The electronic government and its client systems

L.J. van der Starre
University of Groningen

Abstract

Dealing with the government is often regarded as a negative experience. Bureaucracy and a not transparent, inefficient way of working are the troubling thoughts remaining in the back of the heads of the citizens and companies. In the current Internet age people expect all government services, products and information to be on the Internet, and that all the dealings with the government can be done electronically.

In the Netherlands many plans for reforming the government have seen the light over the past decade. Many eventually failed. The action plan "Andere Overheid" is the latest attempt in creating an *electronic government*. This plan calls for an customer-centric government, which is efficient, transparent and conducts 65% of its business over the internet by 2007. This plan gave birth to several initiatives and plans to implement the electronic government.

But how is this electronic government implemented? What are the fundamental principles and the key aspects? And how does a client system fit in the electronic government?

This thesis first looks at the fundamental principles and the reference architecture of the Dutch electronic government. A proposal on how this can can be implemented follows. After having defined what the implementation looks like, the focus shifts to the client systems which have to be absorbed in this electronic goverment. A proof of concept is implemented where an existing system for communal taxes is "SOA-enabled", meaning that web services are build on the system so that it will fit in the big service oriented architecture of the electronic government.

Contents

List of Figures

List of Tables

Preface

This thesis is written by Laurens van der Starre at the Groningen office of the Vertis BV IT company. The research domain is the electronic government and how the GemTax product for the billing and collecting of communal taxes should fit into this electronic government.

The company Vertis BV originated from a collaboration between Avebe and Akzo Systems. From this collaboration a software company was founded, on the 1st of January 1990 Vertis BV was born. Today Vertis BV has five offices spread throughout the Netherlands: Veendam, Enschede, Wageningen, Leidschendam and the head office in Groningen. Vertis BV has approximately 400 employees.

The company Vertis BV is part of a holding company. Magentis BV and ApplicationNet BV are examples of companies which are also part of this Holding. At the end of 2005 the holding has been taken over by the Ordina company. From 2007 the name Vertis BV will disappear and will be replaced by a new one, which will identify it as an Ordina company.

Vertis BV specialises itself as a software company which builds software based on Oracle technology. It is therefore an Oracle Partner. Vertis BV builds standard and custom software based on Oracle Products, but also builds Oracle E-Business Suite, Business Intelligence and Infrastructure solutions.

The customers of Vertis BV are situated in various domains, varying from the food, water and agricultural domain to the government domains.

Acknowledgements

Groningen, the 20th of December 2006.

I would like to thank a couple of people who helped me with my thesis, and people who made my time at the Vertis BV company a lot of fun.

First, of course, my mentor Arjan Loermans, a.k.a. "1337-ment0r". Arjan is a technical consultant at the Innovation and Support department of Vertis and was appointed as my mentor when I started. Arjan supported me in every way with my research and implementation and gave me the confidence and freedom to go my own way in the research for my thesis.

Secondly, Jurjen Melinga. Also from the Innovation and Support department, Jurjen is a software architect involved in a lot of projects at Vertis. His keen insight helped me on my way writing this thesis. As a fellow photography enthusiast, gamer and with almost the same music taste we always had something to talk about or to discuss. However, Canon vs. Nikon, that is where we do not agree.

Roel Strijkstra, my boss from Innovation and Support gave me the freedom to do what I wanted. I hope your sabbatical gives you some rest after working so hard! It was always fun to motivate Harrie van der Laan on every Friday morning at 9am by asking if it was already weekend and if we could get home already. Harrie was always willing to help when I had some database problem. Never far away there was Jan-Lucas van der Ploeg. He happened also to have a Nintendo DS and also the game Animal Crossing. I hope we didn't waste too much of our boss's time multiplayer gaming ... Going from the Performance Wizards to quality control there was Kees Bonting. Quality is the middle name of Kees Bonting, a man with whom I calculated project prices for two proposals. Kees was my roommate at the office for a couple of months.

Special thanks go to Marcel Belinga and Ron Alders. As experienced software engineers they were always willing to help when Oracle's SOA Suite went berserk on my machine. Frustratingly enough everything always seemed to work on Marcel's machine (how does he do it) ...

I want to thank Ruud Overbeek, Toon van der Ploeg and Hinko Frouws. Ruud helped me a lot with his technical knowledge of the GemTax system. Toon and Hinko were always willing to share their functional knowledge of GemTax with me.

Finally I want to name a few more people: Niels Maneschijn, Jan Salvador van der Ven, Jan de Graaf, Wouter Zunneberg, Luit Buist, Alex Harkema, Bart Dopheide, Hugo Schijf, Teijo Doornkamp, Richard Arling, Foskea Raven, Kars Velting, Kees van der Mark, Diet Daleman, Arwen Botterweg, plus everybody I forgot to mention (sorry!).

Last, but not least, professor Marco Aiello from the University of Groningen for guiding my thesis to the desired quality and Jan Jongejan from the University of Groningen for his very good and thorough review.

–Laurens van der Starre.

Part I

Introduction

Chapter 1

Introduction

1.1 Background

Ever since the computer was introduced in the Netherlands, the Dutch government tried to increase the efficiency of their information warehousing. In the 60's this resulted in a central data administration for municipalities (the so called "GBA" or "Gemeentelijke Basis Administratie voor persoonsgegevens", "Municipal Personal Records Database") which houses the personal data of citizens like marital status, offspring etcetera.

This was, of course, a great leap forward from the paper archives. However, the problem was that the number systems within the government kept expanding, resulting in the Dutch government of today which is faced with data spread over *fifteen thousand* government bodies in *thirty thousand* systems [13]. Making matters worse: these systems are more or less developed independently of each other. It may come as no surprise that these systems are not (fully) compatible with each other.

This challenging ICT infrastructure not only sprouted some technical problems, but also the government's service towards their customers, the citizens and businesses, suffered.

In 2003 the Dutch cabinet started an action plan called "Programma Andere Overheid" which called for the realisation of a better functioning, more efficient, demand-driven and customer-centric government [10]. To accomplice this, better cooperation between national government, local government bodies and public agencies is needed. This means that all those different systems introduced earlier have to start working together –somehow. Government agencies and government related organisations have proposed a service oriented approach for the realisation of the *electronic government*.

The "Programma Andere Overheid" is not the first attempt of the Dutch government to implement electronic services for the government. Over the years many plans have been initiated but they did not always result in the goals the policy makers expected [4]. The main reason these earlier plans failed was because the (local) governments in question and their public servants were not motivated enough to really make it happen. The "Programma Andere Overheid" is the latest attempt to reform the ICT services of the government. However, this time it is backed by laws and regulations. The date these laws will be valid is the ultimate deadline for the government and public agencies to comply.

This thesis is done for the company Vertis BV. Vertis BV is part of Ordina NV and makes software based on the products of Oracle for a wide range of customers, including the government. Oracle is one of the world's leading enterprise software companies, and has a wide range of database products, enterprise solutions and development frameworks. Vertis BV has a product for the

billing and collection of communal taxes: GemTax is for use by municipal authorities. GemTax depends on two systems: an operational data storage system ("BAS") and a system that uses that data for the billing and collecting processes ("HEIN"). These systems are not yet compliant with the vision that the Dutch government has. In this thesis GemTax is redesigned as part of a Proof of Concept to fit it into the electronic government framework.

1.2 Research question

This thesis is concerned with the electronic government and, in particular, with the fitting of a tax billing and collecting system into the electronic government framework. The plans that exist for the electronic government are high level documents that should be read more as a guideline rather than technical design documents. Aside from the technicalities the judiciary also has a say in the electronic government. Laws about the protection of privacy sensitive data and archiving formal communications of the government are of course also applicable for the electronic government. This has a profound impact on what can be done, and what not.

The research question of this thesis is therefore the following:

> In what way should the electronic government architecture be implemented such that client systems can use the provided data and functionality in the way Dutch laws and regulations prescribe?

To answer this question the following needs to be investigated:

1. What are the underlying principles of the electronic government?
2. What is the electronic government?
3. What is a viable solution for implementing the electronic government?
4. How should client systems fit in this electronic government?

The theoretical description and viable solution for implementing the electronic government and the way client systems should fit into the electronic government framework is put into practice in a Proof of Concept where the GemTax product will be redesigned to fit into the electronic government as a client system.

1.3 Thesis organisation

This thesis is basically divided in four parts. First there is the introduction. Here the background, research question, fundamental principles and reference architecture of the electronic government together with the service oriented architecture will be introduced.

The second part proposes an implementation of the electronic government. The different parts of the reference architecture will be placed in a more technical context.

The third part is concerned with the Proof of Concept. The GemTax system for billing and collecting communal taxes will get an SOA-extension which will make it fit more into the IT-landscape of the electronic government.

The last part contains the conclusion, the bibliography and the index.

Chapter 2

Introducing the electronic government

This chapter will give an introduction in the world of the electronic government. It will introduce the architecture framework of the electronic government, and explain its fundamental principals. The goal of this chapter is to define the foundation of the electronic government in such a way that in the following chapters the technical implementation of this foundation can be described. The foundation is based on principles and guidelines which are the result of extensive research of the Dutch government and the European Union. These guidelines and principles combined form the basis of the Dutch Government Reference Architecture NORA ("Nederlandse Overheid Referentie Architectuur") [7] which is the reference architecture framework all government agencies and public agencies should (eventually) comply to.

2.1 The basic principles of the electronic government

The architecture framework of the Dutch electronic government is based on a number of principles. These principles originated from four different stakeholders: the Dutch government, the European Union, the Dutch citizens and the Dutch business community. These stakeholders work within the boundaries defined by the fifth stakeholder: the Dutch judiciary. In the following subsections these principles are described. All these principles are transformed into the fundamental principles of the Dutch electronic government reference architecture.

2.1.1 The action plan "'Programma Andere Overheid"

The goal of "Programma Andere Overheid" is to achieve a modern and demand-driven electronic government that [13][16][10]:

1. **Inconveniences the public and the business community with requests for data only when this is absolutely necessary;**
2. **Offers a rapid and good service;**
3. **Can not be misled;**
4. **Knows its facts;**
5. **Instils the public and the industrial community with confidence;**

6. **Is provided at a cost that is no higher than strictly necessary.**

Beside the points above, the government aims to make 65% of its services electronically available on the internet for citizens and the business community in 2007. This means that a customer-centric electronic government has to be created.

2.1.2 Dutch e-Citizen Service Code

An important objective that the electronic government should accomplish is becoming a *customer-centric* government. What this exactly means for this customer when the electronic government is finally implemented is defined in the Dutch e-Citizen Service Code.

The Code specifies ten principles which describe the way the government should service the customer. The term "customer" is defined in the broad sense of the word: not only does it represent the citizen, but also businesses or institutions. These principles can be considered as the core of the total electronic government from the customer's perspective. The principles are defined as follows [8]:

1. **Choice of communication channel.** The customer should be able to choose the communication channel it wishes to use to conduct its business with the government. Channels include the internet, e-mail, phone, mail, counter etcetera. The government should ensure multichannel service delivery and make sure that the different channels work analogously.

2. **Transparency of the public sector.** The customers should know where to apply for information and public services. The government should act as one entity and should provide one-stop-shop service delivery with "no wrong doors".

3. **Overview of rights and duties.** The government should provide the customers with transparent information about their rights and duties in a way that the customers know what services they are entitled to.

4. **Personalised information.** The government should provide personalised information tailored to the customer's needs. The data should be consistent, up to date and correct.

5. **Convenient services.** The customers have to provide information or data only once when asked for. This means that the government should reuse information or data. The information or data is used responsibly and services are only delivered when asked for.

6. **Comprehensible procedures.** The government should keep the customer informed about the procedures the customer is involved in, in a "tracking and tracing" way. This means the customer should be able to see the status of the procedures and the time the status changed.

