BACHELOR THESIS

# Design and Implementation of a Web Service Development Portal –The case study DATAPORT

Submitted by:
Jun Zhang
Matriculation Number: 15239
**Hamburg University of Science and Technology**
**Germany**

Supervised by:

Prof. Dr. Joachim W. Schmidt
and
Dipl. Inform. Rainer Marrone
Software Systems Institute
**Hamburg University of Science and Technology**

Hamburg, 24.10.2005

# Declaration

I declare that:

this work has been prepared by myself, all literally or content-related quotations from other sources are clearly referenced, and no other sources or aids out of the declared reference are used.

Hamburg, 24.10.2005

Jun Zhang

I would like to thank Professor Joachim W. Schmidt of STS for supervising this thesis and being very helpful with guiding the project overall and finding a topic for my work.
Thanks also go to Dipl. Inform. Rainer Marrone, who guided me through the whole project and this thesis and offered great help on developing the whole work.
Thanks also go to Birgit Guth, Jürgen Meincke and Werner Wendt from Dataport, who provided useful information of Dataport and advices through this work.
Dr. Hans-Werner Sehring and Sebastian Boung of STS were very helpful in providing advices during the project.

# Contents

# List of Figures

# Chapter 1

# Introduction

Since the beginning of 2004 Dataport [Dat05] has become the official IT service provider for the government Hamburg and Schleswig-Holstein. The information and communication service areas of Dataport are: IT consult for equipment, planning, data protection and IT security, application development for city governments and communes, computer center provider with different platforms with high availability, system management, IT support for office work, network infrastructure, network management, internet services like planning, realisation, hosting, firewall and E-Mail service, telecommunication services, etc.

One of the most active areas of Dataport is the GovernmentGateway, which has already become a productive field since 2003. The GovernmentGateway plays the role as a bridge between the governments, citizens and companies. It acts as a Gateway and provides the citizens and companies access of various online services and applications. For increasing the interoperability, Web Service is used between the Inter-/Intranet and the backend.

Dataport has many Gateway application developers, not only developers from Dataport itself, but also developers from customers, who develop client applications for interacting with services provided by Dataport, i.e. client applications for Web Services. Since developers are at different locations, it is necessary nowadays to provide a central point for supporting information exchange among all those developers. Because of the large number of service fields provided by Dataport, it is important to manage those huge amount of information in art of documents, code examples, interface descriptions and database backups for sharing and retrieving.

As a solution for providing a central point, a Development Portal should be build. This Bachelor Thesis covers this project as a case study.

## 1.1 Objectives and Goals

As an initial idea, Dataport has suggested to build a UDDI management system as a portal. But the goal of the portal is not only publishing Web Services, but also the management of all Web Service related documents. More important is the requirement, that the dependencies between documents should be traceable, which means, the portal should be able to dealing with semantic relations between documents. Therefore, a portal, which provides document publishment,

document management and document retrieving with the option of tracing document dependencies, is more comprehensive as a single UDDI management system.

The goal of this thesis is to design and build such kind of portal for the company Dataport. The portal will be generally used as a Document Management Portal, which will also be integrated into the GovernmentGateway for providing various information later. The major user groups of this portal are both internal and external developers from Dataport, who are going to deposit and share Web Service related documents (text documents, program source code, binary files, etc.) in the portal, which is so-called Web Service Development Portal.

## 1.2 Structure of the Work

The structure of this thesis is described as follows:

Chapter two will provide the detailed motivation of this project followed by a general overview of Web Services, required core functions and realisation ideas of this Web Service Development Portal.

Chapter three deals with the standard architecture of the information portal platform infoAsset Broker and its core functions.

Chapter four will describe the analysis, design and implementation of the Web Service Development Portal. The major focuses of this project will be described in detail. A general overview of the GovernmentGateway will also be briefed. At the end of this chapter, a short evaluation of the implemented Web Service Development Portal will be provided.

Chapter five, the final one, will summarize the entire work and provide an outlook for the future development.

# Chapter 2

# A Web Service Development Portal

Since years, Information technology support can be found in almost every part of the civil service in government area. After the born of the electronic Government, citizens and companies are given the chance to join the public administration work from the government through all kinds of electronic median over the Internet or Intranet. One example here is: citizens can register/edit their resident status directly over the online GovernmentGateway of Hamburg. An electronic payment system is also integrated into the GovernmentGateway, so that people can conveniently pay the service fee online. Another example is: companies, who would like to retrieve information on the real-time ship position at the Hamburg harbour, can buy that information from the GovernmentGateway over the Internet or Intranet. With all those IT solutions, the cost of such public administration will be reduced and the communication between citizens, companies and the governments will be raised to another higher level.

To realise those solutions, we need portals, because portal is an union of information presentation and application from the Internet and Intranet. As we are living in the time of the Web Services and trying to solve IT interoperability problems, portals that support Web Services are more than welcome. In this chapter I will introduce this bachelor project on building a Web Service Development Portal that will be integrated into GovernmentGateway later.

## 2.1 Motivation of the Web Service Development Portal

In order to raise the level of the automation of all kinds of services or working processes, electrical documents are used in almost every field. During the processing and managing of the electrical documents, traceability and legality are very important in this area, thus, a portal for appropriate deposition and archiving is indispensable.

Developers from Dataport and its customers are distributed in several offices and two federal states, the portal should meet the needs of distributed documents sharing and exchanging. Since a portal is an aggregation of presentation of information and application in the Internet, Intranet and Extranet, a document management portal is a suitable interface between the customers and the providers. As mentioned in chapter one, the portal will be integrated into the GovernmentGateway, therefore, interoperability turns out to be a necessary point and a document management portal with Web Services support will be the best choice to reach the requirements of this project.

## 2.2 Web Service Introduction

In the E-Business world there is a basic scenario: a provider offers a service to a customer, with which the customer should make a Remote Procedure Calls (RPCs) against an object over Internet, Intranet or Extranet. At the time as the eXtensible Markup Language (XML) has not established yet, sending such kind of calls is depending on the platform of the provider, for example: Distributed Component Object Model (DCOM) can only be used for windows platform, Remote Method Invocation (RMI) can only be used under Java Runtime Environment, etc., which means, customer might have to use some kind of "middle man"to build up a communication channel between two applications based on different platforms. But this has been changed since XML came up to the stage, because XML is a standard language that is supported by almost all kinds of platforms. Therefore after the XML-based Web Services began its arena, together with other platform-independent standard technologies like HyperText Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI), it's a whole lot easier to set up cross-platform communication. This chapter is going to give some basic ideas on the Web Service technologies.

### 2.2.1 What are Web Services

Some of the definitions of the Web Service are:

A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [Con04].

A Web Service is a service that communicates with clients through a set of standard protocols and technologies. These web services standards are implemented in platforms and products from all the major software vendors, making it possible for clients and services to communicate in a consistent way across a wide spectrum of platforms and operating environments [Ort05].

A Web Service implements an interface that describes a collection of network-accessible operations through standard XML messaging [ZTP03].

A Web Service is a collection of protocols and standards used for exchanging data between applications or systems [Wik05].

So basically, the Web Service is a service, which is based on XML technology combining with lots of standard protocols and growing technologies, its idea is to use those well-established and proven concepts to increase the interoperability between different applications on different systems.

### 2.2.2 Architecture of Web Service

In the Web Service architecture there are some roles can be defined:

The service provider offers Web Services on the Internet, Intranet or Extranet. The services will be described in the WSDL, which provides a standard interface for the communication. If the provider want to let more requestor to find/reach the services, then the Web Services should be published to a public discovery agency.

The service requestor or service consumer will look for Web Services offered by service provider or search by the discovery agency for certain Web Services. They will become all needed information from the exposed WSDL on how to bind to the service provider and how to invoke the Web Services.

A discovery agency offers a repository for the service providers to register their Web Services, therefore the service requestor can search through the repository and retrieve the desired information on the Web Services.

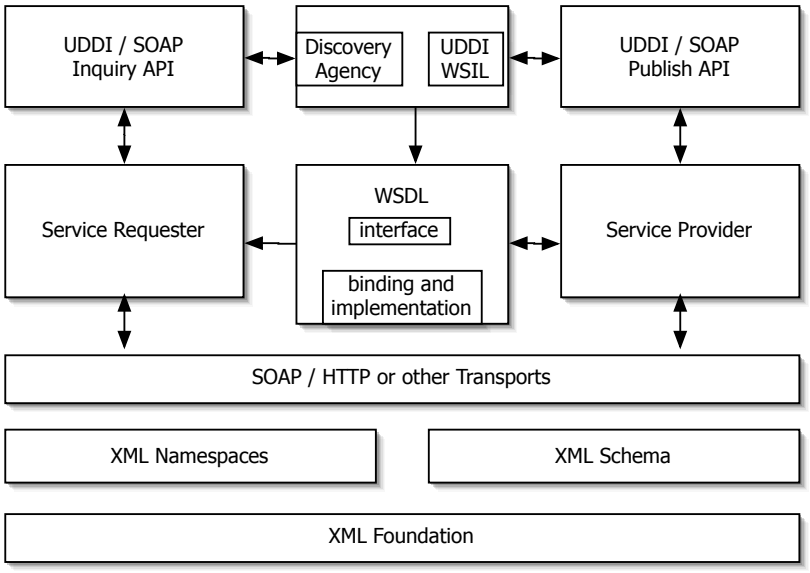


Figure 2.1: Web Services building blocks (static view) [ZTP03]

As we can see in figure 2.1, the WSDL is the key element in this picture, it provides a model and a XML format for describing Web Services. WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as ”how” and ”where” that functionality is offered.

WSDL describes a Web service in two fundamental blocks: one abstract and one concrete. At the abstract level, WSDL describes a Web service in terms of the messages it sends and receives. Messages are enclosed in some Operations. Datatypes are defined which will be used

in the messages. At the concrete level, a binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. And finally, a service groups together endpoints that implement a common interface. For a detailed introduction, please visit [Con05a].

The underlying communication protocol between requestor, provider and discovery agencies is SOAP. SOAP is a standard protocol used to exchange information between applications regardless of the object models, programming languages and operation systems. SOAP uses XML messaging type and by far in Remote Procedure Call (RPC) format.

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics [Con03].

The discovery agency uses UDDI or Web Service Inspection Language (WSIL) to offer the repository for the provider. UDDI defines a structure for the registry, together with a publishing and an inquiry Application Programming Interface (API) for accessing the registry. WSIL is a lightweight alternative to UDDI.

UDDI represents a set of protocols and a public directory for the registration and real-time lookup of Web services and other business processes. In many ways, UDDI models a “White Pages”, providing a listing of services available within a network [Wav04]

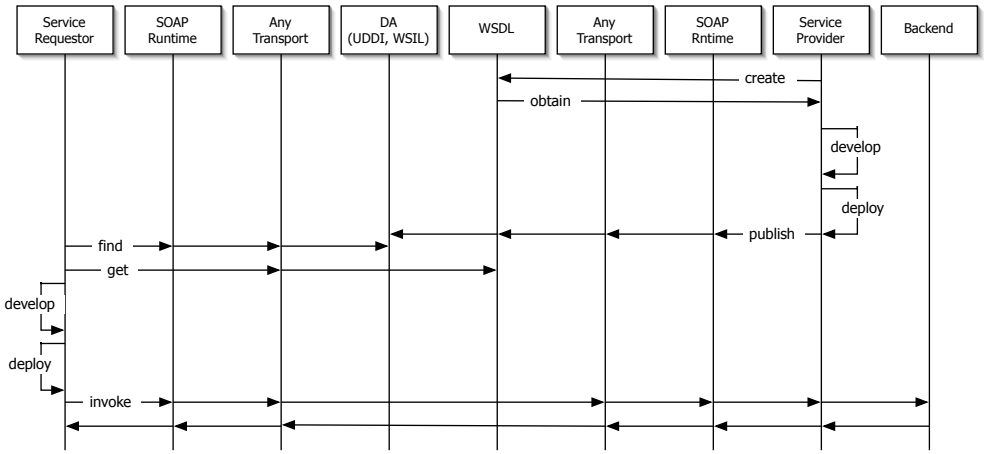


Figure 2.2: Web Services building blocks (dynamic view) [ZTP03]

In figure 2.2, the interactions between all the components are described in a UML sequence diagram. As we can see, communication between service requestor and service provider in the A2A solution is via XML document exchange over Hypertext Transfer Protocol (HTTP). Hence, there is no human-to-machine interface at all and totally independent from the backend platform.

## 2.3 Core Functions of the Web Service Development Portal

A portal system for document management should be build, which will be integrated into the GovernmentGateway later and will also be used in other development projects. The Development Portal should provide an aggregation of necessary information for all participated developers, whereas the traceability and legality must be ensured, hence, not every participant is allowed to access everything. Since the access of the Development Portal come from both Intranet and Internet, the GovernmentGateway should be used as the interface between the Internet access and the Development Portal. Advantage of this idea is: the proved authorization system of the GovernmentGateway can be used as the authentification mechanism for the Development Portal dealing with users from the GovernmentGateway.

The access (which is the read-access) of the Development Portal from Internet will be provided over the GovernmentGateway. The access (which is the read-/write-access) of the Development Portal from Intranet will be provided over a login-system from the Development Portal itself. Access over GovernmentGateway will be authenticated by the Gateway, whereas the available user-role will be carried forward and observed. A Web Service interface will be realised for the information presentation from the Development Portal over the GovernmentGateway.

The Authorisation Concept should be designed as flat as possible for later extension. The read-access shall be based on the document-level and the write-access shall be based on the directory-level. Which means, that every single document should be able to be granted with read-access to users (user group), no matter whether the user group is new or not.

Content dependencies and relationships between documents, especially between programming code documents, should be presented with hyperlinks, through which an overview of the dependencies and relationships can be provided as well a tracing function can be realised.

Documents in the Development Portal should not be deleted, instead of deletion, the documents should be able to be deactivated. Deactivated documents will not be shown in normal utilisation, so that the view of documents can be reduced of the most important documents. Deactivated documents though should be hyperlinked and be traceable.

A subscription service should be realised in the Development Portal, so that the users can be notified if new documents are deposited or any modification are made on the documents, whereas the users shall subscribe for those documents they are interested in.

