPML [9] XSL script. In order to feed the workflow management system with the processes, EPML is converted using XSLT into the utilized workflow description language (in our case XML Process Definition Language – XPDL) in the next step. Finally it can be uploaded into the system via the user interface.

2.3. Backend components

The backend system is needed to provide additional functionality for the central system. The software used here is mainly third party. The components of this system can be easily exchanged to adapt it to work with existing software in authorities. These components are:

- the user management needed by the authorization module of the central system,
- the GIS map service needed especially for the test procedure, official approval of building plans, to provide maps and plans,
- the document management needed for the execution of procedures with linked forms, requests, texts of law and
- the database needed for saving configuration values, logging information and as data storage of the WfMS.



Your worklist! Choose the tasks you want to execute by clicking on them.

execute	state	priority	title	milestones
	running	normal	check building plan	created: January 15th, 2005 acception deadline: not set execution deadline: not set
	running	high	inform concerned citizens by letter	created: January 17th, 2006 acception deadline: not set execution deadline: January 30th, 2006

processible tasks

© 2005 Chemnitz University of Technology, Department of Computer Science | download as (attention - a new window is opened): PDF

Figure 2. Output of a worklist in a web browser

3. Mechanisms for the support of distributed procedures

Interoperability is necessary between two or more RAfEG systems within different authorities as well as between authority internal systems and systems that are accessible from external sites. External sites are citizens that can participate in governmental procedures and public authorities as well as external institutions. In our specific test procedure, the official approval of building plans, such institutions are agencies of urban planning. Interoperability is needed due to the fact that many official procedures are executed involving two or more authorities and institutions with their own IT infrastructure. These procedures are modeled as workflows and processed by the underlying workflow management system. Each workflow definition contains a description of all tasks that have to be executed and their dependencies. Every task has a performer or a group of performers that defines which person/persons have to perform the task. Additionally there are also tasks that are executed by the system itself. These tasks are particularly important for the start of an external part of the workflow. In RAfEG distributed workflows are modeled as an overall hierarchical workflow containing subflows that describe the external parts of a procedure. Every subflow can be executed by an external authority. The definition which authority and external company, respectively, has to execute a specific part is defined within the workflow of the procedure itself. This means that the subflow has additional information on the address of the corresponding system. To allow an easy rearrangement without changes to the workflow itself this information is a generic name that is mapped to the actual real address at runtime via a configuration file.

An example for the first part of a workflow of an official approval of a building plan procedure is shown in Figure 3. An official approval of a building plan starts with the receiving of a plan. The authority that received the plan forwards this plan to all concerned communes. This is the point in the workflow where communication between the authority and all involved communes takes place. The communes check the plan they received and make requests for changes if necessary. This checking of the plan is a workflow that can consist of one ore more tasks and is executed within the communes. The requests are sent back to the authority. The authority checks this requests and includes them in the received original plan if possible.

As outlined in the example, the sharing of data and the distribution of tasks between authorities and other institutions takes place on workflow level. Because free workflow management systems have not yet implemented the interoperability features proposed by the WfMC, RAfEG has its own interoperability component that can easily be adapted to WfMC conform WfMSs. The implementation is completed for Enhydra Shark at present and work on the adaptation for WfMOpen is in progress. The realization is based on a client/server architecture. Figure 4 illustrates the core components of the architecture and shows the steps that are necessary when an external subflow is executed.

When the execution of a procedure reaches the point where an external subflow should be executed, a system task is started which starts an application agent. This application agent communicates with a server object called `FactoryserverBean` to start the subflow on the child workflow system. The terms child and parent workflow system refer to the involved workflow management system components. Parent means that this system is executing the overall workflow and child means that this system is executing the subflow. The `FactoryserverBean` is an Enterprise JavaBean object that contains a set of factory objects. A factory object is the representation of an existing work-