7. **Smart administration.** The customers can suggest improvements and lodge complaints. The government should compensate mistakes and improve its services accordingly.

8. **Enable benchmarking.** The government should supply benchmark information such that the customer can compare, check and measure government outcome.

9. **Empowerment.** The customer is invited to participate in the decision making and to promote its interests. For this, the government should provide the necessary information and the appropriate instruments.

10. **e-File.** The government should provide insight into the information it has about the customer and for which purposes this data is used.

2.1.3 European Union directives

Within the European Union there is an ever stronger need to improve collaboration and product exchange between the government and public agencies of the European Union. The European Interoperability Framework also has its effects on the Dutch electronic government. The eight principles of this European Interoperability Framework are described next [7]:

1. **Accessibility.** Government agencies should provide different channels on which citizens and the industry can communicate with the agencies.
2. **Multilingual.** The electronic services of the government should be provided in different languages. However, in the case of the Dutch government this means that only if the law dictates otherwise, the Dutch language is solely used.
3. **Security.** The government should supply its services in conformance with the (national) guidelines for security such that the identification, authentication and confidentiality is transparent for the user.
4. **Privacy.** The electronic services of the government agencies should respect the privacy sensitive nature of the data it works with. It should therefore comply to the (national) privacy laws.
5. **Subsidiarity.** Subsidiarity basically means that not every detail of the electronic government is defined by the central government. In most cases the specific know-how of the specific processes and the domain is in the lower (decentralised) levels. Lower levels, specific government organisations for example, should be given the freedom to determine for themselves how to realise the electronic government, taking into account the advisories that are given from the higher levels of the government.
6. **Open standards.** To improve interoperability the use of industry open standards is preferred.
7. **Open source.** Open source alternatives to closed source software should be given an "honest" chance in the choice for software.
8. **Multilateral solutions.** The wish for collaboration between a large number of government and public agencies calls for local, national and EU-wide information hubs.

2.1.4 Specific wishes from the business community

The business community in general wishes a reduction of the administrative burden when dealing with the government. Elaborating on this, four important principles can be identified.

The government should:

1. **Reduce the number of rules.** The business community wishes to do its business with the government in a efficient way, without constantly being bothered by a wide array of rules.
2. **Request needed data or information only once, and ensure reuse.** Information that is already known by the government should not be asked for again.
3. **Make use of Key Data Stores.** The government should have its authentic data in order in centralised databases.
4. **Ensure optimal usage of ICT in their processes.** The government should have good ICT support for their processes to ensure good service to the business community in an electronic way.

2.2 Dutch judiciary

The government always works in correspondence to the law which dictates its rights, plights and boundary conditions of its public tasks. Not all laws and regulation come into view in this thesis. However, looking at the electronic government with its ambition to lower the administrative burden and improve the service over the internet a couple of judiciary aspects come into view:

1. **Law on the protection of privacy sensitive data ("Wet bescherming persoonsgegevens") (Wbp).** Lowering the administrative burden and reuse of data means that (private) information and data about the citizen will be processed more, in different systems and in more places. The Wbp can prevent this from happening when the data is processed for another goal than for which it was originally gathered. Also, if there is no clear public need or judicial basis in the form of a formal law the Wbp can disallow the processing [12].

2. **Law on the archiving of official government documents and communication ("Archiefwet").** Governmental documents and communications need to be archived according to this law. This also includes electronic documents of the electronic government.

3. **Law on the Citizen Service Number ("Wet algemene bepalingen Burger-servicenummer").** For sharing privacy sensitive data between government agencies and between the citizen and the government a "privacy insensitive" identifying number is created for the citizen. This number is numerically equivalent to the social security number of a Dutch citizen. This number is stored in the GBA Key Data Store. Government agencies can identify all personal information about the citizen (when needed) by only requesting this "BSN" number. This allows for a relative trustworthy way of sharing personal and privacy sensitive information. This law will probably be effective in 2007.

4. **Law on the Key Data Stores.** The law on the protection of privacy sensitive data only allows processing of privacy sensitive data when the Wbp allows it. For processing data in relation to the centralised databases for storing the key data of the customers, a judicial basis in the form of a formal law for every Key Data Store is needed. For the GBA Key Data Store this is already a fact.

5. **Feedback duty.** For a centralised data organisation such as the Key Data Stores to work, the clients of these data stores need to be able to report possible errors in the data originating these data stores. To make sure clients give this feedback and the authentic registers deal with this feedback in a serious way, a law is in the making which is concerned with this "feedback duty". This law dictates that the authentic registers must have a way for clients to give feedback on the data and that this feedback is investigated by the authentic register.

2.2.1 Dutch electronic government reference architecture

The principles from the previous sections form the basis of the Dutch electronic government reference architecture. The basis, or core, of the Dutch electronic reference architecture principles can be subdivided in six "themes" [7], where the principles of the previous sections can be accommodated:

1. **High quality of service.** The government should provide electronic services next to their existing channels for service (mail, telephone etcetera) which citizens, companies and other organisations can use. The services should be easily accessible and transparent. To use these services one government-wide electronic identity should be provided and used. The quality of service is assessed and improved when necessary, and a simple and transparent way to lodge complaints or to make suggestions should be provided.

2. **Reduction of administrative burden.** The government should only ask for information or data once. It should not ask for information which it could already know. The number of rules should be reduced and their complexity simplified to lower the administrative burden.

3. **Transparency.** The government should give insight into the state of processes and decisions relevant to the user. Relevant governmental information should also be easily accessible.

4. **Pro-active service.** The government should point out services relevant for the citizen or company.

5. **A government that operates as a single and trustworthy entity.** The government presents itself to the citizens and companies as a single trustworthy entity.

6. **Improved effectiveness of the government.** Generic services and reuse of components or services is preferred.

The themes mentioned here form the core of the Dutch electronic government reference architecture used to create the architecture framework.

2.3 Architecture framework

The architectural framework for the Dutch electronic government is in essence an enterprise architecture framework which gives different scopes on the electronic government: the conceptual scope (the business architecture), the logical scope (the information architecture) and the physical scope (the technical architecture). The Dutch electronic government reference architecture describes an architectural framework on these three scopes divided in three different views: "Who", "What", and "How". It shows a clear resemblance to a subset of Zachman's Enterprise Architecture Framework [18][17]. In this thesis Zachman's framework will form the basis of the Dutch electronic government reference architecture framework. The framework is depicted in Figure 2.1.

The electronic government and the architectural framework can be viewed as a guideline or a layer on top of every government agency. It defines the way the government agencies should implement a customer-centric electronic government that acts as a single entity and enables one-stop-shopping for the customer.

In the sections below the different scopes are briefly introduced whereas chapter 4 goes into the (technical) details when an implementation is proposed for the electronic government.

2.3.1 Business model

The business architecture is concerned with the government agencies which form the electronic government. There are around 1600 different government and public organisations in the Netherlands, and within the electronic government they have to work together and provide services mostly spanning across these organisations. These organisations operate as sovereign entities, but have to give insight into their processes and services in a way that these can be combined and found by other entities, such that the citizen or company can get their information from any of these entities. In this way a "no wrong door" policy is ensured that helps the government to act as a single entity.

Government organisations provide services as specified by the law or its "job description". These services can be part of a product or process the government provides. This means that the services have to be coordinated to create an end-product for the citizen or company.

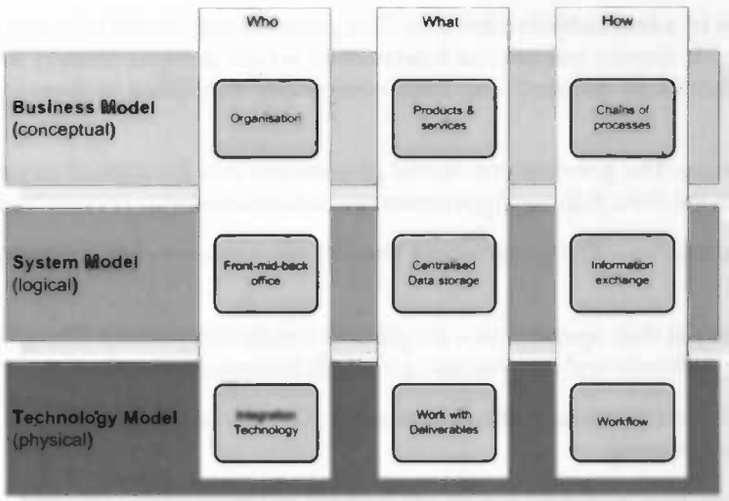

Figure 2.1: The architecture framework of the electronic government based on the Zachman framework [17].

2.3.2 System model

The information architecture is concerned with how information is structured and from what kind of components it is built up from. It describes the function of the information and how it contributes to the vision the policy makers have in relation to the information services [14].

In the case of the electronic government it is therefore important to realise which applications and employees take part in the sharing of (automated) data, what this data is, what it contains and how this information is shared.

2.3.3 Technology model

The technical architecture is the physical realisation of the conceptual and logical scope of the business and information architecture. It is concerned with the technical realisation of the needed technical components, the data storage and the network infrastructure.

2.3.4 Security

Security is an important aspect of the electronic government. It is concerned with the technical aspect of security (that is: the protection of privacy sensitive data, authorisation and authentication processes), but also with issues as the continuity and governance of electronic government processes. Security influences the electronic government on the different levels of the architecture framework.

2.3.5 Maintenance

The maintenance aspect of the electronic government is concerned with the functional, application and technical maintenance of the government's services, applications and ICT infrastructure. Besides that, the communication standards, the Service Level Agreements and release planning

for new or updated government processes is something that belongs to the maintenance dimensions of the electronic government. It directly influences the different models of the architecture framework.

2.4 Service oriented approach

In the business model it is mentioned that government organisations should provide services that can be orchestrated into products or specific government processes. It is this "service oriented thinking" together with the desire to be open and transparent combined with the sharing of information that led to the design decision that the electronic government should be based on a service oriented architecture. In Chapter 3 the service oriented architecture will be introduced.

2.5 Summary

In this chapter the underlying principles of the Dutch electronic government were introduced. Six different stakeholders exist, each with their own set of basic principles which find their way into the architecture framework of the Dutch electronic government reference architecture. This thesis uses a subset of the Zachman framework to describe the architecture framework for the electronic government. This framework consists of three scopes (conceptual, logical and technical) and for each scope three different views (who, what and how). This framework can be seen as a layer on top of governmental agencies to guide their implementation of the electronic government.

Chapter 3

The Service Oriented Approach

To implement the Dutch electronic government with the requirements as described in Chapter 2 a service oriented approach is suggested [7]. This means evolving from rigid monolithic systems to an open and flexible service oriented landscape which allows the government to build flexible and government-wide processes in a relatively easy way. This chapter introduces the service oriented architecture and describes the key aspects of it.

3.1 What is a Service Oriented Architecture?

Service oriented computing (SOC) is a computing paradigm that utilises services as fundamental elements for developing applications [5]. The services contain functionality that can be published for others to find, discovered and binded to get access to the functionality. A service oriented architecture ("SOA") builds on the foundation of service oriented computing as it is defined by the industry and academic community as a way of designing systems composed of software components with remotely-callable interfaces. However, in practice an SOA is defined as designing systems composed of web services using enabling technologies, frameworks and standards which are overall being accepted as industry standards.

These "SOA-enabling frameworks" build upon the ideas of web services and message-based information exchange and provide a complete infrastructure framework to build enterprise-wide SOA-enabled applications. Examples of SOA-enabling Frameworks are Oracle Fusion, Microsoft BizTalk and IBM WebSphere. In this thesis the SOA-enabling Frameworks are not elaborated upon, but the Proof of Concept (starting at Chapter 5) will make use of Oracle's SOA Framework.