The quality of documents in the Development Portal will be controlled. This will be reached through a special user group in the Development Portal, called “Fachliche Leitstelle”, which is a professional controller group. Developers create new documents, modify documents or apply deactivation of documents, a request on the “Fachliche Leitstelle”will be followed. The “Fachliche Leitstelle”will then decide if the documents can be published or the authors of the documents should make some further modification on the documents. In the second case, modification advices will be sent back to the developers. In the first case, the “Fachliche Leitstelle”will change the state of the documents, which means, the new documents will be published, new version of a document will be published (which cause the old version of document to be deactivated) or a document will be deactivated. The “Fachliche Leitstelle”at this place

can only change the state of the document but not the Meta-Data of the documents themselves.

Developers should also be able to search the documents. Documents can be searched by categories, keywords and classification, etc. as well as with full text search. Deactivated documents will not be included in the normal search process, but those will be included in the detailed search process. Documents version should be shown in the search result to keep the new version of documents traceable.

## 2.4 Implementation Choices

As realisation solution there are many successful portal software in the market, such like: Plumtree Corporate Portal from Plumtree, Microsoft Sharepoint Server 2003 from Microsoft and infoAsset Broker from infoAsset AG. Now i will introduce these three portals shortly and our choice of solution comes after that.

**Plumtree, Plumtree Corporate Portal, Search Server**

The Plumtree Corporate Portal brings together people, processes and systems into composite applications, indexes and organizes content, and rationalises security and user informaiton. The portal hosts the user experiences of composite applications, as well as the administration framework. The architecture of the portal, alone in the industry, is entirely based on Web Services.

Plumtree Search Server indexes all of the resources accessible through the portal pages, communities and web applications deployed across the enterprise. These resources include:

- content indexed from file systems, web sites and document databases
- project documents and web pages created and stored by Plumtree Collaboration Server and Plumtree Content Server
- applications, portlets, communities and users

With Search Server, portal and application users can access a wide expanse of content, services and people in a single stroke, and the organization does not have to maintain separate indexes for portal, collaboration and content management products. [Plu05]

**Microsoft, SharePoint Portal Server 2003**

Microsoft SharePoint Portal Server 2003 enables enterprises to develop an intelligent portal that seamlessly connects users, teams and knowledge so that people can take advantage of relevant information across business processes to help them work more efficiently. SharePoint Portal Server 2003 provides an enterprise business solution that integrates information from various systems into one solution through single sign-on and enterprise application integration capabilities, with flexible deployment options and management tools. The portal facilitates end-to-end collaboration by enabling aggregation, organization and search capabilities for people, teams and information. Users can find relevant information quickly through customisation and personalisation of portal content and layout, as well as by audience targeting. Organizations can

target information, programs, and updates to audiences based on their organizational role, team membership, interest, security group, or any other membership criteria that can be defined.

SharePoint Portal Server 2003 enables a single point of access to multiple systems such as Microsoft Office System programs, business intelligence and project management systems. The portal, built on a scalable, highly distributed architecture, provides flexible tools for deployment, development and management, all of which enable the portal to grow with the organization's needs. These integration features enable customers to harness information to make use of their company's resources. Users can extract and reuse timely and relevant information from systems and reports, and quickly locate and access documents, projects, and best practices across the company. The portal features search technology developed by Microsoft Research that enables users to search file shares, web servers, Microsoft Exchange Server public folders, Lotus Notes, and Windows SharePoint Services sites out of the box. In addition, user can organize documents and information by topic and browse for relevant content. Alerts notify users when new information is added or existing information changes to help users better use the data. [Mic05].

**infoAsset AG, infoAsset Broker**

The infoAsset Broker is a platform independent server software that is a pure Java-2 product, which can use various file systems, database systems or content management systems as information depository.

The data, business logic and presentation layers are strictly separated. A Web Service interface is supported, which allows users of GovernmentGateway to authenticate themselves and retrieve information from the broker. With the Web Service interface it enables the infoAsset Broker to act as a backend information service and other web servers to take over the presentation layer.

Documents in the infoAsset Broker can be bidirectional linked, through which the knowledge base and semantic relation can be presented over these hyperlinks. The built knowledge base in the infoAsset Broker is visualised through Java Applet, and works also as part of the navigation system. Such kind of hyperlinked knowledge base is almost indispensable in a portal where the complexities of the information relationships are really high.

The requirement on a controller user group, the "Fachliche Leitstelle", is also realised in the infoAsset Broker. This "Fachliche Leitstelle"controls the state changes of documents, which include acceptance or denial of publication of new documents and new versions of documents or deactivation of documents. In case of a denial action, the "Fachliche Leitstelle"will send a feedback to the author of the document with reasons of denial and advices. New documents in the infoAsset Broker can be automatically published at a given time in the future, only if the document is proved by the "Fachliche Leitstelle".

Old or useless documents in the infoAsset Broker can be deactivated, through which those documents will not be listed in the active directory. Those deactivated documents that do not have any follow-up new versions can still be found through search function by searching after historical documents. Those do have sequel versions can also be traced through the hyperlink

of historical documents of the new versions.

Subscription service is also provided in the infoAsset Broker, with which the users of the Broker can trace the changes on documents they are interested in. Subscribed users will receive notifications when changes are made to the documents. User can choose how often they want to be notified, provided time intervals are: immediately, daily, weekly or monthly. Notifications can be retrieved directly in the infoAsset Broker user interface, they can also be sent in E-Mail format at selected time interval.

The authorisation system in the infoAsset Broker is flat and flexible, which is also based on document level. This allows the infoAsset Broker to be extended in future demands.

## 2.5 Summary

All three systems have their basic strength: modeling of business processes, user and user role management and document management. All three systems have also their own templates system, with which the presentation layer can be adapted in customer's way.

The Plumtree Portal Server has a separate searchmachine, whereas a searchmachine with complete functionality are fully integrated in both Microsoft SharePoint Server and infoAsset Broker. Documents in both systems can be classified and categorized with keywords.

Among those three systems, only the infoAsset Broker provide the function of a controller user group, which is called "Fachliche Leitstelle", through which the document publication and deactivation will be controlled.

The critical point for choosing a platform is the integration into the GovernmentGateway. The authentification over GovernmentGateway is strong necessary, only then can the existed and proved authorization and privacy protection concepts for external customer of the GovernmentGateway be taken over.

Above are three main differences between these three systems. As we can already see, only the infoAsset Broker meets all the requirements of this project. To choose infoAsset Broker as the realisation platform is therefore the final decision.

In the coming chapter three, I will introduce you more details of the infoAsset Broker.

# Chapter 3

# The infoAsset Broker

The basic motivations of the infoAsset Broker are based on problems from using and organizing company knowledge resources. Those problems are:

- Explosive growth of knowledge and content Assets
- Documents and Data in the Intranet are deposited on different servers at different places, which leads to a complex administration with higher costs
- In most cases there are no sustainable documentation systematices, neither is the repository transparent enough
- Hierarchical data repository is not scalable with more complex content assets
- Access authorisation is not transparent, which is difficult to administrate
- Persons who have the know-hows are not identifiable

These problems result in:

- Existing knowledge resources are not used efficiently
- Neither completeness nor relevance are provided by information retrieval

The infoAsset Broker is an enterprise information portal platform for knowledge management, which is a platform-independent server software for the construction of personalized content portals for the Internet as well as corporate knowledge and information portals used in corporate Intranets. It provides interactive knowledge maps, integrated document management, skill management functions, automatic classification of documents and notification service depends on personal interest as well. File systems, databases or content management systems can serve as information repositories. A combination of these repositories can be used according to customer needs [inf05]. With these features, problems briefed above are solved. An overall schematic diagram of infoAsset Broker is shown in figure 3.1.

The following text describing Concepts, Basic System and Core Functions of the infoAsset Broker are based on the original german text from [inf05].


Figure 3.1: Schematic Diagram of infoAsset Broker [inf05]

## 3.1 Concepts Introduction

**Unified Asset Management**
All information objects in the infoAsset Broker, for example, concepts, documents, directories, persons, groups, projects, are all concerned as ”Asset” consistently. This concept makes it possible, that all the generic functions of the infoAsset Broker can be applied to all Assets, regardless what type of information object the Asset is. Those generic functions are for example: searching, hyperlinking, classification and event-based notification service.

**Unified Link Management**
All the Assets in the infoAsset Broker can be crosslinked. The Knowledge-Map and the semantic relationship between Assets can be presented easily over the hyperlinks. The Unified Link Management concept enables the heterogeneous link collection and tracing of information, no matter the information is popular or not.

**Separation of Content and Layout**
The Template-Technology in the infoAsset Broker supports a clear separation of processing-logic, contents and layout. Dynamic generated WML-/HTML-pages will be applied with a standard layout, besides, this Template-Technology also enables the client-specified customisation.

Placeholders are used in the Template-Technology for placing dynamic contents, the dynamic

contents are provided by the associated Handler. Handlers coordinate the user requests and the responsible business processes. Incoming user requests are dispatched to different Handlers. Each Handler can handle some specific request types, and start some related business processes. When the processes are done, the response will be sent back by Handler using the Template Engine. Figure 3.2 shows the processing of a user request.

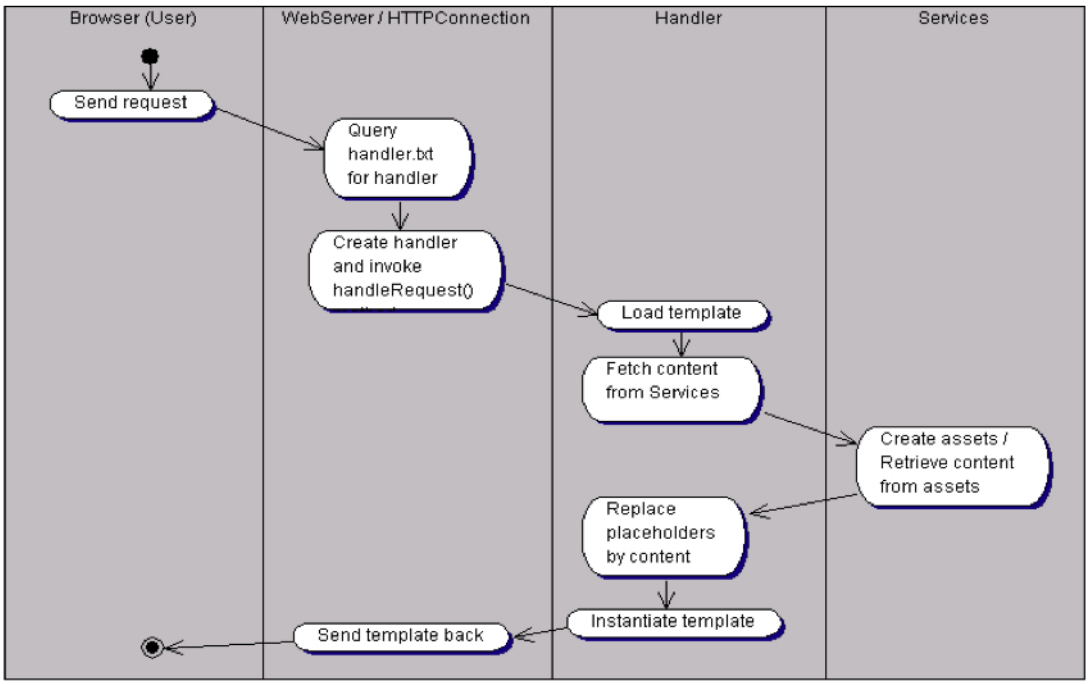


Figure 3.2: Processing user request in the infoAsset Broker [Weg00]

**Open Customisation of Information Objects**
The schema of information objects for modeling Assets is flexible, the structure can be customised on demand of the client.

**Open System Interfaces**
The interfaces of the infoAsset Broker standard software are open over all the system layers, which enables a high extensibility in scope of Services, Tools and Content Management.

**Object-Oriented Programming**
The infoAsset Broker uses exclusively modern Object-Oriented Modeling Methods in designing information structure, business logic and interfaces.

**Easy Customisation**
The infoAsset Broker is a generic standard software, and is applicable out of the box. Its simple customisation mechanism provides an efficient platform for creating client-specific portals.

## 3.2 Basic System Description

The infoAsset Broker is a consolidated tool for utilising heterogeneous knowledge resources in the corporation. With its Unified Asset Management, the infoAsset Broker integrates documents, persons, concepts and rating etc. together in one user interface, therefore it accesses different data sources like file systems, databases or content management systems.

To utilise bulked information assets, thematic classification of the knowledge is needed. With the infoAsset Broker can corporation-wide Taxonomy be created and managed. Terms and concepts will be presented in a Knowledge-Map graphically and interactively. The Knowledge-Map provides a structured and transparent presentation of the knowledge resources, with which direct navigation over terms or concepts to information assets and know-hows is made possible.

The integrated Document-Management-Functions in the infoAsset Broker enables for example Versioning, Publication and Classification of the normal Office-Documents. Documents in the infoAsset Broker are hyperlinked with the authors of the documents, therefore, a bridge between explicit knowledge and personal experiences (implicit knowledge) is build, as a result, knowledge can be discovered more effectively.

An useful Skill-Management is also build in the infoAsset Broker, so that the know-how porters can be found quickly.

Groups can be created and managed easily in the infoAsset Broker. Groups have their own access rights and are administrated decentrally, which reduces administration expenses at a high level.

The access protection in the infoAsset Broker is based on the Group Memberships and Roles, which assures the integrity of all the information objects as well as the collaboration with external partner, who can be granted access to certain part of the enterprise information portal.

### 3.2.1 Architecture of the infoAsset Broker

As an enterprise information portal, the infoAsset Broker is build completely in Java-2. It provides the user the access to the portal through different kinds of end devices, for example: Web-Browser, WAP-Handys, PDAs with WAP/Web-Browser etc.

The infoAsset Broker is build consequently after Client-Server-Architecture, which ensures the scalability of the system. Figure 3.3 shows the schematic composition of the infoAsset Broker architecture.

The clients of the infoAsset Broker communicate with the Broker Server only through HTTP-Request and HTTP-Response, so that the communication is also possible over firewall. The interactive Knowledge-Map of the infoAsset Broker needs Java in Web-Browser to be enabled, besides, the Web-Browser should also be able to process JavaScript. Some supported Web-Browser are for example: Netscape (6.1+), Microsoft Internet Explorer (5.0+), Mozilla (1.0+) and Opera (6.0+).