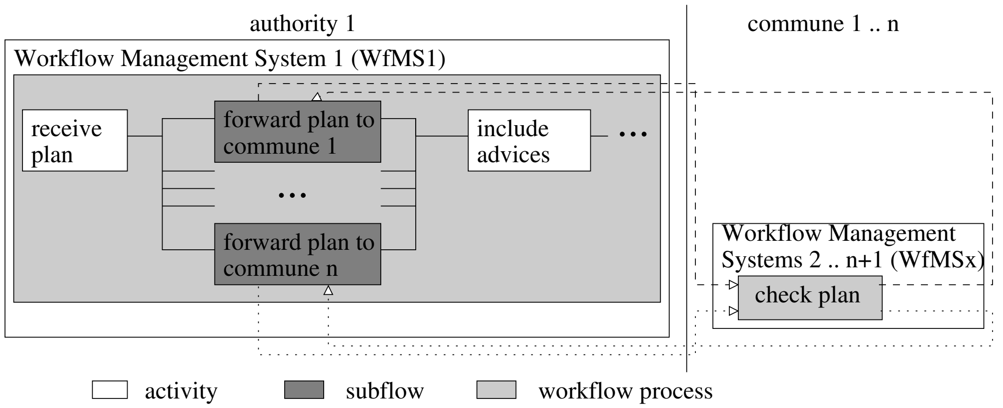

authority 1
commune 1 .. n
Workflow Management System 1 (WfMS1)
receive plan
forward plan to commune 1
...
forward plan to commune n
include advices
...
Workflow Management Systems 2 .. n+1 (WfMSx)
check plan
activity
subflow
workflow process

Figure 3. Example: the first part of the workflow for the approval of a building plan with distributedly executed subflows

flow definition in a WfMS that can be invoked remotely. Every factory object has a unique key that is part of the address defined in the subflow of the parent system to allow the direct communication between application agent and factory object. All information necessary to start the process within the client workflow system are stored in the specific factory object. When starting an external workflow, all relevant workflow data are sent to the `FactoryserverBean` (step (a)). The `FactoryserverBean` communicates directly with the underlying WfMS of the authority it belongs to. After initializing and starting the workflow (step (b)) and the creation of an instance object, it returns information on the successful creation. This information is mainly the key (identifier) of the instance needed for the parent system to associate received messages with a specific subflow. The application agent also creates an observer object (step (c)) within an internal `ObserverserverBean`. This observer is the object that receives information on the execution of the subflow and will be described later. Due to the fact that a factory object is only the representation of a workflow definition and starts the workflow when a request from an application agent is received another component is needed to inform the parent system about state changes and returns data after completion of the process. Therefore, the factory object creates an instance object for each workflow started. The progress of the subflow is reported to the instance by the child workflow system. The instance forwards this information to the observer object (step (d)) and this object forwards it to the parent workflow system (step (e)). The observer object is created by the application agent at the creation time of the external subflow within the `ObserverserverBean` to create a connection endpoint for the instance to send progress information and return values. As described earlier by the creation of an instance object every started external subflow creates its own observer object. When the subflow is completed the return data is signalled to the instance and forwarded to the observer object (step (f)). As by the transmission of progress information the observer forwards the return values to the parent WfMS which writes them back into the parent subflow representation (step (g)) and completes the subflow.

In addition to the start of the external subflow many procedures in e-government have documents like requests, forms and texts of law associated with tasks. To fulfill the requirements of exchanging the needed documents, other components than the workflow management system component are also involved in the external execution. Figure 5 shows these components of RAfEG and the steps that are taken by starting an external subflow. As one can see, the first step when an external subflow is executed is to start and initialize the external subflow (1). This step was described in the previous paragraph in detail. After starting the external subflow, all documents that are needed for execution in the external authority and company, respectively, are read from the document management system (2). After that, the documents are internally sent to the communication component (3). This component sends the documents using the transport method/protocol defined for communication with the communication component of the child system (4). The communication component of the child system notifies the created instance of incoming documents (5) and at last the instance retrieves the documents and saves them in the document management system of the child system (6).

4. Security mechanisms

During execution of a procedure a large number of participants are involved. They are located in different parts of the network. Firstly, there are participants working in the same authority (organization). They share one system. Secondly, there are participants in other spatially separated authorities using their own system. In this case it has to be