In the sections below some commonly used industry standard enabling technologies and standards are introduced.

3.1.1 Services

Functionality in an SOA is provided by the services. A service is a single instance, loosely coupled, discoverable and context independent software entity. In practice this means:

1. Services are loosely coupled because they use a document based message exchange to support interoperability and reduce coupling. These messages are based on the open SOAP standard, a standard based on XML.

2. Services are discoverable in a way that their interface descriptions are published and can be discovered at design time, as well as runtime.
3. Context independent means that services are designed to ignore the context in which they are used, and can therefore be reused in other contexts not known at design time.
4. A service is a single instance that a number of clients can communicate with.

3.1.2 Discoverable services

In section 3.1.1 a service was described as discoverable at design and runtime. To accomplish this, an SOA contains service consumers, service providers and service registries. The meaning of these terms is best described pictorially, see Figure 3.1.

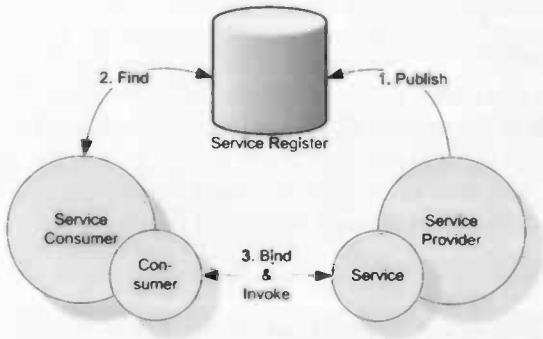

Figure 3.1: Service discoverability.

In this figure the provider publishes its interface description and its location in a service registry. The enabling technology used in the SOA-enabling frameworks for this purpose is called a "UDDI" (Universal Description, Discovery, and Integration). A consumer can communicate with the UDDI to find the service provider it needs. It can then bind and invoke the service provided by the provider to use its functionality. The roles of consumer and provider are not as black and white as the figure might depict: the consumer can at the same time provide services thus being a provider itself, and vice versa.

In the domain of the electronic government it is expected that a large number of services will be deployed. Within a specific government organisation the need for a service registry might not be necessary at first sight, but for government-wide processes service registries (or "product portfolios") are essential.

3.1.3 Service choreography

As briefly mentioned in chapter 2, the electronic government exists of government-wide (business) processes which will make use of services provided by several government organisations and departments. These processes run according to a certain workflow. A workflow is an invocation of services in a specific order, forming a so called (business) process. There are two ways for a workflow to take place. One way is the service choreography and the other is service orchestration. In the service oriented world a workflow is often called a (business process). This thesis will use both terms.

Service choreography is a collaboration between a collection of services to achieve a common goal. The choreography constitutes an agreement on how a given collaboration should occur.

It captures the interactions and the dependencies between these interactions of the participating services to achieve the common goal [1]. A commonly used analogy are dancing partners who work together to perform the dance. In Figure 3.2 service choreography is illustrated.

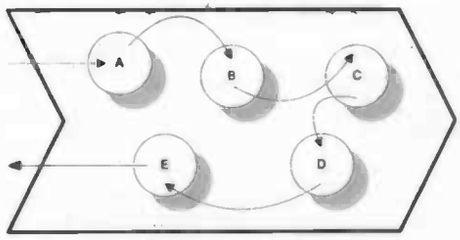

Figure 3.2: Services that form a process by choreography.

In practice this technique is not preferred to create business processes. In the case of the electronic government this technique is not considered as best practice because the individual services will lose their context independence and thus limit the reuse of the services in question. This does not mean service choreography is not used. Services can be choreographed to act as one single instance, loosely coupled, discoverable and context independent "composed service".

Service orchestration is a preferred practice for electronic government processes. This technique is introduced in the next section.

3.1.4 Service orchestration

The workflow of service orchestration is not defined in any way in the services itself. As the term "orchestration" already suggests is the workflow defined at the so called orchestrator. Industry best practice prefers this process flow to be defined in the "Business Process Execution Language" (BPEL). In BPEL complex sequences of service bindings and invocations can be defined to form the processes. These processes are services in their own right and can therefore be part of even larger orchestrations. Service orchestration is illustrated in Figure 3.3.

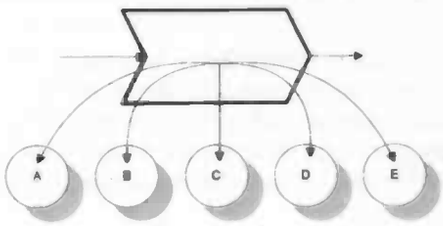

Figure 3.3: Services that form a process by being orchestrated.

Orchestrating services instead of choreographing them will keep the context independence of the individual services and is therefore preferred for use in an ever growing and complex electronic government where service reuse will be a hot issue.

3.1.5 Architectural Layers of an SOA

Given the characteristics of the service oriented architecture, one may still be wondering on how exactly this is going to help the electronic government? The basic idea is to build an SOA on

top of the existing governmental ICT systems. This means that most of the existing systems will stay in function and will be used like nothing has changed. The SOA approach will absorb the systems into one big interoperable and collaborating system. To illustrate how this works, one can say that an SOA can be seen as being built up out of seven architectural layers [2].

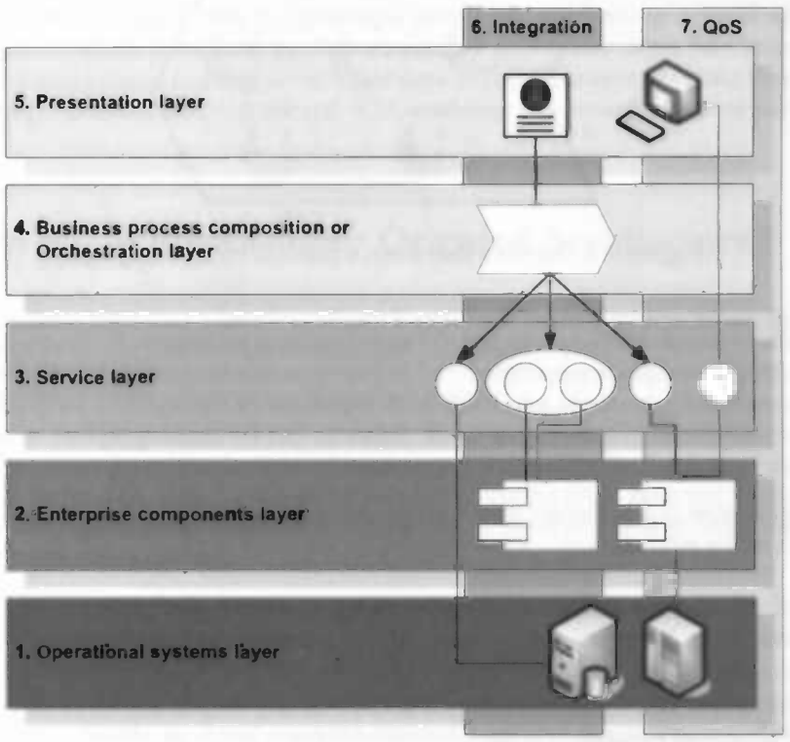

Figure 3.4: The seven layers of a service oriented architecture [2].

- **Layer 1: Operational systems layer.** The lowest layer consists of existing (legacy) systems. A legacy system (or application) is an in-place structure that is neither optimal for modern needs nor modifiable for project purposes. Legacy systems are generally wired into an organisation in a very substantial way [9]. These systems can be "absorbed" in the SOA using service-oriented integration techniques, for example providing specific functionality to the outside world by creating a web service on the system. These systems can be custom build applications used for specific governmental processes, or commercial ERP or database systems.
- **Layer 2: Enterprise components layer.** In this layer the actual functionality of the service reside. This layer typically uses container-based technologies such as application servers to implement the components, workload management, high-availability, and load balancing.
- **Layer 3: Services layer.** The actual exposed services are situated in this layer. These can be services that provide functionality which runs on the enterprise components of the previous layer, or interfaces in the form of service descriptions exposed from the first layer. Some services are built out of other services to form a bigger one. These are called the composed services (service choreography).
- **Layer 4: Business process composition or orchestration layer.** In this layer the (composed) services from the lower layer are orchestrated. The services are bundled into a workflow by orchestration to form a single application, business process or use case. Normally the

orchestrations manifest themselves as web services so that they can also be part of other larger orchestrations.

- **Layer 5: Access or presentation layer.** The services in the previous two layers are not really usable when a human user has to work with them. Therefore this layer is concerned with the interaction of a user with the underlying orchestrations and services, and the way of presenting this interaction to the user. The user interface can for example be a portal website of a government agency where citizens can apply for a permit.

- **Layer 6: Integration.** The sixth layer is concerned with the integration of the services. In practice this is called the ESB which stands for the Enterprise Service Bus. The ESB will provide a single bus which all services can use to connect with each other. The ESB also introduces a reliable set of capabilities such as dynamic context based routing of data to and from services and data transformation methods. It is not uncommon that the ESB contains mechanisms that ensure a certain quality of service. This layer can therefore have some overlap with layer 7.

- **Layer 7: Quality of Service.** The seventh layer of a SOA is concerned with monitoring, managing and maintain a wide range of quality attributes. Most of these attributes are housed in web service standards, commonly referred to as "the WS-*". This layer provides tooling and background services to ensure the quality of service of the SOA applications and services.

3.2 Why use a Service Oriented Architecture?

A service oriented architecture is very useful for a network organisation [11]. A network organisation is defined as:

> A network organisation is a fluid horizontal organisational form in which collaboration and knowledge sharing between internal or external parties takes up a central role to achieve better innovation, a faster response on market developments and better suits the customer's needs.

Comparing this definition to the electronic government's fundamental principles from Chapter 2 an overlap or match between the definition and the principles can be seen. Overall, there are five main reasons for a network organisation to deploy an SOA for its systems. All these reasons are applicable to the electronic government and match with many of its fundamental principles. These five reasons are described as follows:

1. **Optimal utilisation of the existing ICT investments.** By opening up existing (legacy) applications with web services the applications can go up into the SOA ICT-landscape. Opening up these (legacy) systems and taking them up in a SOA-landscape will keep these viable assets in the organisation without having to try to replace them (which is probably a costly affair).

2. **Low investments for new ICT.** When most of the functionality of the organisation's systems is available via web services, this functionality can be reused. New applications consist of large amounts of this "reused functionality". This results in lower costs and less development time.

3. **Integration of existing ICT.** When all applications and systems are opened up by web services they can be integrated on a higher level to form one big collaborating system.

4. **Having a flexible system that can easily be modified if the business process requires it.** In an SOA the ICT and business processes are closely aligned (the so called Business/ICT-alignment), resulting is an almost perfect support of the business by the ICT. The business processes are modelled in a way that they are easy to modify.

5. **Having a better insight into the business processes and having a good ability of monitoring information produced by these business processes.** SOA frameworks often feature elaborate systems to analyse valuable data in real-time (the so called Business Activity Monitoring). Almost all professional SOA-enabling frameworks feature such monitoring tools.

3.3 When not to use a Service Oriented Architecture

Before the thoughts arises that a service oriented approach is the solution for all software and IT challenges, an SOA might not always be the right solution.

A few considerations on this subject are described below:

1. **In a homogeneous environment.** When the systems and technologies used are of a single vendor or of vendors which offer good ways of coupling these systems "under water" and the need for exposing functionality in web services is not eminent an SOA might not be the most cost-effective solution.

2. **When tight coupling is preferred.** Loose coupling is a good solution for building applications out of functionality provided by different systems when you do not exactly know how things work on the other end of the line. However, if an organisation controls all the systems in question a tight coupling between them will prevent a maybe needless overhead by using for example an object oriented approach instead of a service oriented one.