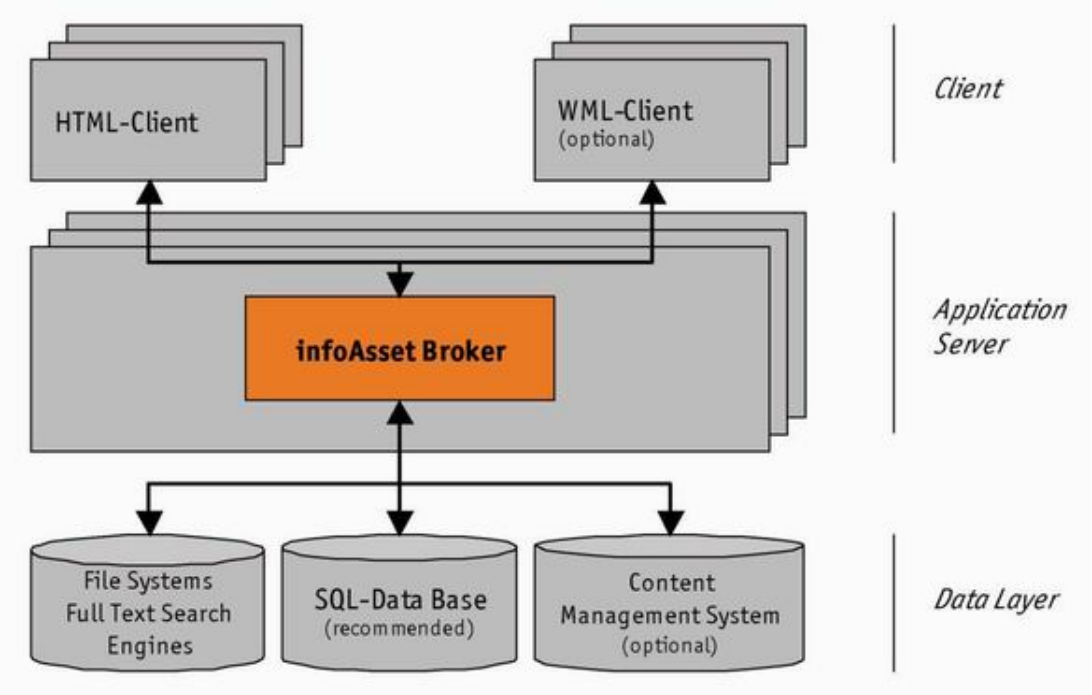


Figure 3.3: Schematic Composition of the infoAsset Broker Architecture [inf03]

Recommended Server Platform are for example: Sun Solaris, HP-UX, Linux, Windows NT, Windows 2000 Server, Windows 2003 Server. Supported repositories are for example: Oracle, IBM DB2, MS SQL Server and MySQL. (All support fulltext search).

Another detailed Architecture of infoAsset Broker will be shown in figure 3.4:

### 3.2.2 Core Functions of the infoAsset Broker

**Documents: collection and distribution of explicit knowledge**
Multimedia-based documents, such as MS Word-Documents, images or video-clips, can be uploaded onto the infoAsset Broker and become accessible.

The integrated Document-Management-Functions include full-text search, metadata-search, versioning, publication and auto-categorisation by means of the Knowledge-Map.

**Personal- and group-profile: description of implicit knowledge porter**
The infoAsset Broker manages standardised and detailed personal-profile for users, anonymous guest-users, colleagues, customers, etc.

The visibility and the level of detail of the personal-profiles in different groups can be adjusted separately.

The Group-Management supports the presentation of areas, departments, project groups, com-

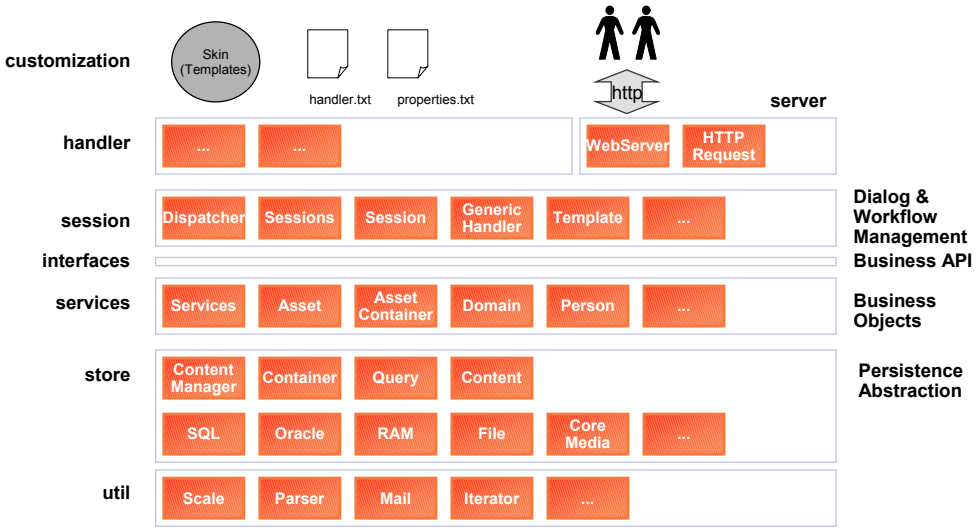


Figure 3.4: Detailed Architecture of the infoAsset Broker [inf03]

munities of practice, etc.

The granting of authorisation is based on groups, which means, the user owns the combined authorisation of all the groups, in which the user has the membership.

Personal data, registration information and group membership can be exchanged with other systems over LDAP (Lightweight Directory Access Protocol) optionally.

**Concepts and Knowledge-Map: Structuring of Knowledge**
The corporation-terminologies will be collected structured using the concepts in the infoAsset Broker.

Concepts and their relationships build up the Knowledge-Map, which is a binding taxonomy for the classification of the information objects.

Starting from just one concept, user can be guided systematically to further relevant information, for example, knowledge porters and documents etc.

**Community-Functions: Support of Teamwork**
Topic Discussion-Forum and Online-Chats are provided for direct knowledge exchange. E-Mails, MS Netmeeting and AOL Instant Messenger are integrated as well.

**Personalised Services**
Personal folders are offered for collecting, on-going working and rating of information objects in any format.

Personalised rating system is useful by sorting documents after personal rating, which raises hit-relevance of the search results.

**Subscription Service and Alerting Extension**
Users of the infoAsset Broker can subscribe to Assets (documents, persons, topics and groups) they are interested in, so that they can be notified whenever there are changes be made on those subscribed Assets through the Alerting Extension.

The Alerting Extension arranges an individual, automatic notification service for creation, modification and deletion of the Assets. Users are allowed to choose when they want to be notified and in what kind of format, they can choose daily, weekly or monthly notification, and in online message report, E-Mail bulked report or sms report formats.

## 3.3 Summary

In this chapter i have introduced the concepts of the infoAsset Broker Enterprise Information Portal, its architecture of the basic system and some core functions of the system. Besides those core functions described above, there are some more advantages of the infoAsset Broker:

- Skill Management and Document Management are integrated and linked to each other, it is easily enough to traverse from topics to documents and know-how porters.
- With the uniform search, different information objects can be searched from file systems, databases, content management systems and discussion forums at the same time.
- Departments, project teams and knowledge groups can define their own areas in the infoAsset Broker and administrate the areas by themselves.
- Part of the infoAsset Broker can be activated for external users, for example corporate partners, supplier, etc.
- Selected contents can also be transferred offline to CD-ROM for customers.
- The open API (Application Programming Interface) allows the great possibility of extensibility.

As mentioned in chapter two, key advantages of the infoAsset Broker for applying to this project are the Unified Link Management, the visualised navigation system upon KnowledgeMap, personalised notification service, access controlling on document level, possibility of easy integration into other systems and the great possibility of extensibility.

Provided the open extensibility, the infoAsset Broker will be modified for integrating into GovernmentGateway over Web Service Interface. The development of this project will be introduced in detail in the coming chapter four, whereas the detailed describe over the extension of Web Service interface will be found in the Bachelor Thesis from Helge Klimek [Kli05].

# Chapter 4

# Design and Implementation of the Web Service Development Portal

In this chapter the whole development process of this project will be covered and explained in detail.

The first part of the development process is the requirements analyzing. The user requirements and system requirements will be defined. A general overview of the GovernmentGateway architecture will be given here, so that a final vision of this project can be described. According to these requirements, an Use Case Diagram will be introduced and explained.

In the second part the design depends on requirements will be given, this includes the document model, document state model, document publishment workflow according to the Use Case, the extended notification service and the authorisation concept for accessing documents.

How these designs are realised, will be covered in the last part. Modification on the infoAsset Broker basic system will be introduced. Integration of the document authorisation into the infoAsset Broker will be explained. And at last, the integration of the Web Services Interfaces will be shortly briefed and some references of more details will be given. An evaluation of the implemented portal will close this chapter.

## 4.1 Analysis of the project

The GovernmentGateway provides a central access to the Management-Softwares from the company Dataport. The basic idea is to offer a centralised infrastructure, so that different methods can be used from a centralised point. In this analysis section, the architecture of the GovernmentGateway will be shortly briefed, followed by a detailed description over the identified Use Cases in the Web Services Development Portal.

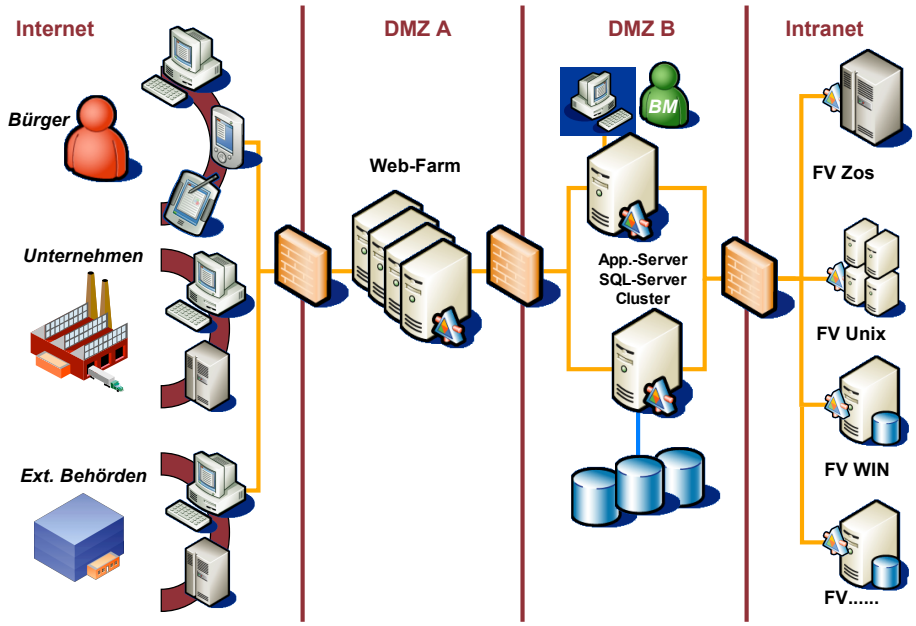


Figure 4.1: Architecture of GovernmentGateway [Dat04]

### 4.1.1 General overview of the architecture of the GovernmentGateway

Most of the civil services are based on different applications on different operation systems. The idea of the GovernmentGateway is to make the existing applications reusable for the E-Government. In order to realise this idea, the standard Web Services technologies like SOAP/XML are used for accessing those applications without changing the existing processes. The concept can be adopted under Windows, UNIX and z/OS operation systems.

Another character of the GovernmentGateway architecture is the asynchronous connection to the case procedures. Data objects do not need to be mirrored or synchronized, which not only increases the security level, but also makes the maintenance of the back-end easier. The citizen becomes the response normally right after the request is sent. If the service is not available or the request is being processed in the processing cue, E-Mail will be sent later to the customers when the result becomes available. The customers can retrieve the result from their mailbox at the GovernmentGateway over a secured SSL (Secured Socket Layer)-connection at any time.

As we can see in figure 4.1, the GovernmentGateway has adopted a layered architecture. The presentation layer can be accessed over the Internet with almost all kinds of end devices, customers here could be normal citizens, corporations and other public authorities. Between the first and second firewall is the area of DMZ (Demilitarized Zone)-A, in which all the Web Servers for the presentation layer are located. The application layer combined with database layer reside in the DMZ-B between the second and the third firewall, interfaces, basic Gateway functions, business logics, case-dependent procedures and database-cluster are located in this layer. The last layer is behind the third firewall, in which all the back-end case-dependent procedures can be found.

An advantage of the GovernmentGateway is that the customers can access all the Internet services from the cities, communes and communities at one point with one design. The customer need only register for the GovernmentGateway once, and then can access all the services that are relevant.

The result of this project, the Web Service Development Portal, will be settled in the back-end area, as one of the case-dependent procedures. As mentioned in the last chapters, the access to this case-dependent procedure will be realised over Web Services (SOAP Calls).

### 4.1.2 Use cases of the Web Service Development Portal

Use Case is a method for describing potential system requirements. Each use case provides one or more scenarios that define how the system should interact with the end user or another system to achieve a specific business goal.

Before creating Use Cases of the Web Service Development Portal, some basic actor groups are identified:

- Basic Architecture Developer, who provide resources such as binary files or documentation for Gateway Procedure Developers (Developers that are responsible for case dependent procedure of the GovernmentGateway).
- Gateway Procedure Developer, who provide resources like know-how for other Gateway Procedure Developer.
- Web Services Standard Developer, who publish Standards, Styles and write Web Services.
- Web Services Developer, are those internal developers, who write Web Services and also provide documentation, know-how and source codes.
- Web Services Provider, who provide Web Services interfaces. Other developers can write application depends on the published Web Services interfaces.

Part of these user group (Developers) are Client Developers from corporations, they use Web Services for retrieving data from the Government and process data for further purposes.

These basic actor groups can be generalised into two Use Case actors, Document-Provider and Document-Subscriber. The functional differences between the basic actor groups will be differed through the User Roles. The generalisation makes it possible to leave the user group structure flat, therefore the extensibility is provided.

Now let us take a look at the Use Case Diagram given in figure 4.2.