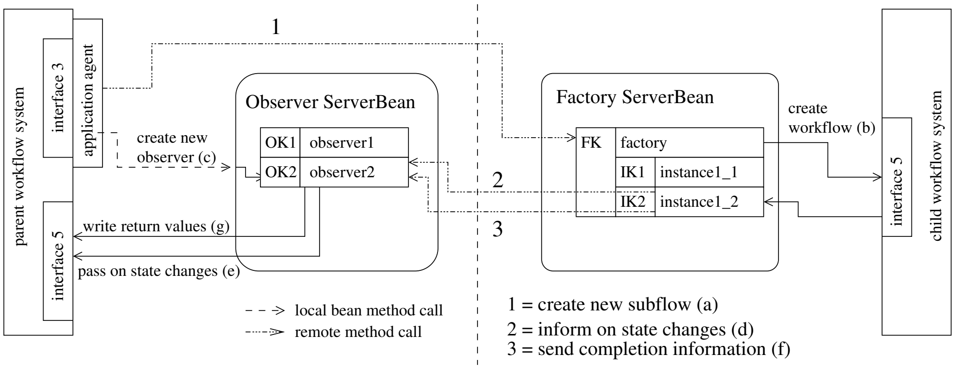


Figure 4. Necessary steps during the execution of an external subflow

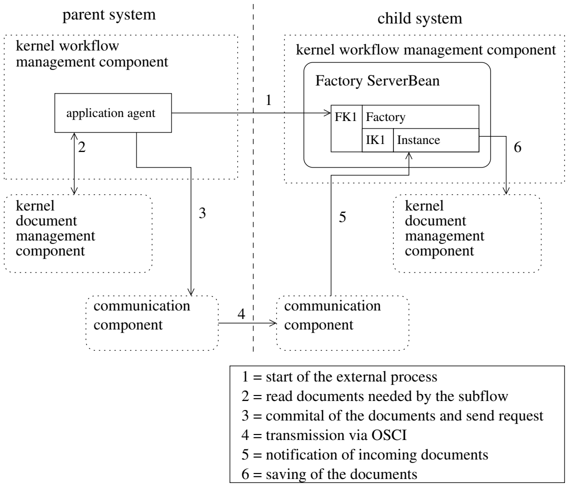

parent system
child system
kernel workflow management component
application agent
kernel workflow management component
Factory ServerBean
FK1
Factory
IK1
Instance
1
2
3
4
5
6
kernel document management component
kernel document management component
communication component
communication component
1 = start of the external process
2 = read documents needed by the subflow
3 = commital of the documents and send request
4 = transmission via OSCI
5 = notification of incoming documents
6 = saving of the documents

Figure 5. Workflow interoperability core components and necessary execution steps when starting an external subflow

considered whether the authority lies within or outside of the intranet-zone. Thirdly, there are several external participants like citizens. These three cases implicate different levels of security risks and must thus be handled separately.

4.1. Mechanisms for internal security

As already mentioned, all system components are remotely accessible by utilizing an application server and the Enterprise JavaBeans technology. In consequence the components can be distributed across several computers inside the organization. Communication between components is managed by the application server itself. Internally, Remote Method Invocation (RMI) is utilized.

To secure the access to remote Enterprise Beans, each bean is associated with a role. The role assigned to each user is stored in the user management system and is read out after a successfull login. If the caller of a bean does not have the specified role, a security exception is raised. Roles can even be attached to single methods of a bean. Thus, a fine granularity is reached. The security manager integrated into the application server is responsible for doing these checks and protects the application against unauthorized access.

The web based user interface (UI) is not only intended for handling the application but also for administration purposes. Hence, it has to provide different functionalities depending on the role of the user logged on. That way a “normal” employee is not able to, e.g., modify the system configuration which is reserved for administrators only. This is realized by a role filter which filters the navigation elements of the UI generated by the presentation component. In the processing chain of the UI generation the role filter is placed between content generation and subsequent XML transformation.

4.2. Mechanisms for external security

4.2.1. Intranet communication. The communication between different systems is mainly used to trigger the start of external subflows of the whole workflow. During this operation forms and documents (e.g. a building application) as well as workflow specific data (context data) is transmitted. In most cases the calling system has to wait for a reply (e.g. the filled request). This type of communication inside the intranet is done via a secure and reliable communication protocol. The protocol OSCI-Transport [13] is the protocol of our choice. This has two reasons. Firstly, it is the prescribed protocol for federal e-government applications in Germany. The use of this protocol will ensure a high compatibility with other applications in e-government. Secondly, the benefits of OSCI-Transport are the capabilities of the protocol to grant integrity, authenticity, confidentiality and traceability of the messages sent. Furthermore, OSCI-Transport has the ability to select between different encryption and authentication mechanisms depending on the security level needed.

4.2.2. Communication between citizens and the system.

The web-portal forms the user interface for the system. It would be possible to share the web-portal between the employees of the authority and participating citizens. This would be beneficial because of the simplified management of data. However it implies that the hosting server would be accessible from outside the intranet. This comprises a very high security risk. A potential attacker could possibly take over the web server and gain access to the naturally sealed off intranet and all its critical data. Hence the public part of the web-portal as well as the user interface for external participants have to be located on a separate server not connected to the intranet.

Synchronization of data between the external server and the system can be done in different ways. In the last resort data transport can be realized via an electronic storage medium like a CD or similar. But also a secure internet connection initiated by the system is possible.