3. **When performance is a issue.** An SOA communicates using synchronous and asynchronous web services. The loose coupling and overhead of an SOA might not guarantee response time needed in performance-critical systems such as real-time systems.

4. **When things don't change.** A popular saying *"if it ain't broken, don't fix it"* summarises this point. If the business is not going to change, and the legacy system still does its job, it might be preferred just to leave things as they are.

The points described above show some considerations when an SOA might not be the solution. It really depends on the specific situation at hand. Even then, the pros of an SOA still might outweigh the cons.

3.4 Summary

In this chapter the service oriented architecture was introduced. The basic ideas behind it, the services, service orchestration and the architectural layers of an SOA have been described. Also, the benefits of an SOA for a network organisation such as the government were shown that justify the service oriented approach for the electronic government.

Part II

Implementation of the electronic government

Chapter 4

Implementing the electronic government

The electronic government is basically a guideline to achieve a customer-centric government. Keeping this in mind, the architecture framework can be seen as a layer on top of government organisations describing a set of common services and common practices to form an electronic government that will act as a single entity. However, not every detail for every government agency can be proposed in advance because of the subsidiarity principle. Because of this principle the government and public agencies are free to choose their own implementation to achieve the goals of the electronic government.

Looking at the electronic government and all the underlying principles there are basically three main aspects to distinguish. These aspects form the basis of the proposed implementation in this chapter. From the business architecture view these aspects are identified as *customer-centric, one stop shopping* and *act as a single entity*. These aspects are each discussed in the sections below.

Table 4.1 shows a subset of a Zachman enterprise architecture for the electronic government implementation that is proposed in this chapter.

Who	What	How
Customer-centric approach.	Front office responsible for all communication with external parties.	Exposing the back office to the front office.
One stop shopping.	Product portfolio.	Specialised electronic counters ("desks").
Act as a single entity.	Reuse of common components.	Common components for authentication and authorisation, central government portal web site, common way for requesting products.

Table 4.1: Implementing the business architecture

In the end these three points form an electronic government layer on top of government and public agencies as depicted in Figure 4.7 on page 33.

4.1 Customer-centric approach

The actual products for the customer are processed in the back office of a governmental agency. This back office consists of departments with their own work domain and tasks. For the customer this situation does not contribute to a transparent and easily accessible government. Instead of finding out where to get a certain product as a customer, the electronic government should provide a front office which can redirect service requests to and from specific departments in the back office to help the customer on its way.

The front office is concerned with the different communication channels the customer has access to and which one it decides to use. Because of the multi channel approach these different channels need to be "converted" to a way of communicating that the back office requires.

In a way, one can say that the back office should expose its services to the front office. Specific work, tasks or communication which normally has to find its own way to the customer should now flow through the front office.

To streamline the communication and cooperation between front and back office a mid office layer between the two is devised [7]. The mid office forms the link between front and back office not only on communication level, but is also concerned with the difference in opening hours: the back and front office of a government agency is usually opened during working hours. However, the internet channel is available 24/7 for the customer. The mid office should provide solutions which can buffer the product requests when the back office is closed. The mid office also integrates the different departments by supplying a channel the different departments can use to cooperate and integrate.

4.1.1 The front office

The front office is concerned with communication to and from the customer. This communication can be divided into five different kinds of communication.

- **Intake.** The request for a service or product is called an Intake. This request can originate from a customer or another government agency.
- **Outcome.** The result of an intake of a request for a service or product is communicated back to the requester via the Outcome communication.
- **Inform.** General information about the government (agency) in general, or personalised information tailored to the customer's need is provided by the Inform communication.
- **Indicate.** One of the principles of the electronic government is that it should be pro-active towards the customer to point out which services the customer is entitled to. This is called the Indicate communication.

The different kinds of communication are accessible via different communication channels. These channels are mail, phone, e-mail and internet, but new channels can be added in the future if it is considered necessary. The intranet channel is only used for connecting different government agencies directly, for example by using a Virtual Private Network. For agencies which have to work together extensively this channel is ideal.

Apart from the different kinds of communication the front office also provides a service repository in which the services this particular government agency provides are registered. Other agencies and customers can then determine which services they can request and in what way.

4.1.2 Process integration in the mid office

The mid office is the integration layer between the front office and back office. There are basically three parts to distinguish in the mid office.

Message broker

While the front office takes care of the incoming and outgoing communication, the message broker of the mid office is concerned with getting this communication to and from the corresponding back office system(s). This means that data has to be converted to and from the internal data format of the message broker and the systems it connects to.

Using a message broker prevents having $O(n^2)$ connections between n systems, where each system features its own specific connection mechanism and data transformation. Instead, using a message broker requires only $O(n)$ number of connections. The data is transformed into and from the standard data representation of the message broker. The message broker ensures greater flexibility and easier system integration. See Figure 4.1 and 4.2 for a simplified depiction of the message broker.

The message broker should also close the gap between the difference in "opening hours" of the front and back office. It can store incoming messages for the back office until the back office accepts them.

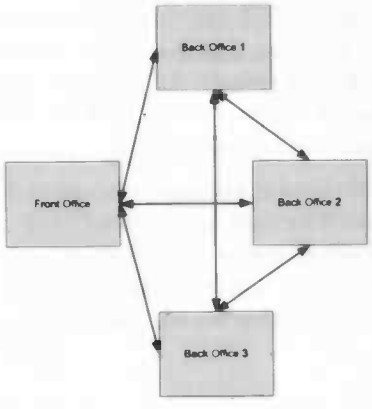

Figure 4.1: No message broker between the front and back office.

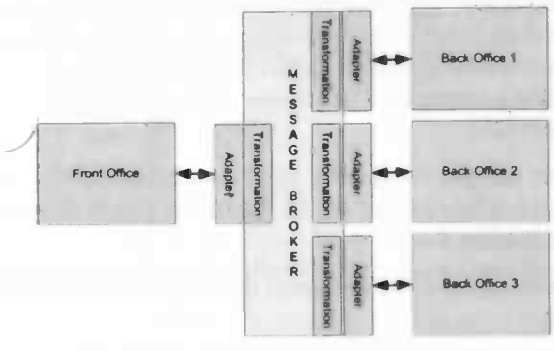

Figure 4.2: Message broker between the front and back office.

The message broker is implemented by an Enterprise Service Bus (ESB) [7]. The ESB will ease the integration of the back office processes and the front office as seen in Figure 4.2. But not only does it integrate the internal systems, it can also integrate ESBs themselves.

An ESB will expose its functionality using web services. These web services give access to an ESB instance which is basically a message flow from the entrance to the exit (again a web service), probably sending output or feedback back to the requester in the end. In the mean time the message can be transformed using XSLT transformations to make the message suitable for the message receiver.

There are basically two kinds of message exchange schemes the ESB provides:

1. **Message routing.** If the consumer party (of parties) and the provider party of an ESB instance is known, the ESB can simply route messages between these parties. Content-based routing belongs also to the possibilities when based on the content of the provider's message, this message is routed to a specific message consumer.
2. **Message queueing: publish-subscribe.** If it is not known in advance who the consumers are of a provider's message, the ESB supports the publish-subscribe model. This means that a provider publishes a message in the message queue of the ESB after which the ESB takes care of distributing this message to the subscribers of the specific message queue. The provider can "fire and forget" a message to the queue without having to know which and how many consumers will need to get the message. Using a message queue the ESB can guarantee "deliver once and only once" message delivery to message subscribers.

The message exchange schemes of the ESB provide an abstraction layer for the service consumers and providers. They only have to know that the ESB is their end point for communication and do not have to be bothered with the specific location (in the form of an URI) of their specific provider. The ESB can be updated at run-time so that the URI of a provider can be updated in one place without having to update all consumers.

Integrating different ESBs can be done in the same way as ordinary services. An ESB can publish messages intended for another ESB in a message queue of which the other ESB is a subscriber, or an ESB instance can be created which routes messages to another ESB's instance. Most commercial ESBs also provide mechanisms to create custom adapters such that customised integration can be achieved like for example creating an adapter which connects to another ESB over JMS (Java Messaging Service).

Data management

For the front office and the different back office processes and departments to work together there has to be consensus about the data they exchange and the availability of this data. It is not acceptable that some data is not always available because of different "opening hours" of the back offices. An operational data store is needed in the mid office to provide a single source for commonly used data in the government processes. It is necessary that all involved parties agree on the meaning of the data [15].

The operational data store merges commonly used data from the Key Data Stores so the data can be used by all kinds of back office processes. In Figure 4.3 this is illustrated. The Key Data Stores are further described in Chapter 4.4.1.

There are a number of reasons for an operational data store to be used instead of directly dealing with the Key Data Stores.

1. **Performance.** A Key Data Store can physically be situated on a different location than the government organisation. If every back office process from several government agencies

had to request (bulk) data from the Key Data Store itself the network or Key Data Store would be soon overloaded. For example: the Municipal Personal Records Database GBA can be accessed over a dedicated ISDN line, which probably does not have the required bandwidth to handle all requests.

2. **Costs.** The use of a Key Data Store is not always free. Developing and maintaining the Key Data Stores is a costly business and the government has decided that the users or clients of these systems pay for its use. In the case of the Cadastre for example, the costs are calculated per requested set of records. When several back office processes need to request the same data and do this on their own, the costs will be higher than when only one system, the operational data store, requests it.

3. **Data merging.** Authentic data is spread over different Key Data Stores. It is not very efficient to let the back offices themselves merge data such as for example name data with address data with property data. It is easier to have a single place where this data is already merged.

4. **Quality control.** It is not unthinkable that with the merging of data (some) inconsistencies in the data are discovered. This is especially true when the Key Data Stores are just put into service. After a while, most inconsistencies will be corrected because of the feedback duty (see section 4.4.1). It is more efficient to do the quality control at one place for all the back office systems. The operational data store should perform these data quality checks when importing data from the Key Data Stores.

5. **Privacy and security.** Access to the Key Data Stores is most of the time restricted. Most of the data in the Key Data Stores is not public, so the law on the protection of privacy sensitive data restricts public access. It is easier and more efficient to request access for one system –the operational data store– than for every back office system itself.

The operational data source should publish its contents read-only only to the systems which need these data to perform their public duty, keeping in mind the directives the law on the protection of privacy sensitive data prescribes. Possible inconsistencies or errors in the data which have been discovered during the quality control have to be fed back to the Key Data Store which is responsible for the specific authentic data.

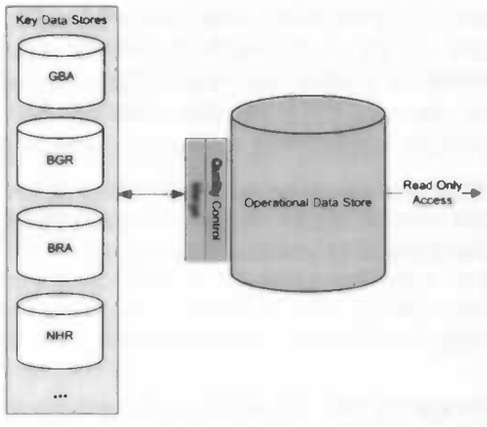

Figure 4.3: The operational data store between the Key Data Stores and the clients.

The Key Data Stores can update the operational data stores with data changes on a regular basis using the publish subscribe method of the ESB, or by using a custom or dedicated way of communication.

Apart from an operational data store which gives a common set of data for the back office and the fact that the products which the government agency provides often are spread over several back office departments, the mid office should provide a case dossier solution [15]. The case dossier contains all the needed information about the product request of a specific customer. The back office departments can complement this data with their specific contribution to the case. This case dossier should be accessible by the customer, to see the status of requested products and to get insight into the procedures and decisions that preceded the realisation of the product. This dossier can be considered as the starting point for the product requests of the customer. The product requests which arrive in the Intake of the front office have to be added to the case dossier as being a new case. After the creating of the new case the case dossier can start the specific BPEL processes to produce the product and settle the specific case. The final result of the product request will be send through the Outcome communication of the front office over a channel of choice back to the customer. This is illustrated in Figure 4.4.