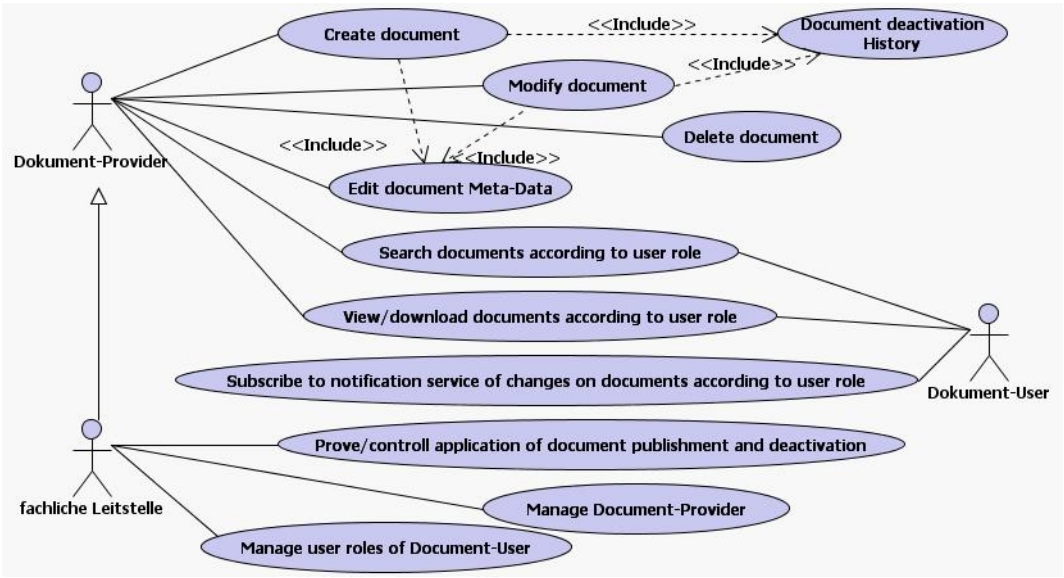


Figure 4.2: Use cases of the Web Service Development Portal

**Document Feeding**
Document-Provider creates new documents, in which the target user groups will be defined, so that only the target user who owns the specified user roles in the target user groups can access the documents. The special user group “Fachliche Leitstelle”will prove the new documents before publishing.

In order to enrich the possibility of document retrieving, some methods are used in the infoAsset Broker: documents will always be provided with meta-data, keywords can be associated to documents, documents will be full-text indexed, documents can be associated with other documents, directories can be created for containing more documents.

**Document Modification/Updating, Deactivation**
Document-Provider who feeds new documents into the infoAsset Broker, is the owner of the documents, which means that he can edit the documents (creates a new version of the documents with the modification) later. As rules defined, publishing a new version of document will be proved by the “Fachliche Leitstelle”.

Since the documents should not be physically deleted from the system, documents that are out-of-date can be deactivated. Deactivated documents can still be found through the extended search function. Document-Provider (document owner) can apply for the deactivation, which will have to be adjusted by the “Fachliche Leitstelle”.

**Historical Document**
Modification on document results in publishing a new version of the document, the meta-data will be overtaken and the old version will be moved to the history-directory of the document. The new version of the document will be indicated with a “Current”tag and always be shown at the first, vice versa, the old version will be indicated with a “Historical”tag and can only be

retrieved over extended search function.

**Document Retrieving**
Document can be retrieved through the integrated search function or over the Knowledge-Map. With the integrated search function, full-text indexed document can be searched through, document can also be searched after meta-data. By traversing through the Knowledge-Map, semantically associated documents can be found. Depending on user roles, search result will be filtered.

**Using Document**
After retrieving the document, document information can be read online, attached files can be downloaded.

**Receiving notification on modification**
User can subscribe to the Notification Service of the document, so that if any modification is made to the document, the user can be notified.

**Managing Document-Provider**
Document-Provider will access the infoAsset Broker directly at the back-end. The special user group “Fachliche Leitstelle”manages the Document-Provider and grant them certain user roles.

**Managing User Roles from Document-Subscriber**
Document-Subscriber access the infoAsset Broker over the GovernmentGateway, the User Roles owned by the Document-Subscriber will be taken along. Those User Roles will also be managed by the “Fachliche Leitstelle”.

## 4.2 Design for the Development Portal

According to the requirements of the project and analysis over the Use Cases, certain design should be made based on the infoAsset Broker basic system, which include document model, document state model, workflow of document publishment, extended notification service and document authorisation concept. In this section, these designs will be described in detail.

### 4.2.1 Document Model

The Information Asset from the infoAsset Broker basic system has the following key characters (following text are based on [Weg02]):

- Every Asset has one AssetType, which is not changeable after creation. The AssetType will identified through its name, for example “Document”.
- Every Asset has its unique AssetId after creation. The AssetId will be used as a primary key in the relational model and will be generated random. An Asset is therefore system-wide identifiable through the combination of AssetType and AssetId, for example “document/klain3siam9”.

- Every Asset has a changeable name besides the AssetId. For Example a document will be identified through its title.
- Besides default Attributes like “Name”, Asset has certain other Attributes and Relationships to other AssetTypes, which are defined by its AssetType and Conceptual Model.
- Every Asset has an Attribute that indicates the last modification timestamp.

For managing Assets in the infoAsset Broker, AssetContainer are used for every type of Asset. Functions provided by AssetContainer are creation, indentification, existence-proving and destroying of Assets of each AssetType. The concept of Asset, AssetType and AssetContainer will be shown in the figure 4.3.

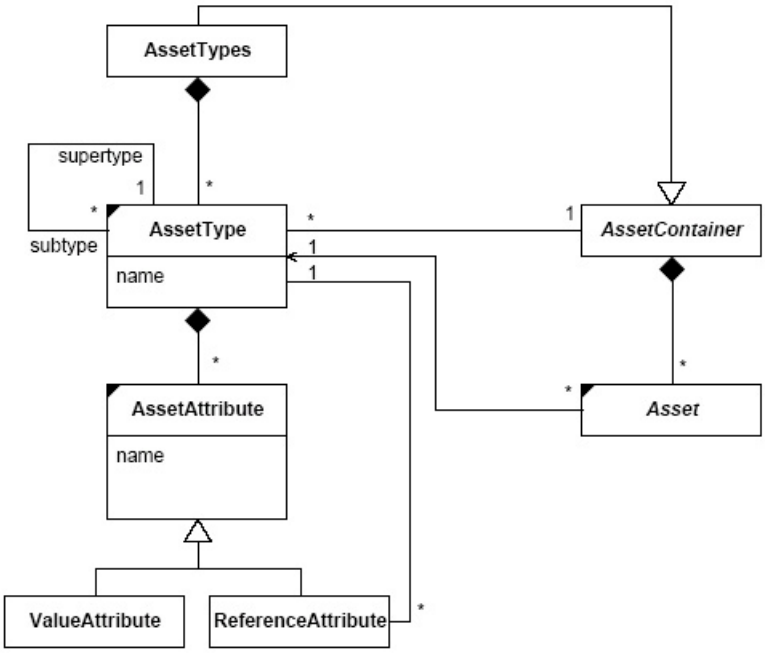


Figure 4.3: Class Diagram Asset, AssetType [Weg02]

The main Asset being dealt with in this project is the Document Asset. The “Document”here has a general meaning, it can represent Web Service Documents, XML (eXtensible Markup Language) Documents, Binary Files, etc. The infoAsset Broker basic system provides some basic attributes for the Document Asset (only relevant attributes will be described here):

- Name, a string value that can be given freely under the Broker naming rules, it will be used partially to identify the document.
- Title, a string value that represent the document title, if it is given.
- DocumentKind, a string value within the predefined domain. It indicates which kind of document the Asset Object is representing.
- DocumentStatus, a string value that shows the state of the document. These are private, intern and public.
- Comment, a string value for adding short notice to the document.

- Concept, this reference attribute references to Concept Assets that build up part of the Knowledge-Map.
- ExpiryDate, a date value that indicates the expiry date of the document if it is given.
- Directory, a reference to a Directory Asset, in which the document is included.
- Filename, a string value for name of the file, which is attached to the document.
- LocalFilename, a string value that represents the full relative path of the file attached to the document on the infoAsset Broker server.
- Version, a string value that shows the version of this document.
- ParentVersion, a reference attribute to another Document Asset, which is the parent version of the document.
- SuccessorVersion, a reference attribute to another Document Asset, which is the successor version of the document.
- Creator, a reference attribute to a Person Asset, who creates the document.
- LastEdior, a reference attribute to a Person Asset, who edits the document last time.
- LastEditedDate, a date value indicates date of last modification.
- LockedBy, a reference attribute to a Person Asset, who is modifying the document and setting the write lock on the document.

The Class Diagram of the basic Document Asset is shown in figure 4.4.

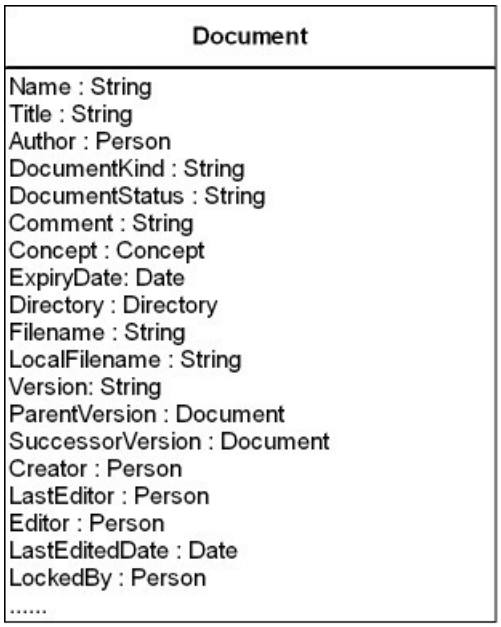


Figure 4.4: Class Diagram of basic Document Asset

According to the core functions of the Web Services Development Portal, a new design of

Document Model based on the basic Document Asset Model in the infoAsset Broker is made.

For the purpose of timed publishing control of the document, some new date attributes are added. The document state will be changed automatically upon those date attributes, if they are given. Those new date value attributes are:

- ValidFromDate
- ReleasedDate
- PublishmentDate

How the document state will interact with these date attributes, will be explained in the following part 4.2.2.

WSDL 2.0 [Con05a] introduces its new Component Model, those Components are interfaces, bindings and services. Thus, a WSDL document can be combined with component documents, which means, documents are related to each other. Such kind of document relationship is realised by two new reference attributes:

- UsedDocument
- UsedByDocument

Through this reference possibility, Web Service components can be easily navigated.

Some new reference attributes to Person Asset are added as well. So that not only documents can be crosslinked to the Knowledge-Map, but also more persons related to the documents can be crosslinked. These reference attributes are:

- Owner
- ProfessionalContactPerson
- TechnicalContactPerson
- ReleasedBy

Documents can be referenced to Concept Asset, which is used to categorise document and enrich the Knowledge-Map. Since there are also different types of Concept Assets needed according the requirements for this Web Service Development Portal. Concept Asset will be categorised into four types, and will be separately referenced by these new attributes:

- Category
- LegalFoundament
- ProfessionalContext
- TargetGroup

Another very important new attribute is the “HistoryDirectory”reference attribute, which references a Directory Asset. Old versions of documents will not be listed together with the current version, according to the requirement, therefore, a directory called “History Directory”is provided for saving and tracking old versions of the document.

Additional attributes are:
Intention, a string value for describing document intention.
DiffPreVersion, a string value for describing differences between this document and the parent version of the document.
ReasonOnExpiry, a sting value for explaining reasons, if a document is supposed to be turned expired before its original expiry date.
URL, a string value for saving URL of the document.
TestURL, a string value for saving test URL of the document.
Availability, a string value for describing the availability of the document.
HistoryStatus, a boolean value that indicates if the document locates in the history directory.

The new Document Model is provided in figure 4.5:


Figure 4.5: New Class Diagram of Document Asset

### 4.2.2 Document State Model

The old Document State Model with three states (private, intern and public) is not sufficient enough to describe the interaction between the document states and the document publishment date attribute, since the “timed-publishment”function is required by the company Dataport. A new Document State Model is therefore defined as shown in figure 4.6.

The State Diagram describes the states that the document can reach during its existence. When the document is created, its initial state is “Unpublished”. At this state, the creator of the document can edit the document freely and can also delete the document if it is needed. After the creator of the document finishes editing the document and applies for the publishment at the “Fachliche Leitstelle”, the document reaches the state “Applied for publishment”.

If the document is denied for publishment, the document returns back to the state “Unpublished”, so that the creator can edit the document again. Otherwise if the document is accepted for publishment, the successive state could be “Waiting for publishment”or “Published”. It depends on the document attribute “PublishmentDate”, which state the document will reach. If the given publishment date is in the future, the document reaches “Waiting for publishment”and will not be published until the publishment date reaches, vice versa, the document will get the state “Published”if the publishment date is already reached.

Document with the state “Published”can be applied for deactivation, which means, to be located in the history directory. The document reaches the state “Applied for deactivation”, if this is applied by the creator of the document at the “Fachliche Leitstelle”. If the answer from the “Fachliche Leitstelle”is positive, the document will reach the state “Deactivated / History”. Otherwise it will return to the state “Published”.