Still, the risk of hostile take-over of the external server exists, but it is not possible for an attacker to gain access to the system located inside the intranet. A potential attacker could at the utmost modify workflow relevant data or could pretend a fake identity. That must be checked manually.

5. Conclusion

The lack of e-government software systems that are flexible, adaptive, applicable in a wide range of e-government procedures and equipped with up to date security functionality has been the reason for us to implement the reference architecture for e-government. The reference architecture enables authorities to execute their procedures computer-assisted and thus efficiently. By the use of a workflow management system and the modeling of procedures as workflows the system gains highest independence from specific application areas and is widely usable in authorities. Due to the fact that all documents are included in the system and can be filled out via the frontend, the paperwork is omitted. The component based design of the RAfEG system and the use of open source software components and formats enables authorities to adapt the system to individual requirements. Additionally it supports inter authority procedure execution and authority to external institution procedure execution. The goal has been to implement a system that can cope with up-to-date security restrictions in e-government that are proposed in the “Standards and Architectures for e-Government Applications" (SAGA) our system can be also installed in high security critical authorities.

References

[1] Austrian Federal Ministry of public service and sport - working group ELAK. Der Elektronische Akt (ELAK), 2001. `http://www.cio.gv.at/elektronischerAkt/`.

[2] B. Brogden, C. D’Cruz, and M. Gaither. *Cocoon 2 Programming: Web Publishing with XML and Java*. Sybex, 1st edition, 2002.

[3] L. G. DeMichiel, L. U. Yalcnalp, and S. Krishnan. Enterprise JavaBeans Specification, Version 2.0, 2001. `http://java.sun.com/products/ejb/docs.html`.

[4] Fabasoft. eGov-Suite, 2005. `http://www.fabasoft.at/html/eGov/egov-produkt.htm`.

[5] German Federal Government Co-ordination and Advisory Agency for IT in the Federal Administration. Standards and Architectures for e-Government Applications (SAGA) 2.0, 2003. `http://www.kbst.bund.de/Anlage304417/Saga_2_0_en_final.pdf`.

[6] D. Hollingsworth. *The Workflow Reference Model (TC00-1003)*. WfMC, Jan 1995. `http://www.wfmc.org/standards/docs/tc003v11.pdf`.

[7] JBoss Application Server, 2005. `http://www.jboss.com/products/jbossas`.

[8] M. Marin. *Workflow Standard - Interoperability Abstract Specification (WfMC-TC-1012)*. WfMC, Nov 1999. `http://www.wfmc.org/standards/docs/TC-1012_Nov_99.pdf`.

[9] J. Mendling and M. Nüttgens. Transformation of ARIS Markup Language to EPML. In *Proc. of the 3rd GI Workshop on Event-Driven Process Chains (EPK 2004)*, Luxembourg, Luxembourg, 2004.

[10] J. Mendling and M. Nüttgens. EPC Markup Language (EPML). Technical report, Vienna University of Economics and Business Administration, 2005. `http://wi.wu-wien.ac.at/home/mendling/publications/TR05-EPML.pdf`.

[11] R. Monson-Haefel. *Enterprise JavaBeans*. O’Reilly, 4th edition, 2004.

[12] R. Norin. *Workflow Process Definition Interface – XML Process Definition Language (WfMC-TC-1025)*. WfMC, Oct 2002. `http://www.wfmc.org/standards/docs/TC-1025_xpdl_2_2005-10-03.pdf`.

[13] OSCI Leitstelle. OSCI-Transport 1.2, 2002. `http://www.osci.de/materialien/osci-specification_1_2_english.pdf`.

[14] Scheer, A.-W. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer, Berlin, 1994.

[15] Shark 1.1 documentation, 2005. `http://shark.objectweb.org/doc/1.1/index.html`.

[16] K. D. Swenson, M. D. Gilger, and S. Predhan. *Wf-XML 2.0 - XML Based Protocol for Run-Time Integration of Process Engines*. WfMC, Nov 2004. `http://www.wfmc.org/standards/docs/WfXML20-200410c.pdf`.

[17] M. Wahl, T. Howes, and S. Kille. RFC 2251: Lightweight Directory Access Protocol (v3), 1997. `ftp://ftp.rfc-editor.org/in-notes/rfc2251.txt`.

[18] Wfmopen, 2005. `http://wfmopen.sourceforge.net`.

[19] E. J. Whitehead, Jr. and Y. Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the web. In *Proc. of the Sixth European Conf. on Computer Supported Cooperative Work (ECSCW’99)*, pages 291–310, Copenhagen, Denmark, 1999.