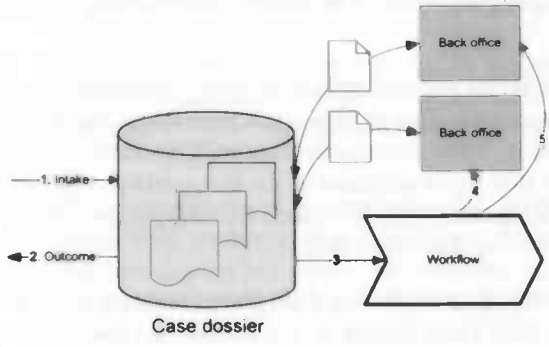

Figure 4.4: The case dossier as starting point of the product realisation.

From all the different communication channels the front office has to support, only the internet channel can be directly linked from Intake to case dossier. For the other communication channels the case dossier has to provide a way to a content management system so that front office employees can convert incoming product request on channels other than the internet one to a case in the case dossier.

The data in the case dossier should be able to be shared with the customer. After identifying the customer personal case data can be sent through the Inform communication in the front office to the customer. Again, only the Internet communication channel can perform this operation automatically, for the other channels, the front office must use the content management solution to provide the customer with the needed data via the other communication channels.

The case dossier contains data that should be archived according to the law on archiving of official government documents and communication as introduced in Chapter 2.2. The case dossier must conform to the rules the law dictates in relation to archiving.

Workflow

The products a government agency provides for its customers often require the contribution from several back office departments. The workflow system should work closely with the available mid office systems and the back office systems in question.

The workflow for the realisation of a product contains automated steps (e.g. getting information from different systems or databases) but also manual ("human") steps when for example a certain process step requires human approval from a government body or the step is too specific for the case in question, so that it can not be automated.

In the service oriented world the Business Process Execution Language (BPEL) is considered as the standard for orchestrating web services to create workflow. The Dutch electronic government reference architecture therefore suggests the use of BPEL for orchestrating web services [7].

BPEL is a business process orchestration language based on serialised XML. It is used to orchestrate web services and features local variables, fault handling, flow handling such as "if-then-else" structures and parallel processing. Basically it describes high level state transitions in processes built from stateless web services.

In the case of product requests the BPEL processes will orchestrate the workflow between the different back office applications or departments and the case dossier. The workflow not only orchestrates systems, but also contains the "human workflow" where certain tasks are delegated to workers as part of the workflow and its deliverables. Human workflow is not part of the BPEL specification, but is implemented using web services. These synchronous services stop the BPEL process and delegate a task to a person. Because the service is synchronous the BPEL process can only continue when the service is finished, and that is when the human task has been completed.

4.1.3 Exposing the back office

The back office is being portraited as the place where the specific work is done and that it should expose functionality to the mid office. This means that applications used in the back office expose functionality to integrate nicely with the workflow, operational data store and workflow of the mid office.

Not only the applications and systems of the back office need to be adjusted to the whole electronic government idea, the people of the back office also need to be aware of their part in the specific process and need to work accordingly [7]. Their work consists of specific deliverables in the whole process which must contribute to the case dossier.

For exposing functionality from the back office the applications of the back office need to open up in terms of providing web services and the employees in the back office should be process aware and see their tasks as an element of a chain which forms a product.

There exists a wide range of possibilities when exposing the back office applications with web services. A normal 3-Tier application consisting of a presentation, (business) logic and data layer can attach web services at any of these layers. To preserve consistency in the data layer the business logic layer is the preferred place to attach a web service. If the application in question is an Oracle Forms application for example, the preferred place to build a web service is by exposing internal PL/SQL procedures of the business logic layer.

Not all applications are open enough to build web services upon. If this is the case one must see whether or not to replace the application by another, or to circumvent the whole web service building. In this case the workflow system should feature "human tasks". Simply put, the human task is a web service which blocks the progress of the workflow until this task is done. It consist of a task a human can claim, manually perform, and finish after which the workflow continues based on the outcome of the human task. In this "ticket"-like system a back office employee can contribute its piece to the product without having to rebuild the application it uses to perform its work.

4.2 One stop shopping

The term "one stop shopping" means products that are spread over more than one department or government organisation can be offered and requested as being one product. Instead of a customer going to each of the departments or organisations himself to obtain the parts of the

product or service, the customer should be able to request the product as a single unique request. The government should "assemble" the product from all the different sub-products and services of the different departments or agencies for the customer. If a customer wants to start a factory for example, does not want to be bothered with all the different permits needed to request, like building and industrial waste permit, but wants to request everything at one place, in one go.

To achieve one stop shopping the services and products a government organisation provides should be catalogued. The front office, which handles all communication, should provide a service catalogue. Government organisations can use the products provided by other agencies to provide one stop shopping for their customers. In a way the front-mid-back office solution as described in the previous section forms the basis for the one stop shopping idea.

4.3 Product portfolio

For the government to take on a pro-active role to indicate to the customer that they have right to a specific product, the product portfolio must be able to match a customer with the services in the portfolio. This means the different service catalogues or portfolios of the government should work together.

The personal internet page on the government portal web site, the digital counters and standard products will search the portfolios with customer-identifiable information to find the products the customer requests or services which provide information relevant to the customer. Products should be described with meta data describing the product, cost, and for whom it is destined to be used.

These portfolios export data about their services in an XML file, after which it is imported in a government-wide index (the Common Index). This index can be searched by the customer on the government portal, personal internet page or government agency website and yield product information including an URL so that the customer can request the service electronically. In Figure 4.5 this is illustrated.

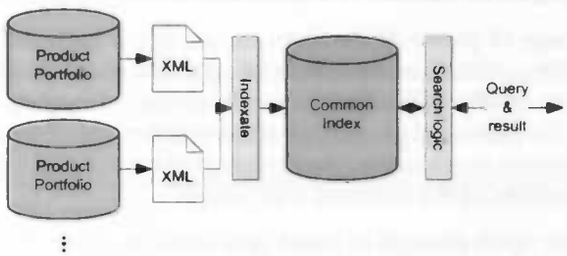

Figure 4.5: Product portfolios and the common index (simplified from [3]).

The exported XML file contains meta data about the services the specific government agency has to offer. This meta data is based on the Dublin Core Standard and will describe the same set of meta data for all services in all product portfolios to prevent semantic inconsistencies [7].

4.3.1 Digital counters

For the common products the government provides such as changing the customer's address, but also requests for certain permits or information about the neighbourhood, specialised digital counters should be provided. In this way the customer has one place to do business with the government, and this one place takes care of everything.

Digital counters enable the customer to do their one-stop-shopping for a product request at a central place. There are several different counters being developed, for example a counter where every aspect concerning building a property can be handled.

The product portfolio plays a central role in finding the products and services available, whereas the digital counter groups these products and services together so that the customer has a central place where to do business with the government.

4.4 Act as a single entity

Despite of the sheer amount of different sovereign government agencies and organisations, the electronic government should present itself to the customer as a single entity. In a way this is already settled by the role of the front office, which shields the customer from the internal organisation of the government. However, the usage of common components will contribute to form a single way to interact with the government. Examples of these components are a single internet portal website for the complete government, one common digital identity and a standard way to request products or lodge complaints by using uniform electronic forms. Beside these components, the Dutch electronic reference architecture NORA and proposed website style guidelines promote the electronic government as a single entity.

4.4.1 Key Data Stores

The IT supporting the processes within the government are often developed independently of each other. Customer data is scattered over these systems. The data and information these systems need about the customer have to be presented every time a customer has to deal with these specific systems.

In Chapter 2 one of the recurring basic principles is lowering the administrative burden and encouraging the reuse of data. From a customer's point of view, acting as a single entity also means not asking for information already known by the government. For this reason the Key Data Stores are devised by the government.

A Key Data Store is a central database which contains authentic and non-authentic data. They are considered as the source for authentic data, and usage will be mandatory by law. The non-authentic data is data which is present in the specific Key Data Store but is not categorised as being authentic data, meaning it is authentic data copied from another Key Data Store. The Key Data Store is the only place the authentic data can be added or changed. The Key Data Store is responsible for the quality and correctness of its authentic data. Users of the data imported from a Key Data Store have the duty to report possible errors in this data. The Key Data Store must investigate these possible errors. This so called feedback duty makes sure that the quality of the data in the Key Data Stores is maintained.

The information from the Key Data Stores is accessible from the mid office and in this way available for all back office processes. This ensures data reuse within the whole government and makes it act as a single entity.

The first six Key Data Stores which will be implemented in the years 2007 to 2009 are:

1. **Gemeentelijke Basisadministratie Persoonsgegevens (GBA).** The GBA is the Municipal Personal Records Database which is already around for quite some time. It is promoted to the status of Key Data Store. It contains authentic personal data about all the citizens in the Netherlands, such as name, date of birth, social security number, sex etcetera, and non-authentic data as the address.

2. **Nieuw Handelsregister (NHR).** This datastore for the trade and business community contains data about all the companies and businesses in the Netherlands, their company number etcetera.

3. **Basis Gebouwen Registratie (BGR).** Al the houses, industrial buildings and monuments are contained in the Building Key Data Store with all their identifying properties.

4. **Basisregistratie Kadaster (BRK).** The Cadastre contains all the administrative data about properties, parcels (premises) and their values.

5. **Basisregistratie Topografie (BRT).** The whole of the Netherlands is charted on a scale of 1:10.000 and stored in the Key Data Store for Topography.

6. **Basisregistratie Adressen (BRA).** All the addresses in the Netherlands are stored in the Key Data Store for Address information.

There are four more Key Data Stores identified which will be implemented in the future and several more possible authentic registers are being investigated to become Key Data Stores. For an overview of the Key Data Stores and their relation to each other is shown in Appendix 8.4.

Access to these Key Data Stores can happen in various ways. It actually depends on the maintainer of the data store. Some GBA data stores require a dedicated ISDN line, whereas the AKR data store can be accessed using web services.

4.4.2 Digital identity

The electronic government needs to know who it is dealing with. Therefore a digital identity for its customers (in this case citizens and companies) is needed to provide a uniform way for authentication and authorisation. In the case of the Dutch government "DigiD" is developed for this cause. The customer can request a DigiD login from the central government agency which maintains DigiD.

The DigiD identity is linked to the BSN number for persons, meaning that after logging on using DigiD the BSN is known to the electronic government. This BSN number is used in various products and services of the electronic government.

Companies do not have a BSN, but a similar number: the company number. This number is used in the same way as the BSN number.

4.4.3 Personal Internet Page

To give the customers a starting point for their interaction with the government a central portal website for the electronic government is devised. Here the customers have access to the common components of the government as introduced in Chapter 4.2.

The personal internet page features a personalised portal for the customers. After login they should be able to get personalised information and see the status of their products and product requests. This is also the place where the customers can access the common solution like digital counters (see Chapter 4.3.1).

The personalised information is gathered by searching the electronic government Common Index with the BSN of the customer. In this way all the relevant information for fthe specific customer is gathered, resulting is a pro-active electronic government.

4.4.4 Electronic Forms

As a customer, requesting products electronically is done in a uniform way. The so called electronic forms allow the electronic government to provide a common way to let customers request products. These forms are in wizard form: they feature several wizard steps that are common for different product requests. Wizard steps like identification and personal data are the same for all the electronic forms. For the visual style there are style guidelines just to make the electronic government "show one face".

Figure 4.6 shows an example of a step in the wizard of an electronic form.