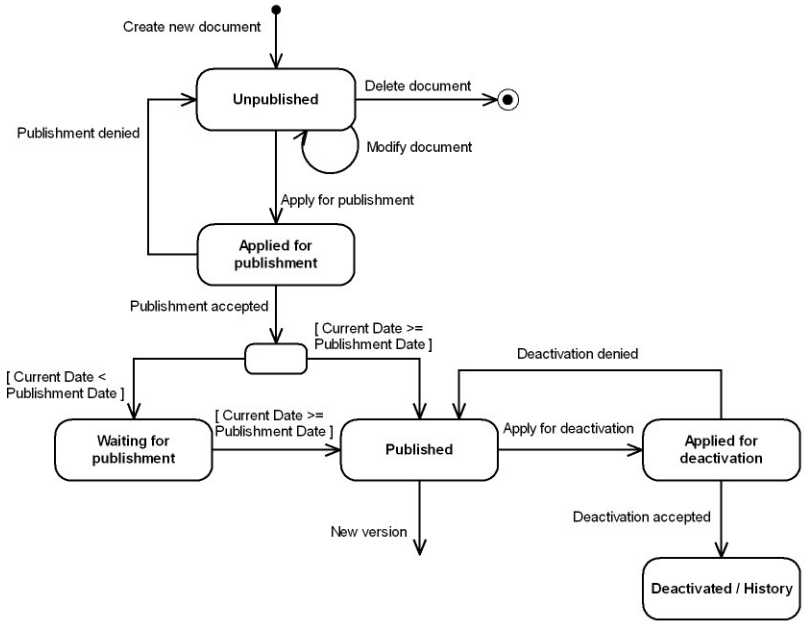


Figure 4.6: State Diagram for Document State Model, part 1

New version of a “Published”document can be created by the document creator. Once the new version is created, it has the state “Unpublished”and will exist together with the old version. In the case that the new version is accepted by the “Fachliche Leitstelle”, the document attribute “PublishmentDate”will be used again to decide the states of both new and old versions. If the publishment date is already reached, the new version becomes the state “Published”and the old version will be set to “Deactivated / History”automatically at the same time. If the publishment date is still in the future, the new version reaches the state “Waiting for publishment”and the old version keeps the state“Published”. A parallel State Diagram for this case is shown in figure 4.7:

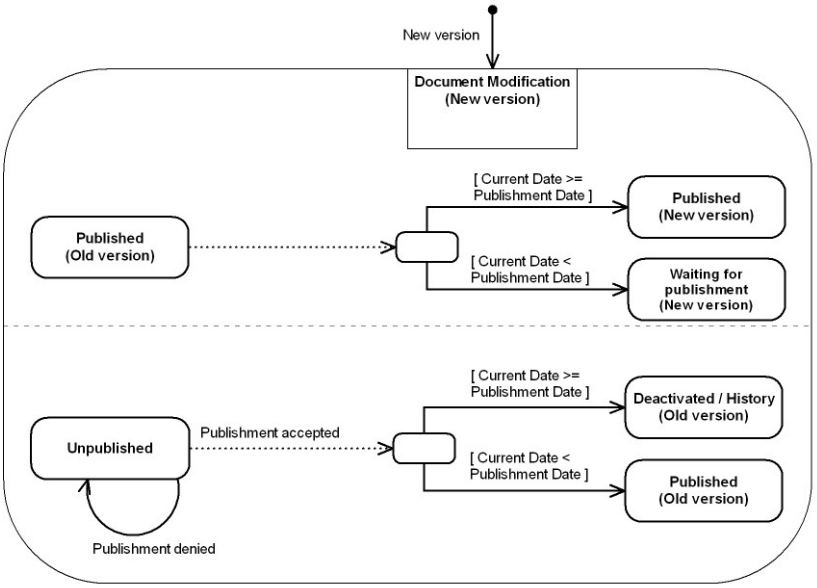


Figure 4.7: State Diagram for Document State Model, part 2

### 4.2.3 Document Publishment Workflow

Since a control function (“Fachliche Leitstelle”) is required in the Web Service Development Portal for the company Dataport, a new workflow for document feeding, modification, deletion and deactivation in the infoAsset Broker will be modeled.

The main difference is that the documents will not be published directly from the creator of the document, but from the “Fachliche Leitstelle”, who is assigned to prove those documents supposed to be published. The “Fachliche Leitstelle”will prove, if the content of the document is qualified, if all the legal requirements are covered and if the valid time of the document is given correctly. If there is no problem with the document, the “Fachliche Leitstelle”will release the document for publishment. Otherwise, an online text message in the infoAsset Broker will be sent to the document creator by the “Fachliche Leitstelle”, a reason why the document can not be released for publishment will be given briefly in the message. Upon the returned message, the document creator can modify the document and try to apply for the publishment again later. Same control workflow happens, when a new modified version of a document is applied for

publishment or a document is applied for deactivation. The deactivation of the document is required specially from the company Dataport, since the old documents should still be able to be traced, so there is no deletion allowed for the published documents in this Web Service Development Portal, hence, deactivated document will be moved to the history directory instead of being physically deleted. The workflow for this “Fachliche Leitstelle”function is shown in the figure 4.8:

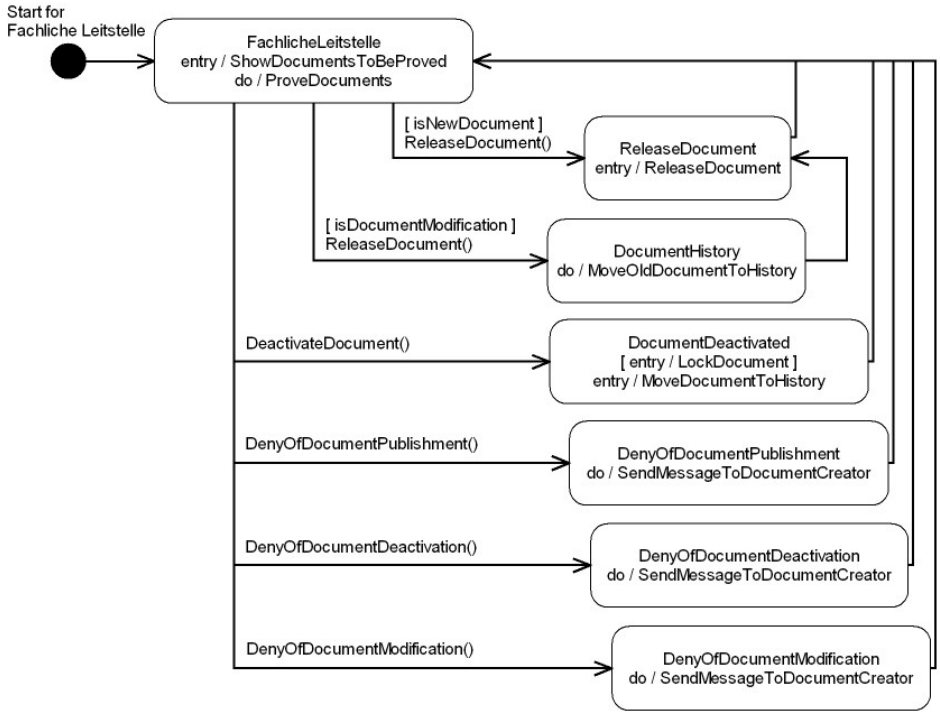


Figure 4.8: Activity Diagram for Workflow of “Fachliche Leitstelle”

For normal infoAsset Broker user, the entry point of the Broker is a list of document directory, to which the user has the access right. This is another requirement of the company Dataport, which is, user cannot browse anything they do not have right to access. How this Authorisation Concept is modeled, please see the coming part 4.2.5.

Broker user can create document in authorised directory. After inputting sufficient meta-data of the document, target groups who are supposed to have read access to the document, should be granted the access by the creator. As mentioned before, the creator has to apply for the publishment of the document at the “Fachliche Leitstelle”at the end.

Document will be write-locked automatically, if the creator of the document starts to modify the document. If the document is still unpublished, the creator can continue edit the document information or delete the document. If the document is published, a new version of the document will be created automatically, and all the meta-data and description information of the old version will be taken over, the creator will work on the new version for the modification. After the modification, the new version can be applied for publishment.

The workflow for normal infoAssetBroker user is shown in figure 4.9:

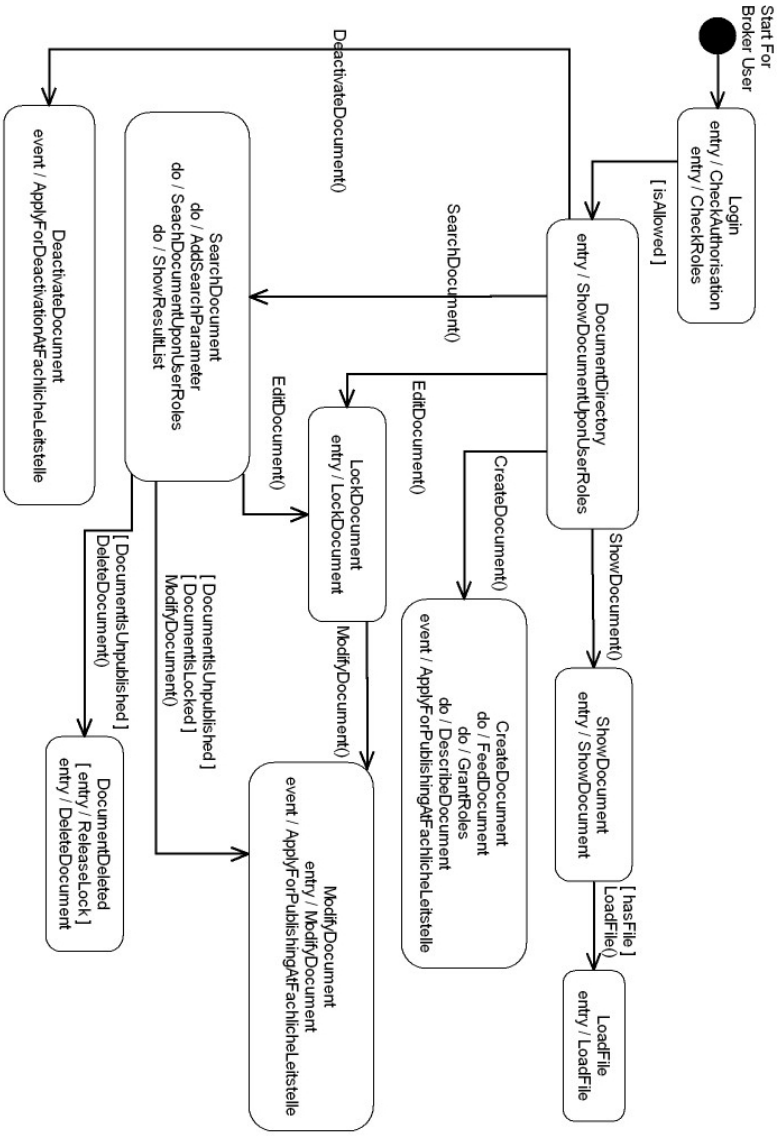


Figure 4.9: Activity Diagram for Workflow of normal infoAsset Broker user

### 4.2.4 Extended Notification Service

The infoAsset Broker basic system brings the subscription and alerting/notification service. Users will be notified (via online report or E-Mail) of changes on documents they have subscribed to. As i introduced in section 4.2.1, WSDL‘s Component Model enables the reusability of seperate components. This could, though, raise a potential fragility of Web Service-based application because of the increasing dependencies on used components as well as on called Web Services [ST05]. A new version of an existing component document or service often replaces the old version rather than leaving two versions of the services operational simulta-

neously. According to the Dataport‘s requirement, the old version of an existing document will not be removed from the repository. Suppose that a user used a seperately defined WSDL interface component, or a defined WSDL for a Web Service. If a new version of the document is released, there will not be a guarantee that the new version of component or service will be perfectly matched in the existing client implementation.

For dealing with such kind of problem, the subscription and alerting/notification service as well as the versioning function of the infoAsset Broker are used. Documents in the portal will always have a version number, if new versions exist. Users will be automatically subscribed to the documents they have used, so that they will be kept informed with any changes on the documents with the document version number included. This enables the information-push instead of information-pull. The document users then can decide on their own, if they want to use the new version of the document. In order to keep an overview of the document dependency for the “Fachliche Leitstelle”, a dependency information will also be provided, if a new version of an existing document is applied for publishment.

### 4.2.5 Authorisation Concept

The access control of the document in the infoAsset Broker basic system is based on Authorisation Roles of Group Document Directory. Every Group Document Directory has four Authorisation Roles:
**ReaderRole**
Role that is allowed to read public documents in the directory
**WriterRole**
Role that is allowed to write documents in the directory and read non-public documents
**EditorRole**
Role that is allowed to modify Meta-Data of the directory and create or delete subdirectories of the directory
**AuthorRole**
Role that is allowed to create new documents within the directory and to change documents created by themselves

User group in the infoAsset Broker is used to arrange a number of users, who own different gourp memberships of the group. Each user group can has one group document directory, whereas users of the user group obtain the Authorisation-Roles of the group document directory are assigned with group memberships. Only so can user get access to documents or create documents in the group document directory. This basic concept of infoAsset Broker is shown in figure 4.10:

As introduced in the requirements and core functions of the Web Service Development Portal in chapter two, the Authorisation Concept should base on document level. There are two types of users who can get access to the portal. On one hand, users (Gateway-User) access the portal content over the GovernmentGateway only with Read-Access and only to those documents they are allowed to access. On the other hand, users (Broker-User) access the portal over the intranet in the company Dataport. Combined with the Use Cases described earlier in this chapter, three authorisation cases can be defined and modeled based on the basic Authorisation Concept:

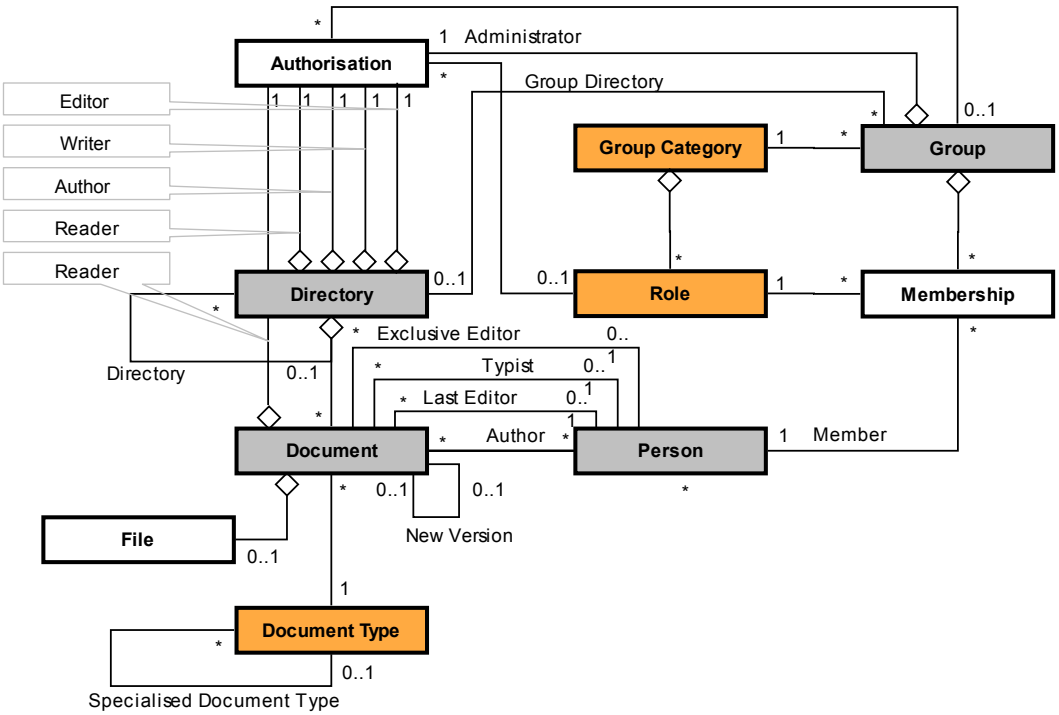


Figure 4.10: (Part of the) Conceptual Datenmodel in the infoAsset Broker [Mat03]

**Authorisation over Group-Directory for Broker-User**
Broker-Users in a group, who have the membership “Developer”, own AuthorRole and WriterRole, so that they can create documents in the group document directory, edit documents created by themselves and read published documents created by other developer in the group.