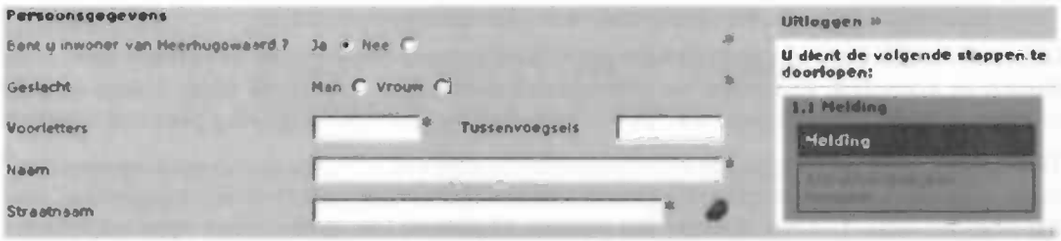

Figure 4.6: Example of a step in the wizard of an electronic form.

4.4.5 Digital dossiers

The digital dossier (or "electronic dossier") contains information about the customer. On the Personal Internet Page information from these dossiers focused on the specific customer (customer-centric information) should be made available to give the customer insight into its product requests, information the government has about the customer etcetera. There are several dossiers in development, for example the Electronic Patient Dossier and Electronic Education Dossier.

For the customer to get access to this information from the government portal web site the product portfolio can be used for example. After the customer logs on, the Common Index can be searched for digital dossier services available to the customer. These services can then be accessed using the BSN of the customer to provide the customer with the information it wants.

4.4.6 Payment methods

When products are not free, the customer should be able to pay using a common payment method. The electronic government is in this way not so different from a normal web store. A wide range of electronic payments methods or money transfer by bank should be provided.

4.5 Security and privacy

Exposing functionality is normally hidden away in dedicated applications and systems, so making it available for others calls for security measures to prevent unauthorised access to data and information.

4.5.1 Access control

First there is the DigiD authentication and authorisation service which enables customers to identify themselves. A central government organisation maintains DigiD and makes sure that every Dutch citizen or company can obtain a unique DigiD identity.

Within a government organisation access control to case dossiers and the operational data stores should be provided by internal access control like Novell iChain, LDAP or (other) Single Sign On solutions.

4.5.2 BSN

Communication between government organisations and components of privacy sensitive data should be according to the law on the protection of privacy sensitive data. Usage of the BSN number is preferred because it is a "privacy insensitive" method of sharing personal information.

The BSN (Burger Service Nummer) is numerically equivalent to the social security number. This number is unique and every Dutch citizen has one. This number is "privacy insensitive" because the number alone can not identify the person. However, the government agencies which have access to the GBA Key Data Store can, because there the link between the BSN and the person in question can be made.

In theory the government systems only have to store the BSN in their systems instead of name, address etcetera when the agency needs to store personal information about their customers. By doing this, a certain level of privacy is guaranteed and the integration of the systems into the electronic government by opening up the application is more easily.

Identifying information about companies is not protected by the law on the protection of privacy sensitive data. However, a similar number for companies exists as introduced in section 4.4.2. The usage is equivalent to the usage of the BSN number.

4.6 Summarising: the complete picture

In this chapter more details of the electronic government are introduced. It centers around the the ability to find products and services that government and public agencies provide for the customer. On the government portal web site customers can be served with products and services they are entitled to by searching the Common Index and giving access to these products and services in a central place such as the Digital Dossiers and Counters. It can be summarised as in Figure 4.7.

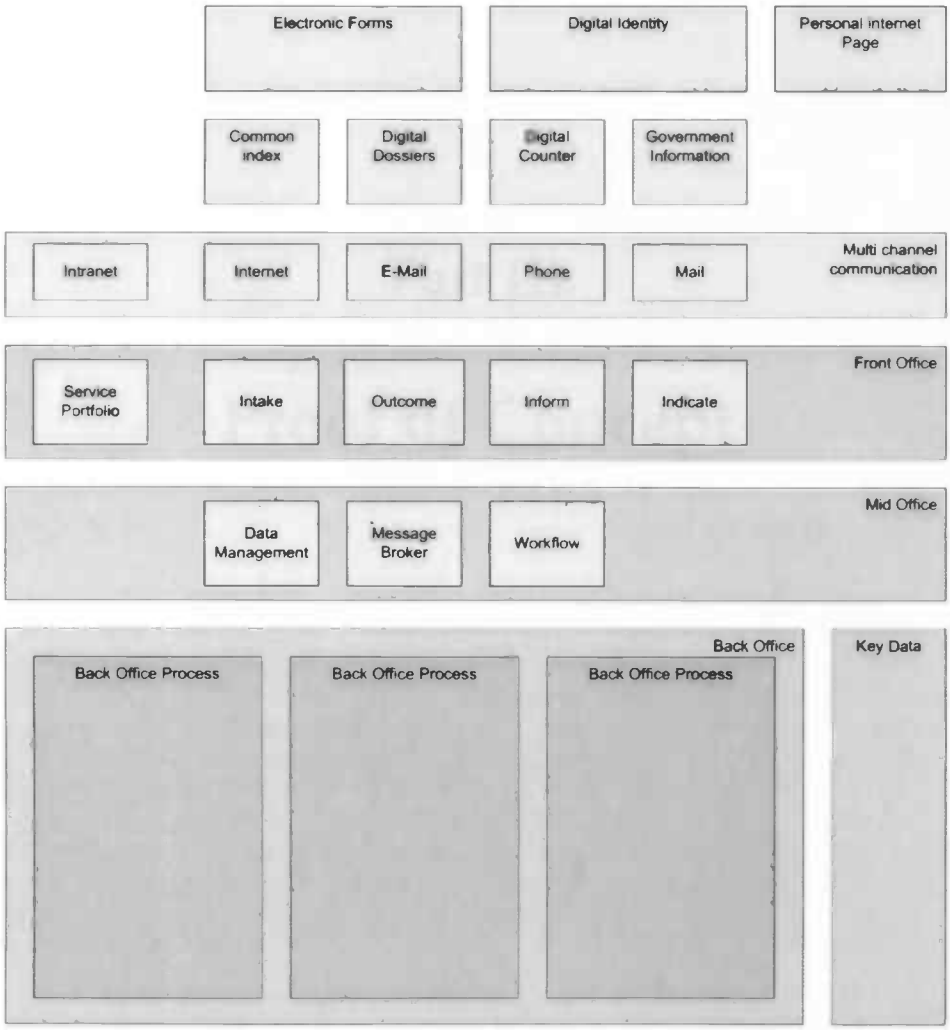

Figure 4.7: The electronic government layers.

Part III

Proof of Concept

Chapter 5

Proof of concept: the GemTax client system

In the previous chapters the electronic government was introduced and several implementation details were proposed. A client system of the electronic government is a system which will participate in a governmental back office process, or is concerned with a specific service or product of the electronic government implementation itself.

GemTax is a product developed by Vertis BV. It is a software solution for the billing and collecting of communal taxes.

5.1 Positioning the client systems

In the electronic government a client system can not be isolated from the rest. The chains of services which offer the customer agency-wide products, one stop shopping and make the government act as a single entity, requires more context independent client systems. The systems and the employees of the departments have to work more in a customer-centric way, providing services or products. For the client systems this means that they have to provide certain functionality for the other systems to use. Opening up the back office systems, as described in Chapter 4.1.3, is of course applicable for client systems situated in the back office.

A client system does not necessarily have to be a system for use in the back office. It can of course also be a system for use in the mid office or front office for example. Where ever the client system is situated, it needs to be aware of the inner workings of the electronic government. Because of the subsidiarity principle, as introduced in Chapter 2.1.3, the actual implementation of the electronic government can be very diverse. A client system must allows to be integrated using mid office solutions and must work together with the common IT solutions such as the case dossier and the product portfolio when available. A client system must therefore be open to be included into the electronic government implementation.

5.2 System for communal taxes

GemTax is a system for the billing and collecting of communal taxes and will be "SOA-enabled" by means of proof of concept to fit into the electronic government. Communal taxes are taxes for citizens and companies of a municipal authority. The most common taxes are property and

house refuse taxes. The billing and collecting can be done by one back office department or by two, when the billing and collecting are two different processes within the municipal authority.

The billing is done by aggregating the necessary data about a person or company and using this data to calculate the taxes that have to be paid. The needed data originates from the Key Data Stores and other data sources and the calculation model for determining the amount of taxes is in the system itself. The calculation model and the amount of taxes to be paid varies per municipal authority.

5.3 Different subsystems of GemTax

GemTax consists basically of two more or less separate systems. On the one hand there is BAS ("Basis Administratie Systeem"), the operational data store, and on the other hand there is HEIN ("Heffen En Innen"), a tax billing and collecting system. See Figure 5.1 for a simple depiction of GemTax.

BAS is an Oracle 9i database with user interaction by provided Oracle Forms. HEIN is an Oracle e-Business Suite (EBS) system which is, bluntly said, also a big database with a large number of tables on which user interaction is provided by Oracle Forms. BAS and HEIN are coupled with the use of Oracle InterConnect.

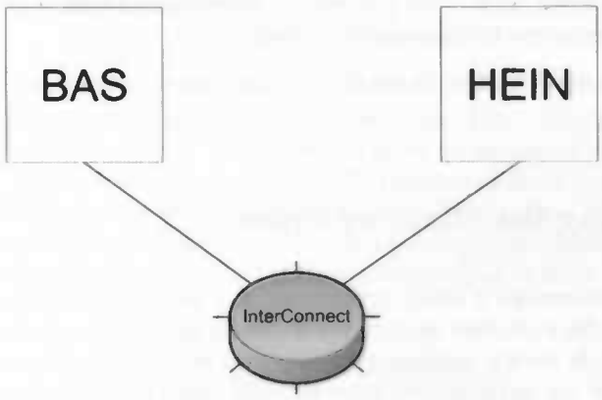

Figure 5.1: Simplified overview of the GemTax system.

One can say that GemTax basically consists of two databases. The reason for this is that BAS can be used as a mid office solution within an organisation as an (or "the") operation data store, and HEIN as the back office solution for tax billing and collecting. This means that BAS is intended to be used by other applications than HEIN alone.

GemTax is a product which is constantly in development. The design decision is made that GemTax (and its subsystems) will be a considered COTS (commercial off-the-shelf) product. This means that the proof of concept will be build on top of GemTax while not changing anything in the existing product itself.

5.3.1 BAS

BAS is an operational data store. It gets its data from external sources and combines it in its core. Before it reaches the core different stages have to be completed in which numbers of (quality) tests are performed. In Figure 5.2 BAS is depicted.

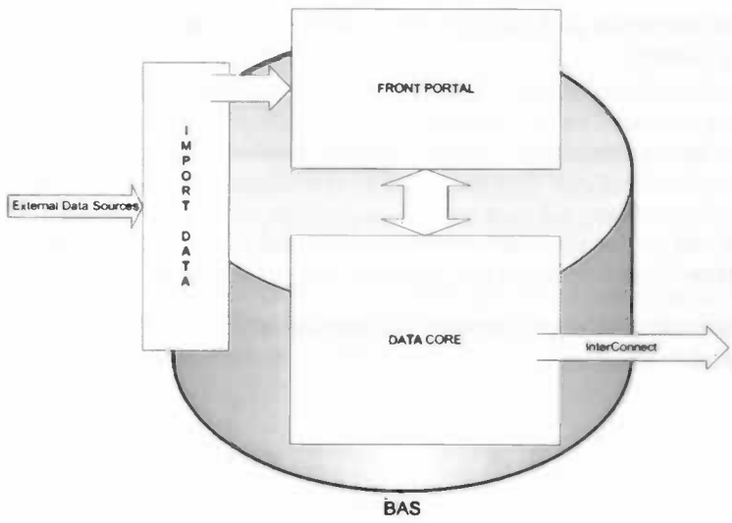

Figure 5.2: Simplified overview of the BAS subsystem.

BAS features an elaborate user interface for manual quality checks, importing data but also geometric information services such as displaying the data in a map etcetera.

The Import Data stage

BAS imports its data from a number of sources. Most of this data is imported by using third party tools. These tools can shape the data into the form that BAS can handle. However, there can also be other reasons for the use of third party tooling. Often, only official authorised clients may connect. Also, using the Key Data Store is not free and rather complex, so by using third party tooling this is circumvented.

For the Cadastre data third party GisKit tools are used which can read the Cadastre output files and place them in the database tables of the import data stage. For getting data from the GBA Key Data Store an authorised tool made by CompeT&T is used which connects by ISDN to the Key Data Store and directly imports the data into BAS.

BAS imports data about subjects, premises (administrative and geometrical data), and WOZ Objects (data needed for property taxes). This data comes from Key Data Stores GBA, Cadastre (here called "AKR", but this will change into "BRK" in the near future), the WOZ data store and the LKI. The LKI contains geometric data of the premises. The WOZ data store will probably become a Key Data Store in the near future [6] and the geometric data LKI will become part of Key Data Store for Topography. Table 5.1 shows which data is imported from what source. The X indicates the data that is imported while O indicates this data is present in the source, but not imported.

Source/data	Parcel Administrative	Parcel Geometric	Addresses	Subject	WOZ Objects
GBA			X	X	
AKR	X		O	O	
LKI		X			
WOZ			O		X

Table 5.1: The imported data and its sources.

When the data is imported and merged in BAS a series of tests are performed on the data. This will ensure data quality.

Front Portal

Before any data can go the data core of BAS it is stored in the Front Portal. Here data inconsistencies ("ascertains") are displayed and data errors can be solved.

Also, when data from the Data Core needs inspection or editing, it is moved from the Data Core to the Front Portal.

Data Core

After the Front Portal the data is stored in the Data Core. This core consists of the "truth" and its history. Only data from the core is eventually passed on to other systems. Every data record has a start and end date. In this way the history of the data is stored.

5.3.2 HEIN

HEIN is built as an Oracle Application in the e-Business Suite. EBS consists of a large set of stock modules that can be programmed and configured to form an extensive software solution for a wide array of business processes. Roughly said, HEIN consists of three main modules: the CRM (Customer Relation Management), the Finance and Custom modules. These modules are connected (and share information and data) using Oracle's Trading Community Architecture (TCA). HEIN gets all its data from the InterConnect pipeline originating in BAS. HEIN is illustrated in Figure 5.3.

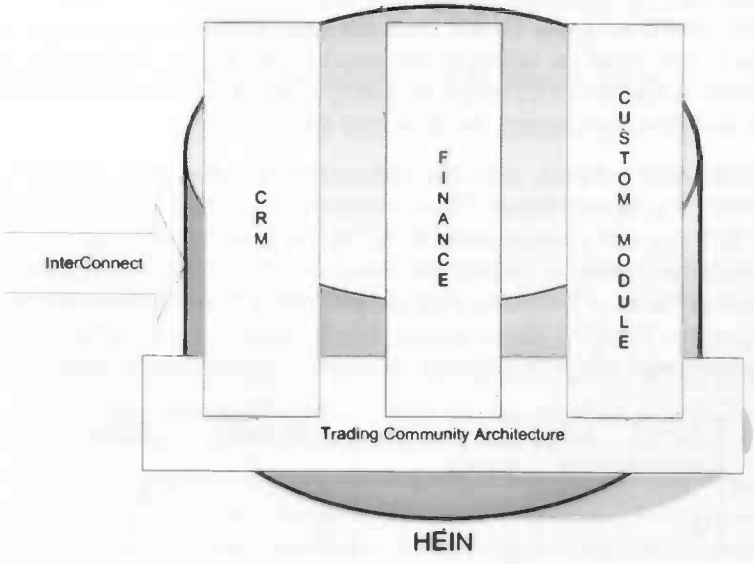

Figure 5.3: Simplified overview of the HEIN subsystem.

5.3.3 Oracle InterConnect

To get data from BAS to HEIN Oracle InterConnect is used. Oracle InterConnect is an integration solution to integrate heterogeneous systems. It is a message broker based on the Hub and Spoke model. It features adapters which connect InterConnect to applications and converts messages to and from the internal data representation of Interconnect. Under the bonnet InterConnect features an Oracle database with Advanced Queueing so that the publish-subscribe method can be utilised.

The way InterConnect works is basically that an application sends messages to its adapter which puts it in the message queue of the Oracle database. Subscribers to it and transform it into their own data representation by using their adapter.

5.4 The Oracle SOA Suite

GemTax is a product based on Oracle software products. The Vertis BV company is as an Oracle Partner specialised in Oracle software. Therefore the product of choice for opening up the GemTax product is by using Oracle's Service Oriented Architecture Suite (SOA Suite).

5.4.1 SOA Suite components

The Oracle SOA Suite contains several components. The components used in the proof of concept will briefly be introduced. the proof of concept is build using SOA Suite 10^g version 10.1.3.1 Developers Preview edition.

Oracle's SOA Suite is an end to end solution for creating service oriented solutions within an organisation. It provides all the software solutions needed in an SOA together with a single Integrated Development Environment to implement the SOA. Looking at Figure 3.4 on page 16, where the seven layers of an SOA are depicted, the Oracle SOA Suite is in all layers, except the first.

Oracle BPEL Process Manager

Oracle BPEL Process Manager is a 100% native BPEL engine. The processes run as BPEL on the engine. The BPEL processes manifest themselves as web services to ensure interoperability with other systems. The BPEL processes can also be invoked through Java APIs, for which proprietary Oracle components must be used.

Designing the BPEL processes is done by the BPEL designer which is a development tool in the Oracle jDeveloper development environment. The BPEL designer is an acquired product from Collaxia and can be considered as mature product. The BPEL Engine has standard out-of-the-box process dehydration: it stores long running processes in a database to ensure recoverability.

The BPEL Engine provides a web based management console that allows every process that is currently running or has already completed, to be debugged, analysed, stopped or deleted. The complete BPEL Process Manager and engine runs on the Oracle Application Server and is implemented using Java technology.

Changes in the BPEL processes, or new versions, will be handled in a "versioned" manner, and will not disrupt the running processes. The new processes or new versions can be independently tested because processes can be called using the specific version number in the call. Only one version of the process is the default one which will be invoked when no version information is provided in the call. In this way different versions of the same process can run at the same time.

Oracle Enterprise Service Bus

The Oracle Enterprise Service Bus is Oracle's message broker in the service oriented world. Oracle supplies series of adapters for connecting with the ESB: for Oracle products (database, Oracle Applications etcetera), JMS, email, FTP, Files, PeopleSoft, Tuxedo, Siebel, JD Edwards, SAP, CICS, IMS and TPF. All adapters conform to the Java Connector Architecture (JCA) 1.5 open standard. JCA allows developers to create custom adapters for connecting "exotic" systems. Oracle ESB supports the industry-standard protocols as web services, SOAP, JMS, HTTP(s) and JCA. The ESB is also fully compliant with industry security specific standards such as SSL, WS-Security and S/MIME to encrypt data.

The ESB runs in the Application Server of Oracle just as the BPEL Process Manager. It is also implemented in Java.

Oracle Application Server 10^g

The Application Server is used in many products of Oracle. It is basically an Apache web server which hosts the so called Oracle Containers for Java (OC4J). In these containers Java applications and web services can be deployed.

The OC4J containers can also be used for presenting user interfaces. If a BPEL process features a human workflow service the user interface is automatically generated and deployed to the Application Server.

Oracle Web Service Manager

The Web Service Manager is a solution to manage web services. It features a Policy Manager, Enforcement and a Monitoring Dashboard.

It can enforce policies concerning security or operational issues by providing gateways that manage access, message encryption and decryption for a group of web services. These security specific features therefore do not have to be implemented in the web services themself. Besides web service gateways the Web Service Manager can provide agents to enforce additional, fine-grained level of security by plugging directly into an service.

The Monitoring Dashboard enables the user to collect data, change and inspect policies, gateways and agents.

Oracle jDeveloper

The development tool jDeveloper is used for designing and building all SOA-related solutions. It features a complete integrated development environment for designing for example BPEL processes and ESB instances.

It comes complete with stock web services which can be used in the BPEL processes or Enterprise Service Bus instances. These stock web services include services for XSLT transformations, human workflow web services, e-mail functionality as a web service etcetera.

Chapter 6

GemTax SOA extension

Fitting the GemTax product into the electronic government a new architecture for the "SOA extension" has to be made, compatible with the vision the Dutch government has defined.

6.1 Rationale

With the Dutch government promoting the service oriented architecture approach for implementing the electronic government and the impending deadline for finishing the development, local governments see the need for service oriented products. It is expected that the municipal authorities will not buy "normal" applications any more; the applications need to be SOA-compliant.

The idea that more and more information needs to be available online is almost becoming a commodity. Citizens and companies have the need to do their business with the government using the internet, and want to have insight into personalised information provided on web sites.

GemTax has a great amount of potential interesting information inside. Giving people insight in their tax payments or due taxes is a service GemTax can and must provide. Because GemTax is built on an Oracle platform with Oracle tooling, it would be possible to migrate it to the SOA-enabled Oracle Fusion platform without too much of a hassle.

Apart from the advantages for the end-user, SOA-enabling GemTax and placing it in the electronic government SOA-landscape comes with the advantage that the product can make use of the Key Data Stores in an easy way. GemTax should make full use of this and should obey the law considering the usage of these data stores.

6.2 Architectural vision

> To provide a system for the billing and collection of communal taxes that fits perfectly into the service oriented landscape of the electronic government, and thus is compliant with the requirements and recommendations the Dutch government provides.

The success of the electronic government will largely depend on the availability of the data for others. The reference architecture for the electronic government is based on SOA and therefore client systems need to be SOA-compliant to integrate in this SOA-landscape.

6.2.1 Electronic government compliant GemTax

For a customer there is not much interaction with the GemTax system. It is not really necessary. The changes in the data on which the tax bills are based, are taken care of by the Key Data Stores or the responsible department or person when it concerns an other data source. This means changes in address, valuation of the property or changes in the family household are automatically synchronised to the client system GemTax. Minor changes such as the choice of automatic money withdrawal or manual bank transfer is functionality GemTax should expose. Other functionality is support for the electronic forms for lodging complaints when a customer finds a fault in his or her tax bill.

For a customer the tax bill itself and the amount of money owed is valuable information. This information can be exposed by the tax system.

Looking at the Dutch electronic reference architecture and its principles from Chapter 2.2.1 the following can be said:

1. **High quality of service.** GemTax features a helpdesk application which should be used in the front office for serving the customer for all the channels except the internet channel. For the internet channel GemTax should provide services for use with the electronic forms and information services for use on the government portal website. GemTax should be customer-centric on all communication channels.
2. **Reduction of administrative burden.** GemTax should make use of the Key Data Stores and not ask the customer for information GemTax could already know via the Key Data Stores.
3. **Transparency.** The information services of GemTax should give insight into the tax billing and collecting state, and the state of the lodged complaints (if any).
4. **Pro-active service.** The services and products GemTax provides should ultimately be taken up by the product portfolio.
5. **A government that operates as a single and trustworthy entity.** GemTax should provide helpdesk functionality in the front office and use the electronic forms for communicating via the government portal web site. Communication with the customer should be based on the BSN number or company number.
6. **Improved effectiveness of the government.** GemTax should also seek integration with as much of the mid office solutions as possible (if available).