**Authorisation over “TargetGroup”Membership for Broker-User**
As mentioned in the new Document Model, a new type of Concept Asset is defined, which is called “TargetGroup”. This reference attribute of the Document Asset is used to grant ReaderRole to portal users, which therefore realises the authorisation at document level. Both the author of the document and the “Fachliche Leitstelle”of the group document directory can modify this reference attribute, in order to grant ReaderRole to other portal users. In this case, the ReaderRole can be granted to other Broker-User from other group. Through this reference, those invited Broker-User will receive the membership “TargetGroup”of the group document directory, where the document is located, and as a result, they own the ReaderRole of the group document directory and have the read access to dedicated documents.

**Authrisation over “TargetGroup”Membership for Gateway-User**
Gateway-Users have only Read-Access to documents they are allowed to access, so they only need a ReaderRole for those directories where dedicated documents are located. Gateway-Users access the portal over the GovernmentGateway, but they will also have their own group defined in the portal. Therefore, Gateway-Users can be treated easily like Broker-Users in case two above. The author of the document and the “Fachliche Leitstelle”of the Group-Directory can add “TargetGroup”Concept references for those Gateway-User-Groups, so that

invited Gateway-Users will own membership “TargetGroup”of the group document directory and the ReaderRole as well. They can then have the read access to those dedicated documents.

The idea of this realisation, that a new type of Concept is used to reference the Document with the Read-Access, is using the Knowledge-Map for tracing and retrieving granted Read-Access. It also establishes a basis for the Web Service Interface of the Development Portal, through which a list of accessable documents can be generated easily and transferred over Web Service to the Gateway-Users. Details on how this is realised, please read the Bachelor Thesis from Helge Klimek [Kli05].

## 4.3 Implementation of the Development Portal

Now let us see how these designs are implemented in the real world. Short describe on Java Class Methods will be made and some screenshots from the Development Portal will be shown in this part of the chapter.

### 4.3.1 Modification in the InfoAsset Broker

First of all, some modification on the Document Asset according to the Design will be shown bellow, a complete list of new methods will be provided in the appendix.

Two of the new methods in the Class IMPDocument that implements Document interface are for example:

```
0 public String getReasonOnExpiry() {
     return this.getProperty(Documents.asReasonOnExpiry);
  }

  public void setReasonOnExpiry(String reasonOnExpiry) {
5    this.setProperty(Documents.asReasonOnExpiry, reasonOnExpiry);
  }
```

getReasonOnExpiry() gets the reason for moving the document into the history directory (de-activation) before the defined expiry date.

setReasonOnExpiry(String reasonOnExpiry) sets the reason for moving the document into the history directory (deactivation) before the defined expiry date.

These two methods are part of the fundamental getter and setter methods for those new String Attributes (see 4.2.1 Document Model), other methods can be found in the appendix.

Two of the new getters and setters for Date Attributes in this Class are for example:

```
0 public Date getFreigegebenDate() {
     return this.getProperty(Documents.asFreigegebenDate);
  }

  public void setFreigegebenDate(Date freigegebenDate) {
5    this.setProperty(Documents.asFreigegebenDate, freigegebenDate);
  }
```

getFreigegebenDate() returns the date, when the “Fachliche Leitstelle”has accepted the publishment of the document.

setFreigegebenDate(Date freigegebenDate) sets the date, when the “Fachliche Leitstelle”has accepted the publishment of the document.

One of the new methods for referencing new Concept types in the Class IMPDocument is:

```
public Iterator getZielgruppen() {
   HashSet zielgruppen = new HashSet();
   Iterator allConcepts = services.getConcepts().getConceptsOfDocumentIds(getId());
   while (allConcepts.hasNext()) {
      Concept thisConcept = services.getConcepts().getConcept((String)allConcepts.
          next());
      for (Concept c = thisConcept.getSuper(); c != null; c = c.getSuper()) {
         String cName = c.getName();
         if ("Zielgruppen".equals(cName)) {
            zielgruppen.add(thisConcept.getId());
            break;
         }
      }
   }
   return zielgruppen.iterator();
}
```

getZielgruppen() returns all Concept Asset ids with the type of “TargetGroup”, which are referenced to the document, as an iterator. In this method, all referenced concepts of the document will be retrieved over the concept service. Every retrieved concept will be proved if one of its super concept is “TargetGroup”. All concepts that have “TargetGroup”as one of their super concept will be collected in a hash-set and be returned as an iterator.

To reference a Concept Asset to the document, the old Method addConceptId (String conceptId) will still be used, since all types of the Concept are Concept Assets, and they will only be classified when these Concept references will be retrieved as shown in the example above.

Now we come to some new methods in the IMPDocument Class, which are used to build up references between documents, so that related documents are linked and can be retrieved in both direction. One of these methods is as follows:

```
public void addUsedByWSDocument(String wsDocId) {
   Document usedByWSDocument = services.getDocuments().getDocument(wsDocId);
   if (!this.hasAssociatedAsset
   (Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole(), usedByWSDocument)) {
      this.createAssociation
      (Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole(),
      usedByWSDocument,
      Documents.DOCUMENT_USEDBYWSDOCUMENT.getThatRole());
   }
   if (!this.hasAssociatedAsset
   (Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole(), usedByWSDocument)) {
      this.createAssociation
      (Documents.DOCUMENT_USEDWSDOCUMENT.getThatRole(),
      usedByWSDocument,
      Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole());
   }
}
```

addUsedByWSDocument(String wsDocId) adds two new references. One is created in the “document_ usedbywsdocument”reference table, another is created in the “document_ usedws-

document”reference table. These two references build up the bi-directional relation between Web Service documents, which will be used for the Knowledge-Map.

Who are the Knowledge-Porters of the document? Who has released the document for publication? Those are important questions according to the Dataport requirement, so that the relationships between Persons and Documents can be traced. One of the methods for those new Attributes introduced in 4.2.1 is:

```
public void addAnsprechpartnerFachlichId(String id) {
   Person ansprechpartnerFachlich = services.getPersons().getPerson(id);
   If (!this.hasAssociatedAsset
   (Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH
   .getThisRole(), ansprechpartnerFachlich)) {
      this.createAssociation
      (Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH
      .getThisRole(), ansprechpartnerFachlich,
      Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.getThatRole());
   }
}
```

addAnsprechpartnerFachlichId(String id) adds a Person as one of the Professional Contact Persons to the current Document. A new reference in the “document_ person_ ansprechpartnerfachlich”reference table will be created, so that this relation can be traced.

Another example is:

```
public void setFreigegebenVon(String id) {
   Person freigegebenVon = services.getPersons().getPerson(id);
   this.createAssociation
   (Documents.PERSON_DOCUMENT_FREIGEGEBENVON.getManyRole(),
   freigegebenVon, Documents.PERSON_DOCUMENT_FREIGEGEBENVON.getOneRole());
}
```

setFreigegebenVon (String id) adds a reference to the Person (the responsible “Fachliche Leitstelle”) who releases the current Document for publishment.
A new reference in the “document_ freigegebenvon”reference table will be created, so that this relation can be traced.

### 4.3.2 User Interface of Document Publishment Workflow

The implementation for extending Document Status from 3 States to 6 States is done in the Class ChangePublicationStateHandler. Changes of Document States are done in the Business Logic and these changes depend on the entire Workflow in the infoAsset Broker. I will now introduce part of the Workflow using the following figures.


Figure 4.11: New document created with “Unpublished”Document State

After a new document is created and all the necessary meta-data and description are filled, the document will have the state “Unpublished”. As shown in figure 4.11, the hyperlink right beside the document state is used by the user for applying the publishment of the new document. A click on this hyperlink will trigger a Document State Change from “Unpublished”to “Applied for publishment”, see figure 4.12:


Figure 4.12: New document with the Document State “Applied for publishment”

At the same time, the “Fachliche Leitstelle”of the Group Directory will be automatically subscribed to the new document, so that the “Fachliche Leitstelle”will be informed of the application of document publishment over the notification service. Figure 4.13 shows this event:


Figure 4.13: “Fachliche Leitstelle”receives a new notification

In the notification, information on event object, object type, event action, event time and actor will be shown, see figure 4.14. In the case of a new document being applied for publishment, document name, document version and document author will be listed:


Figure 4.14: Notification on new document applied for publishment

Over the hyperlink of the document, the “Fachliche Leitstelle”can navigate to the detail page of the new document. The “Fachliche Leitstelle”will check the information of the new document and decide, whether the publishment of the new document should be allowed. Two new hyperlinks for the next Document State Change will be provided beside the document state, as you can see in figure 4.15:


Figure 4.15: Detail page of the new document viewed by “Fachliche Leitstelle”

If the document is not well described or mistakes are found in the document, the “Fachliche Leitstelle”can deny the application of the document publishment. In this way, the “Fachliche Leitstelle”must provide the document author a reason why the publishment is denied:

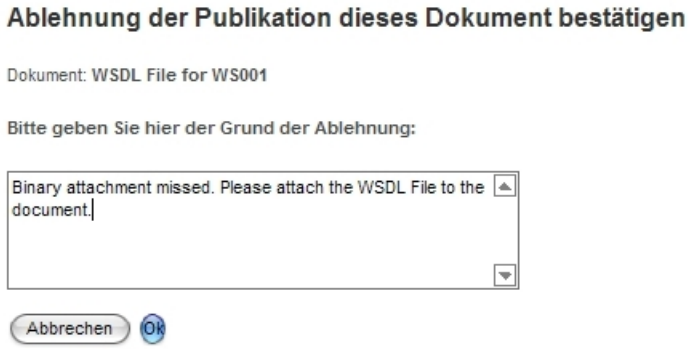


Figure 4.16: Denial reason of document publishment by the “Fachliche Leitstelle”

After that, the document state will be changed back to “Unpublished”, which then allows the document author edit the document again. The document author will receive a notification, which indicates that the document publishment is denied with the attached reason.

In other way, if the document fulfills all the requirements, the “Fachliche Leitstelle”will accept the application of document publishment. The document author will receive a notification, which informs about the publishment of the new document. At the mean time, the document state will be changed to “Published”:


Figure 4.17: Document published

But remember, there is another document state that is called “Waiting for publishment”. After the document publishment is accepted, the defined publishment date will be checked. If the date given is still in the future, the document state will be directly changed to “Waiting for publishment”(figure 4.18), and will be published automatically, when the given date is reached.


Figure 4.18: Document with State “Waiting for publishment”

The implementation of Document State Change from “Published”to “Deactivated / History”is realised in the same way and in the same Class, so i will not describe it here.

### 4.3.3 Integration of the document access authorisation in the InfoAsset Broker

In the basic system of the infoAsset Broker, document access is at document directory level, which means, if the user has the read right to the document directory, and then he can read all the documents in the directory.

In order to change the document access to be controlled at document level, we first defined three memberships in a group, which are “Fachliche Leitstelle”, “Developer”and “Target-Group”. The access control at the group document directory level will still be used, user who owns “Fachliche Leitstelle”membership will have the right to edit the meta-data of the group document directory, user who owns “Developer”membership will be able to create, modify or deactivate documents in the group document directory (those actions will have to be proved by the “Fachliche Leitstelle”for sure), and finally, user who has “TargetGroup”membership will only have the right to view the directory and read those documents that are selected to be accessed by the user. The membership “TargetGroup”is used to grant read access, users who receive this membership are invited to read certain documents, they can be considered as a virtually gathered member group.

How can user be able to own a “TargetGroup”membership in that group? This is realised through the new concept type “TargetGroup”. When one group in the infoAsset Broker is created, one and only one concept in type of “TargetGroup”will also be created with the same group name as the concept name, this new concept will be associated with the new group. Every document can be referenced to one or more concepts with the type “TargetGroup”. Let us now see one example: user A is a developer in group GA, user B is a developer in group GB. User A has published a document in group directory DA, the document was accepted by the “Fachliche Leitstelle”. Now as “TargetGroup”for the document, group GB is referenced, so every developer (those who have the membership “Developer”) in group GB will become a new member in group GA with the membership “TargetGroup”.
After receiving the “TargetGroup”membership, how can those users read certain documents? The answer is easy. Let us see another example: user A is a developer in group GA, but also a member with “TargetGroup”membership in group GB. Document D1 is published in the group document directory of group GB, it has a “TargetGroup”concept reference to the group GA, which enables user A to have the read access to this document. Document D2 is published in the group document directory of group GB as well, but without “TargetGroup”concept reference to the group GA, which ensures that user A will not be able to view the document.

In order to reduce the cost of time, the “Fachliche Leitstelle”of the group can still modify the “TargetGroup”reference of the documents even after the document is published, so that the read access of documents can be granted dynamically.

User‘s “TargetGroup”membership will be revoked, if there is not more documents in the group directory, which has a reference to user‘s original group, where the user has the “Developer”membership.

### 4.3.4 Integration of a Web Service interface for the infoAsset Broker

The integration of a Web Service for the infoAsset Broker is realised with Axis and Jetty.

Apache Axis [Fou05] is an open source Web Service framework that is an implementation of the SOAP. It is used for generating and deploying Web Services applications. In this project, the Axis will be extended, and will be reduced mainly for dealing with Web Service request and response. The infoAsset Broker will still manage the entire business logic.

Jetty [Con05b] is also an open source product. It is a Java HTTP server and servlet container. Jetty will be integrated into the infoAsset Broker without any modification, the main job for Jetty is to process the Axis Servlet. The integrated Jetty will then be provided as a Broker Service.

The figure 4.19 shows the extended architecture of the infoAsset Broker with Axis and Jetty:

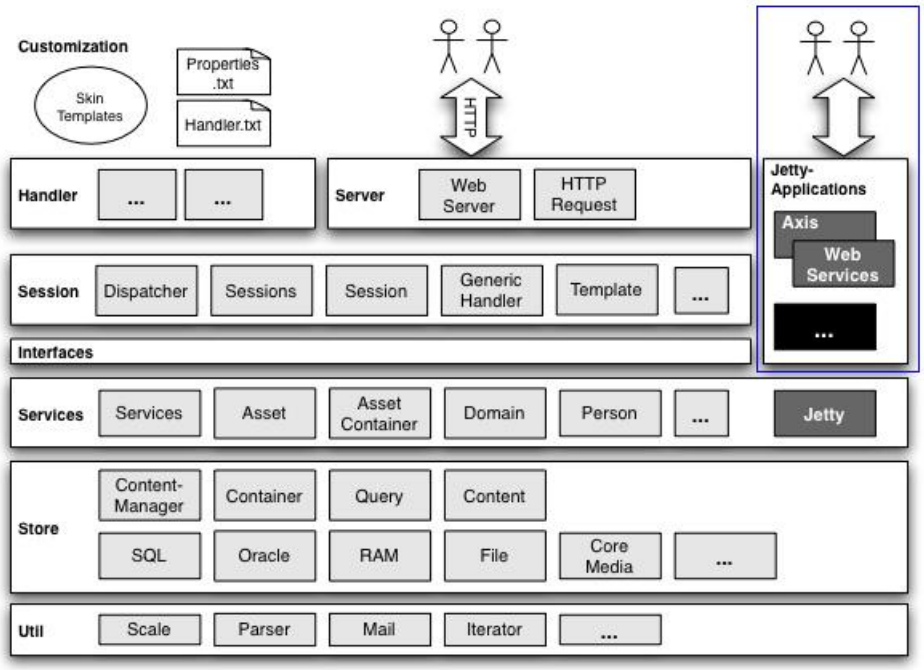


Figure 4.19: Extended architecture of the inforAsset Broker

The entire work design and implementation of this Web Service interface for the infoAsset Broker is as mentioned, to be found in the Bachelor Thesis of Helge Klimek [Kli05], a detailed introduction is beyond the scope of this thesis.

## 4.4 Evaluation of the implementation

In first three subchapters I have introduced you the development process of the project. The implemented Web Service Development Portal is integrated into Dataport‘s back-end, internal

developers of Dataport access the Web Service Development Portal over the Intranet and will use the original frontend of the Portal. External developers of Dataport access the Web Service Development Portal over the Internet and a standardised GovernmentGateway frontend will be provided, through which the external developers can use a tree-view navigation for retrieving documents they are allowed to view and access.

In this subchapter, these two access methods of the Web Service Development Portal will be shortly evaluated.

### 4.4.1 Comparison of the two access environments of the portal

**Access over the intranet**
The access of the Web Service Development Portal over the intranet is relatively simple. Developer accesses the Web Service Development Portal need only a valid user account in the portal. Developers will then be assigned to different groups according to the on-going projects, and normally, they will become the membership as “Developer”in the groups.

After a developer login into the portal, a home directory will be shown, in which all the group document directories that the developer has the right to access will be listed. These listed group document directories can be classified into two types. One type is those groups document directories, in which the developer has the “Developer”membership. In this kind of group document directories, the developer can read all published (visible) documents, create new documents, create new version of his documents and apply for deactivation of his documents. Another type of the group document directories are those directories, in which the developer has the “Target-Group”membership and is only allowed to view (read-access) certain published and dedicated documents.

The administration cost in this case is low, because the access comes from the Intranet within the Dataport and the login system of the portal is enough for the authentication purpose. Developers only have to be assigned membership to dedicated groups, so that they can use all the portal functions and access to documents they are allowed to.

**Access over the GovernmentGateway**
External developers of the Dataport will access the Web Service Development Portal over the Internet and will be authenticated by the GovernmentGateway. After a successful authentication, developer‘s username and membership roles will be carried forward to the portal over the Web Service of the portal. According to developer‘s username and membership role, a tree-view of categorised and dedicated documents will be sent back to the developer over the Web Service. By viewing the documents, external developers can download the files attached to the documents if it is provided, they can also subscribe to documents that they want to be kept informed.

In this case, the administration and maintenance costs is relatively high: Since the external developers‘accounts are managed and authenticated by the GovernmentGateway, a synchronised copy of those accounts has to be provided in the Web Service Development Portal. According to Dataport‘s decision, those accounts will be manually maintained and synchronised,

which accrues a high administration cost. A suggestion to replace this costly management is to build a callback function in the Web Service Development Portal, which checks the carried username and membership role against the account management system at the side of the GovernmentGateway. But a follow-on problem of this suggestion will arise, because since the GovernmentGateway is highly secured, such callback function that will access the account management system may violate the security policy of the GovernmentGateway.

In order to process the subscription function of the Web Service Development Portal, a valid E-Mail address have to be provided. But the question is, should every external developer bring his valid E-Mail address to the GovernmentGateway and this E-Mail address will be carried over the Web Service to the portal? This would mean that the Web Service interface would be extended with one more parameter that is used for carry the E-Mail address. A followed question is, since the Web Service Development Portal is located in the back-end of the whole GovernmentGateway system, is the portal allowed to send notification E-Mails to those external developers who have subscribed to certain documents? If the portal is not allowed to send E-Mails directly, another administration cost will arise, because some other functions should be provided to deal with the E-Mail notifications. A better solution for these two questions could be: external developers‘accounts at the GovernmentGateway will use a valid E-Mail address as the username, which will be carried to the Web Service Development Portal later, so that no extra parameter for E-Mail should be provided in the Web Service interface. For sending E-Mails, an E-Mail server should be provided, which is allowed to send E-Mails from the GovernmentGateway to outside, only the E-Mail server address should be registered to the Web Service Development Portal, because the Web Service Development Portal has an open property for using external E-Mail server.

### 4.4.2 The Authorisation Concept at document level

As introduced in the last chapter, the authorisation concept at document level is realised through the Concept Asset in the infoAsset Broker. Through the subtype of the Concept Asset “TargetGroup”, groups that shall be chosen to have read-access (which are in the sense of target groups) are connected with the dedicated documents.

This implementation has raised somehow the code complicity and certain maintenance costs. The reason why this realisation is chosen is because of the thinking of using the advantages of the Knowledge-Map in the infoAsset Broker. The navigation of the Knowledge-Map in the infoAsset Broker is realised over the Concept Asset. By connecting target user groups and documents through the Concept Asset, a clear list of the TargetGroup-Documents-relations can be provided and navigated, which is certainly helpful for the administration. But the contra point could be that the maintenance cost on keeping the 1:1 relation between groups and their “TargetGroup”concepts.

Another implementation option could be that a direct relation between target groups and dedicated documents should be build. Hence, by choosing target groups for the document, who are allowed to view the documents, a list of existing groups in the Web Service Development Portal will be listed, the document will have a reference attribute for referencing target groups.

## 4.5 Summary

In this chapter I have introduced you the whole development of this project.

It starts with the analyzing the architecture of GovernmentGateway, in its back-end the Web Service Development Portal will be integrated into. The analysis on Use Cases for the Development Portal is a fundamental work for the entire work, all the designs are following those Use Cases, which were discussed together with the company Dataport at the early phase of this project.

At the design phase of the project, a modified Document Model is made to support some new features and enrich the semantic relation between documents. Because of the requirement on a user group (“Fachliche Leitstelle”), who should control the document publishment, a new document publishment workflow is created with the extended document states. The notification service is extended for dealing with the versioning issue. A new authorisation concept is also constructed in order to meet the needs of access control at document level.

Due to the high amount of the source code, only some new Methods in important Classes are introduced. Some screenshots of the Web Service Development Portal are shown, in order to explain the document publishment workflow. The integration of a Web Service interface for the infoAsset Broker is covered in another Bachelor Thesis, so only the extended architecture of the infoAsset Broker is outlined.

There are some remarks during the implementation of the Web Service Development Portal, which will be shortly briefed in the coming chapter.

# Chapter 5

# Summary and Outlook

In this last chapter, a summary of this project work will be outlined first and an outlook will be described in the second part.

## 5.1 Summary

This Bachelor Thesis is about the case study DATAPORT, for which a Web Service Development Portal should be designed and implemented.

The company Dataport has numbers of internal and external developers who are working on Web Services related projects. The need of a central point for publishing and sharing the increasing information objects and documents from those developers becomes the driving force of this project. And since Dataport has various types of platforms, a platform-independent portal software is the best choice for Dataport‘s requirements. For providing this portal for external developers, the Web Service Development Portal will be integrated into the back-end system of the GovernmentGateway through a Web Service interface. In order to retrieve information more effectively, semantic relations and dependencies between documents should be built up in the Web Service Development Portal.

After introducing the project motivation, the Web Service with its standard technologies that are used in Web Services, i.e., SOAP, WSDL and UDDI have been briefly described. This has provided a basic understanding of future vision of this project, since the Web Service Development Portal should not only manage Web Service related documents but should also provide its own Web Service.

Core function requirements for the Web Service Development Portal have been further introduced, so that a standing point has been given and the development of this portal can be proved against the requirements later. Since there are many successful portal platforms that can be chosen to be base platform for this Web Service Development Portal, three options, which are Plumtree Portal Server, Microsoft SharePoint Server and infoAsset Broker, have been introduced and their cons and pros are outlined, according to the requirements.

The infoAsset Broker is chosen to be the base platform for developing the Web Service Devel-

opment Portal. Its architecture, concepts and core functions are described in detail, so that the reason why the infoAsset Broker is chosen to be used for this project is given.

The entire development process of this Web Service Development Portal is covered. The analysis phase of the project includes the introduction of the GovernmentGateway architecture and the analysis of Use Cases of the Web Service Development Portal. The design phase of the project describes the design of the document model, document state model, document publishment workflow, extended notification service and authorisation concept. At the implementation phase, modifications on the infoAsset Broker basic system are introduced and the integration of the Web Service interface are covered.

Two access environments and the realisation of the authorisation concept are finally evaluated, where the administration and maintenance costs are taken in concern and some suggestions are made.

## 5.2 Outlook

Because of time constraint, some implementation parts can not completely realised or could still be discussed, some suggestions can be summerised for the future development:

In the current implementation, relations between “TargetGroup”concepts and user groups are manually maintained, another relation between documents and “TargetGroup”concepts will be established by the document author or the “Fachliche Leitstelle”. Through these two relations, authorisation at document level is realised. But the code complexity of this realisation is high. The relation between “TargetGroup”concepts and user groups could also be generated automatically.

The Web Service is concerned as the future for dealing with interoperability between different systems. The infoAsset Broker basic system used in this project is without a Web Service interface. The Web Service interface for this project purpose is well designed and implemented, a standard Web Service interface should be provided based on this implementation for the infoAsset Broker basic system.

The original idea of the project was building up a UDDI management system. In order to cover all the requirements of this project, the infoAsset Broker is chosen as the base of a Web Service Development Portal, which can manage Web Services related documents and offers more possibilities for retrieving Web Services related documents. In the future, an integrated UDDI management system can be implemented in the infoAsset Broker, since the future UDDI could be more powerful.

In the current version, Web Service documents are feeding manually into the portal, which costs certain working time and might be fed with mistakes. A function should be realised, which can read the input Web Service documents and extracts the meta-data from the documents and creates the documents in the portal automatically.

With this implementation, attachment of the document in the Web Service Development Portal

will be transferred as Base64-Coded message over the Web Service. Therefore, the attachment with binary content will have to be recoded before being send back to the GovernmentGateway, and at the side of GovernmentGateway the recoded content should be transformed back to the original format. These operations raise the processing cost and increase the network loading. A FTP (File Transfer Protocol) Server, instead, can be used at this place. A possible realisation could be that a FTP link to the attachment will be sent back to the user, and the user shall provide necessary authenticate information to retrieve the dedicated attachment over the FTP link.

# Bibliography


[Con03] World Wide Web Consortium. Soap version 1.2 part 1: Messaging Framework [online]. June 2003 [cited 10 August 2005]. Available from: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

[Con04] World Wide Web Consortium. Web Services Glossary [online]. February 2004 [cited 10 August 2005]. Available from: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/.

[Con05a] World Wide Web Consortium. Web Services Description Language (wsdl) version 2.0 [online]. August 2005 [cited 10 August 2005]. Available from: http://www.w3.org/TR/2005/WD-wsdl20-20050803/.

[Con05b] Mort Bay Consulting. About Jetty [online]. 2005 [cited 20 September 2005]. Available from: http://jetty.mortbay.org/jetty/.

[Dat04] Dataport. Dataport GovernmentGateway Präsentation. Dataport Presentation, Dataport, June 2004.

[Dat05] Dataport. About us [online]. 2005 [cited 23 July 2005]. Available from: http://www.dataport.de/.

[Fou05] Apache Software Foundation. Introduction [online]. 2005 [cited 20 September 2005]. Available from: http://ws.apache.org/axis/.

[inf03] infoAsset. Der infoasset Broker - Architektur, Anpassung und Erweiterung. infoAsset Broker Documentation, Software Systems Institute, Hamburg University of Science and Technology, February 2003.

[inf05] infoAsset. The infoasset Broker [online]. 2005 [cited 10 August 2005]. Available from: http://www.infoasset.de/contents/products/index.htm.

[Kli05] Helge Klimek. Eine Web Service Schnittstelle für ein Web Service Entwickler Portal. Bachelor Thesis, Software Systems Institute, Hamburg University of Science and Technology, October 2005.

[Mat03] Florian Matthes. Uml-datenmodell für den infoasset broker 1.8. infoAsset Broker Documentation, Software Systems Institute, Hamburg University of Science and Technology, August 2003.

[Mic05] Microsoft. Sharepoint Portal Server 2003 Overview [online]. 2005 [cited 10 August 2005]. Available from: http://www.microsoft.com/office/sharepoint/prodinfo/overview.mspx.

[Ort05] Ed Ort. Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools. April 2005.

[Plu05] Plumtree. Plumtree Search [online]. 2005 [cited 11 August 2005]. Available from: `http://www.plumtree.com/products/search/`.

[ST05] Robert Steele and Takahiro Tsubono. Reserving immutable services through web service implementation versioning. In *WEBIST*, pages 125–132, 2005.

[Wav04] Rogue Wave. An Introduction to Web Services. February 2004.

[Weg00] Holm Wegner. The infoasset Broker. infoAsset Broker Documentation, Software Systems Institute, Hamburg University of Science and Technology, 2000.

[Weg02] Holm Wegner. Analyse und objektorientierter entwurf eines integrierten portalsystems fr das wissensmanagement. page 158, January 2002.

[Wik05] Wikipedia. Web Services [online]. August 2005 [cited 10 August 2005]. Available from: `http://en.wikipedia.org/wiki/Web_Service`.

[ZTP03] Olaf Zimmermann, Mark Tomlinson, and Stefan Peuser. *Perspectives on Web Services-Applying SOAP, WSDL and UDDI to Real-World Projects*. Springer-Verlag Berlin Heidelberg, 2003.

# Appendix

## .1 New Methods in Class IMPDocument

```
0  package de.infoasset.broker.services.core;

   import java.io.File;
   import java.util.Date;
   import java.util.HashSet;
5  import java.util.Iterator;
   import java.util.Vector;
   import de.infoasset.broker.interfaces.cis.Unit;
   import de.infoasset.broker.interfaces.cis.Units;
   import de.infoasset.broker.interfaces.core.AssetLock;
10 import de.infoasset.broker.interfaces.core.Concept;
   import de.infoasset.broker.interfaces.core.Concepts;
   import de.infoasset.broker.interfaces.core.Directory;
   import de.infoasset.broker.interfaces.core.Document;
   import de.infoasset.broker.interfaces.core.Documents;
15 import de.infoasset.broker.interfaces.core.DomainValue;
   import de.infoasset.broker.interfaces.core.Group;
   import de.infoasset.broker.interfaces.core.Membership;
   import de.infoasset.broker.interfaces.core.Person;
   import de.infoasset.broker.services.asset.AssetDelegate;
20 import de.infoasset.broker.store.AttributeSignature;
   import de.infoasset.broker.store.Container;
   import de.infoasset.broker.store.Content;
   import de.infoasset.broker.store.Query;
   import de.infoasset.broker.store.QueryAnd;
25 import de.infoasset.broker.store.QueryEquals;
   import de.infoasset.broker.util.textFilter.TextFilter;

   /**
    * IMPDocument is an implementation of
30  * {@link de.infoasset.broker.interfaces.Document}.
    */

   public class IMPDocument extends AssetDelegate implements Document {

35 //......

      public String getZweck() {
         return this.getProperty(Documents.asZweck);
      }
40
      public void setZweck(String zweck) {
         this.setProperty(Documents.asZweck, zweck);
      }

45    public String getDiffVorversion() {
         return this.getProperty(Documents.asDiffVorversion);
      }

      public void setDiffVorversion(String diffVorversion) {
50       this.setProperty(Documents.asDiffVorversion, diffVorversion);
```

```
    }

    public String getReasonOnExpiry() {
       return this.getProperty(Documents.asReasonOnExpiry);
    }

    public void setReasonOnExpiry(String reasonOnExpiry) {
       this.setProperty(Documents.asReasonOnExpiry, reasonOnExpiry);
    }

    public String getURL() {
       return this.getProperty(Documents.asURL);
    }

    public void setURL(String url) {
       this.setProperty(Documents.asURL, url);
    }

    public String getTestURL() {
       return this.getProperty(Documents.asTestURL);
    }

    public void setTestURL(String testUrl) {
       this.setProperty(Documents.asTestURL, testUrl);
    }

    public String getVerfuegbarkeit() {
       return this.getProperty(Documents.asVerfuegbarkeit);
    }

    public void setVerfuegbarkeit(String verfuegbarkeit) {
       this.setProperty(Documents.asVerfuegbarkeit, verfuegbarkeit);
    }

    public Date getGueltigVonDate() {
       return this.getProperty(Documents.asGueltigVonDate);
    }

    public void setGueltigVonDate(Date gueltigVonDate) {
       this.setProperty(Documents.asGueltigVonDate, gueltigVonDate);
    }

    public Date getFreigegebenDate() {
       return this.getProperty(Documents.asFreigegebenDate);
    }

    public void setFreigegebenDate(Date freigegebenDate) {
       this.setProperty(Documents.asFreigegebenDate, freigegebenDate);
    }

    public Date getVeroeffentlichDate() {
       return this.getProperty(Documents.asVeroeffentlichDate);
    }

    public void setVeroeffentlichDate(Date veroeffentlichDate) {
       this.setProperty(Documents.asVeroeffentlichDate, veroeffentlichDate);
    }

    public Iterator getAllConceptIds() {
       return services.getConcepts().getConceptsOfDocumentIds(getId());
    }

    public Iterator getConceptIds() {
       HashSet normalConcepts = new HashSet();
       Iterator allConcepts = services.getConcepts().getConceptsOfDocumentIds(getId())
           ;
       while (allConcepts.hasNext()) {
          Concept thisConcept = services.getConcepts().
```

```
            getConcept((String)allConcepts.next());
         boolean flag = true;
         for (Concept c = thisConcept.getSuper(); c != null; c = c.getSuper()) {
            String cName = c.getName();
            if ("Fachliche Kontexte".equals(cName) ||
            "Rechtliche Grundlagen".equals(cName) ||
            "Zielgruppen".equals(cName)) {
               flag = false;
               break;
            }
         }
         if (flag == true) normalConcepts.add(thisConcept.getId());
      }
      return normalConcepts.iterator();
   }

   public Iterator getFachlicherKontexte() {
      HashSet fachlicherKontexte = new HashSet();
      Iterator allConcepts = services.getConcepts().getConceptsOfDocumentIds(getId())
          ;
      while (allConcepts.hasNext()) {
         Concept thisConcept = services.getConcepts().
            getConcept((String)allConcepts.next());
         for (Concept c = thisConcept.getSuper(); c != null; c = c.getSuper()) {
            String cName = c.getName();
            if ("Fachliche Kontexte".equals(cName)) {
               fachlicherKontexte.add(thisConcept.getId());
               break;
            }
         }
      }
      return fachlicherKontexte.iterator();
   }

   public Iterator getRechtlicheGrundlagen() {
      HashSet rechtlicheGrundlagen = new HashSet();
      Iterator allConcepts = services.getConcepts().getConceptsOfDocumentIds(getId())
          ;
      while (allConcepts.hasNext()) {
         Concept thisConcept = services.getConcepts().
            getConcept((String)allConcepts.next());
         for (Concept c = thisConcept.getSuper(); c != null; c = c.getSuper()) {
            String cName = c.getName();
            if ("Rechtliche Grundlagen".equals(cName)) {
               rechtlicheGrundlagen.add(thisConcept.getId());
               break;
            }
         }
      }
      return rechtlicheGrundlagen.iterator();
   }

   public Iterator getZielgruppen() {
      HashSet zielgruppen = new HashSet();
      Iterator allConcepts = services.getConcepts().getConceptsOfDocumentIds(getId())
          ;
         while (allConcepts.hasNext()) {
         Concept thisConcept = services.getConcepts().
            getConcept((String)allConcepts.next());
         for (Concept c = thisConcept.getSuper(); c != null; c = c.getSuper()) {
            String cName = c.getName();
            if ("Zielgruppen".equals(cName)) {
               zielgruppen.add(thisConcept.getId());
               break;
            }
         }
      }
      return zielgruppen.iterator();
```

```
    }

    public Iterator getUsedByWSDocuments() {
        return this.getAssociatedAssetIds
            (Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole());
    }

    public void addUsedByWSDocument(String wsDocId) {
        Document usedByWSDocument = services.getDocuments().getDocument(wsDocId);
        if (!this.hasAssociatedAsset(
        Documents.DOCUMENT_USEDBYWSDOCUMENT.
        getThisRole(), usedByWSDocument)) {
            this.createAssociation
            (Documents.DOCUMENT_USEDBYWSDOCUMENT.
            getThisRole(), usedByWSDocument,          Documents.
                DOCUMENT_USEDBYWSDOCUMENT.getThatRole());
        }
        if (!this.hasAssociatedAsset(
        Documents.DOCUMENT_USEDWSDOCUMENT.
        getThisRole(), usedByWSDocument)) {
            this.createAssociation
            (Documents.DOCUMENT_USEDWSDOCUMENT.
            getThatRole(), usedByWSDocument,
            Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole());
        }
    }

    public void removeUsedByWSDocument(String wsDocId) {
        Document usedByWSDocument = services.getDocuments().getDocument(wsDocId);
        this.removeAssociatedAsset(
            Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole(),
            usedByWSDocument,
            Documents.DOCUMENT_USEDBYWSDOCUMENT.getThatRole());
        this.removeAssociatedAsset(
            Documents.DOCUMENT_USEDWSDOCUMENT.getThatRole(),
            usedByWSDocument,
            Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole());
    }

    public Iterator getUsedWSDocuments() {
        return this.getAssociatedAssetIds
            (Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole());
    }

    public void addUsedWSDocument(String wsDocId) {
        Document usedWSDocument = services.getDocuments().getDocument(wsDocId);
        if (!this.hasAssociatedAsset(
        Documents.DOCUMENT_USEDWSDOCUMENT.
        getThisRole(), usedWSDocument)) {
            this.createAssociation(
                Documents.DOCUMENT_USEDWSDOCUMENT.
                getThisRole(), usedWSDocument,
                Documents.DOCUMENT_USEDWSDOCUMENT.getThatRole());
        }
        if (!this.hasAssociatedAsset(
        Documents.DOCUMENT_USEDBYWSDOCUMENT.
        getThisRole(), usedWSDocument)) {
            this.createAssociation(
                Documents.DOCUMENT_USEDBYWSDOCUMENT.
                getThatRole(), usedWSDocument,
                Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole());
        }
    }

    public void removeUsedWSDocument(String wsDocId) {
        Document usedWSDocument = services.getDocuments().getDocument(wsDocId);
        this.removeAssociatedAsset(
            Documents.DOCUMENT_USEDWSDOCUMENT.getThisRole(),
```

```
250            usedWSDocument,
               Documents.DOCUMENT_USEDWSDOCUMENT.getThatRole());
           this.removeAssociatedAsset(
               Documents.DOCUMENT_USEDBYWSDOCUMENT.getThatRole(),
               usedWSDocument,
255            Documents.DOCUMENT_USEDBYWSDOCUMENT.getThisRole());
       }

       public void addEigentuemerId(String eigentuemerId) {
           Person person = services.getPersons().getPerson(eigentuemerId);
260        if (!this.hasAssociatedAsset(
           Documents.DOCUMENT_PERSON_EIGENTUEMER.getThisRole(), person)) {
               this.createAssociation(
               Documents.DOCUMENT_PERSON_EIGENTUEMER.
               getThisRole(), person, Documents.         DOCUMENT_PERSON_EIGENTUEMER.
                   getThatRole());
265        }
       }

       public void removeEigentuemer(String eigentuemerId) {
           Person person = services.getPersons().getPerson(eigentuemerId);
270        this.removeAssociatedAsset(
               Documents.DOCUMENT_PERSON_EIGENTUEMER.
               getThisRole(), person, Documents.         DOCUMENT_PERSON_EIGENTUEMER.
                   getThatRole());
       }

275    public Iterator getEigentuemern() {
           return this.getAssociatedAssetIds(
               Documents.DOCUMENT_PERSON_EIGENTUEMER.getThisRole());
       }

280    public Iterator getAnsprechpartnerFachlichIds() {
           return this.getAssociatedAssetIds(
               Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.
               getThisRole());
       }
285
       public void addAnsprechpartnerFachlichId(String id) {
           Person ansprechpartnerFachlich = services.getPersons().getPerson(id);
           if (!this.hasAssociatedAsset(
           Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.getThisRole(),
290        ansprechpartnerFachlich)) {
               this.createAssociation(
               Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.
               getThisRole(), ansprechpartnerFachlich,
               Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.
295            getThatRole());
           }
       }

       public void removeAnsprechpartnerFachlichId(String personId) {
300        Person person = services.getPersons().getPerson(personId);
           this.removeAssociatedAsset(
               Documents.DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.
               getThisRole(), person, Documents.
                   DOCUMENT_PERSON_ANSPRECHPARTNERFACHLICH.getThatRole());
       }
305
       public Iterator getAnsprechpartnerTechnischIds() {
           return this.getAssociatedAssetIds(
               Documents.DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.
               getThisRole());
310    }

       public void addAnsprechpartnerTechnischId(String id) {
           Person ansprechpartnerTechnisch = services.getPersons().getPerson(id);
           if (!this.hasAssociatedAsset(
```

```
315        Documents.DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.
           getThisRole(), ansprechpartnerTechnisch)) {
              this.createAssociation(
              Documents.DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.
              getThisRole(), ansprechpartnerTechnisch,
320           Documents.DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.
              getThatRole());
           }
        }

325     public void removeAnsprechpartnerTechnischId(String personId) {
           Person person = services.getPersons().getPerson(personId);
           this.removeAssociatedAsset(
              Documents.DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.
              getThisRole(), person, Documents.
                  DOCUMENT_PERSON_ANSPRECHPARTNERTECHNISCH.getThatRole());
330     }

        public String getFreigegebenVon() {
           return this.getAssociatedAssetId(
              Documents.PERSON_DOCUMENT_FREIGEGEBENVON.getManyRole());
335     }

        public void setFreigegebenVon(String id) {
           Person freigegebenVon = services.getPersons().getPerson(id);
           this.createAssociation(
340           Documents.PERSON_DOCUMENT_FREIGEGEBENVON.
              getManyRole(), freigegebenVon, Documents.
                  PERSON_DOCUMENT_FREIGEGEBENVON.getOneRole());
        }

        public boolean getHistorieStatus() {
345        return (this.getProperty(Documents.asHistorieStatus) == 1 ? true : false);
        }

        public void setHistorieStatus(boolean flag) {
           this.setProperty(Documents.asHistorieStatus, (flag ? 1 : 0));
350     }

        public String getHistoryDirectory() {
           return this.getProperty(Documents.asHistoryDirectory);
        }
355
        public void setHistoryDirectory(String historyDirectoryId) {
           this.setProperty(Documents.asHistoryDirectory, historyDirectoryId);
        }

360     public void checkDocumentForPublish() {
           Date today = new Date();
           if (today.after(getVeroeffentlichDate()) ||
           today.equals(getVeroeffentlichDate())) {
              setDocumentStatus("3");
365           if (getParentVersion() != null) {
                 Document parentVersion =
                 services.getDocuments().getDocument(getParentVersion());
                 if (parentVersion.getHistoryDirectory() != null) {
                    parentVersion.moveToDirectory
370                 (parentVersion.getHistoryDirectory());
                    parentVersion.write_LEGACY();
                 }
              }
              write_LEGACY();
375        }
        }

        public void checkDocumentForExpire() {
           Date today = new Date();
380        if (today.after(getExpiryDate())) {
```

```
            setDocumentStatus("5");
            if (getHistoryDirectory() != null) {
               moveToDirectory(getHistoryDirectory());
               write_LEGACY();
            }
         }
      }

      //......

}
```