6.3 System overview

The new architecture sees BAS and HEIN really as one on a functional level. This has a number of implications:

- **BAS is the source of the data for HEIN.** Although HEIN has its own database and largely consists of the same data BAS has, BAS is considered the source and maintainer of the dataset.
- **When BAS is updated, HEIN must follow.** BAS and HEIN need to be synchronised.
- **BAS as a data source in GemTax will not be bothered with importing data.** The mid office integrates the different systems in the back, mid and front office. BAS as an operational data store in the mid office should be integrated with the Key Data Stores using mid office integration solutions such as a message broker. This means that BAS itself should not be bothered with importing the data from the Key Data Stores itself (because this will make it a back office system): the message broker should integrate BAS and the Key Data Stores.

In Figure 6.1 the overview of the new architecture is given. For the sake of simplicity the overview is only depicted in terms of front, mid and back office.

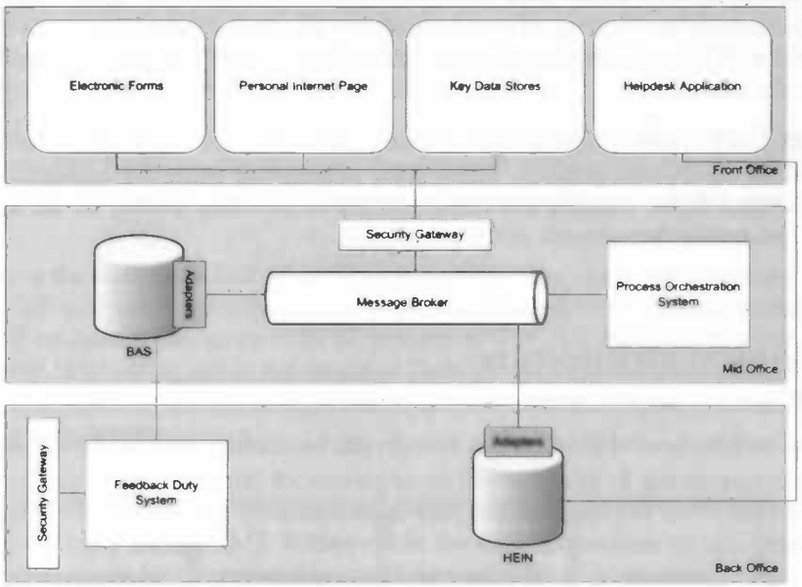

Figure 6.1: The front-mid-back office solution of GemTax.

6.3.1 Front office

The front office provides the entrance for the customers to their business with the government body. For GemTax this means that besides the help desk application which is already present in GemTax only the Personal Internet Page, Electronic Forms and the Key Data Stores are in the Front Office.

The reason the Key Data Stores are placed in the front office instead of the back office is because GemTax as a commercial client system will not know in advance how the data from the Key Data Stores is obtained. The Cadastre for example has several different ways of supplying updates, ranging from CD-ROM to FTP and e-mail. For this reason GemTax will use the multi channels of the front office for importing the data from the Key Data Stores.

6.3.2 Mid office

The Mid Office is the layer in which requests from the front office find their way to the correct system(s) or department(s) in the back office. In the Mid Office GemTax requires a message broker. This message broker can share data and integrate systems without being part of a business process. It will receive messages from the front office and route them to the process orchestration system or directly to the subsystems of GemTax.

BAS as an operational data store is located in the mid office.

A Security Gateway is present to prevent unauthorised access to the mid office. This gateway also handles security tasks like encryption and decryption of in, or outbound data.

6.3.3 Back office

The HEIN system belongs to the back office. The HEIN system may even be used in two or three different departments depending of the organisation on the back office, but that is not important in this overview. What is important is to see that BAS and HEIN are integrated using the message broker from the mid office.

The Feedback Duty System is a new system that must handle the feedback to the Key Data Stores in case GemTax discovered possibly faulty data originating from the authentic registers. It is placed in the back office because the responsibility of the data quality of the operational data store should be with a department in the back office.

6.4 Technical architecture

The technical architecture of the GemTax system can be summarised as shown in the Figure 6.2.

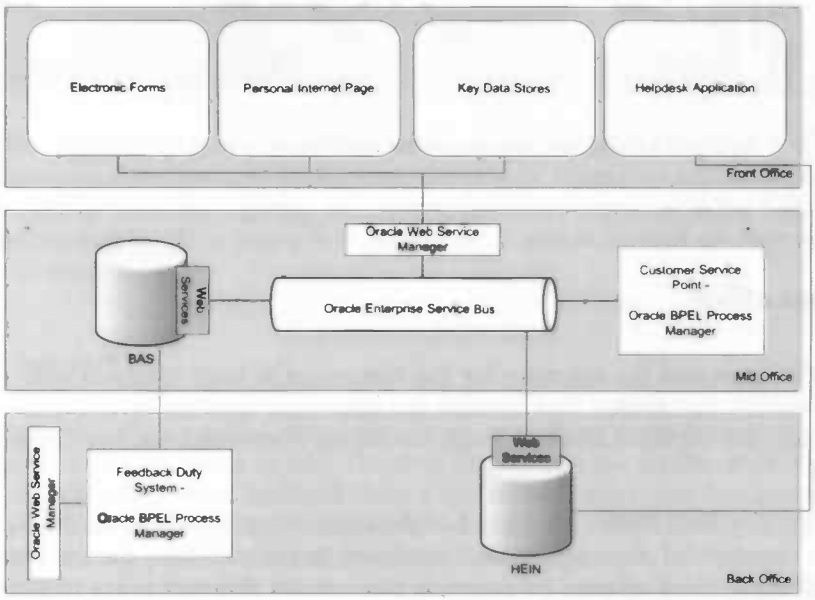

Figure 6.2: The technical front-mid-back office solution of GemTax.

6.4.1 The Customer Service Point (CSP)

Outbound service

A government portal or a web site of a municipal authority can provide its user with personalised tax information. These web sites provide in the authentication and authorisation of the user using DigiD. The web sites then invoke a web service for providing the personalised tax information in the Customer Service Point by providing the BSN of the user. This Web Service is a BPEL process which requests the information from the HEIN system.

Processing electronic forms

The Customer Service Point (CSP) is concerned with receiving the output from the electronic forms via the Enterprise Service Bus. An electronic form is a web based form hosted on a government central site where customers can lodge a complaint when they do not agree with their tax bill, or request a change in their payment method.

These electronic forms are designed by a consortium of municipal authorities. The proof of concept will contain the functionality required for the web services to change payment method, getting personalised information about their tax bills and lodge complaints as specified by the consortium.

The CSP receives the data from ESB which it received from the electronic forms as an XML and a PDF file. The CSP will automatically process the data received and will modify data in BAS and HEIN. This will be done by an array of BPEL processes. The PDF file should be saved in the case dossier, however, this belong not to the scope of GemTax because this systems was not available at the time of implementation. The later released Oracle ContentDB and RecordsDB can be used to implement a case dossier.

Electronic forms are hosted off-site, for example on the web site of the municipal authority, or the government portal. Here the authentication and authorisation of the users take place. The electronic forms output a large XML document in the case it concerns an objection form, or the BSN if it concerns a form for changing the payment method.

Distinctions can be made between three forms of electronic form output:

1. A document in XML and PDF containing all the information needed for making an objection to a tax bill;
2. A BSN identifying the customer for changing the payment method from automatic money withdrawal to manual money transfer;
3. A BSN identifying the customer for changing the payment method from manual money transfer to automatic money withdrawal.

The three distinctive forms are processed by different web services in the Customer Service Point.

6.4.2 The Feedback Duty System (FBDS)

Because GemTax uses data from authentic registers, the Key Data Stores, it has to comply with the feedback duty. The feedback duty is the duty of the client systems to report possible inconsistencies or errors in the data imported from the authentic registers back to these authentic registers. This means that until the authentic register finishes the investigation on the report, the data in question has to stay flagged as possibly incorrect and no further action can be taken.

There are three stages to identify in the event that data is possibly incorrect:

1. **Notice stage.** There is a considerable doubt that some data is correct. This can be a result from feedback received by the Customer Service Point or it is observed when new data is merged into BAS. This particular piece of data has to be investigated.
2. **Investigation stage.** The possibly incorrect data is fed back to the source (the authentic register). The source has the obligation to conduct a serious investigation and come with a possible correction within a set time limit dictated by law.

3. **Correction stage.** The authentic register has finished the investigation and sends the result back to the service that reported it, and other interested parties via a publish-subscribe protocol. It is only then that the data in BAS can be flagged as correct.

The Feedback Duty System is a set of BPEL processes which read the possibly faulty data from BAS and invoke an asynchronous BPEL process for each item which communicates with the specific Key Data Store. This item is then sent to the Key Data Store in question. The process will end when it receives the (corrected) response from the Key Data Store.

6.4.3 Data synchronisation between BAS and HEIN

In this architecture Oracle InterConnect is not needed. The Enterprise Service Bus performs the synchronisation process instead.

The communication of data between BAS and HEIN has only one direction: from BAS to HEIN. This means that if data is changed in HEIN, and not in BAS, data inconsistencies occur. It is therefore important that besides the technical changes in the system the "human factor" also changes its way of work, so that data changes are done in BAS, not in HEIN.

6.4.4 Web Service Manager

The CSP and the FBDS have to communicate with the outside world through the ESB and thus special considerations about security and privacy have to be taken into account. The Web Service Manager from Oracle acts like a proxy between the outside world and the CSP and FBDS to en/decrypt the messages. This proxy will also enforce access control so only messages from trusted sites will come through.

6.4.5 Data Import

For BAS to stay a mid office solution, the system must be nothing more than a big datastore with operational data for other systems to use. The data import and the feedback duty must be done automatically, or in the case of the feedback duty, a specialised back office department. In the mid office, BAS should be automatically synchronised with the Key Data Stores using the available technology solutions, the publish-subscribe method for example.

When BAS is used as more than an operational datastore as is described in Chapter 5.3.1, one could say it is more a back office system supporting a back office process (data warehousing) rather than a mid office solution.

The ESB comes standard with a wide variety of adapters. One of these adapters provides in automatic processing of files from a folder. This File adapter is used to automatically process the output generated by the third party tools used for importing data from the Key Data Stores. The ESB allows routing this File adapter to a database adapter in BAS such that the BAS is filled or updated.

6.4.6 Adapters

The individual systems BAS and HEIN of GemTax use Database Adapters. These adapters are manifest themselves as web services for executing queries or executing existing PL/SQL procedures in the database. These adapters can be generated using Oracle's development tool jDeveloper and are based on the JCA 1.5 standard. Using the Database Adapters the opening up of the

system can take place on two different levels: the data layer (by executing SQL database queries) and the logic layer (by executing PL/SQL procedures).

HEIN is an e-Business Suite implementation. The Oracle e-Business Suite has an enormous database with a complex data structure. Hundreds, maybe thousands of database tables contain the scattered data. The different modules have often different identifying keys, causing complex queries to link all the data in the proof of concept. This complexity is because normally the functionality of the e-Business Suite is generated and/or configured on a higher level than the data layer.

6.4.7 Oracle BPEL Process Manager

The Oracle BPEL Process Manager is a process orchestration engine which runs processes as native BPEL. BPEL describes a flow of web service calls plus some logic to form a business process. The Oracle BPEL Process Manager is an enabling technology to orchestrate web service workflow. The processes manifest themselves as web services in their own right. This allows for the creation of processes orchestrations which orchestrate other processes.

Chapter 7

Proof of concept implementation

This chapter is concerned with the implementation of the new archtecture of GemTax. Functional and technical implementation details will be described

7.1 System overview

Figure 7.1 shows the proof of concept front, mid, back office view. From first sight it is obvious that the difference is significant. To keep the figure simple, the help desk application is not depicted in the front office because it belongs not to the scope of the proof of concept.

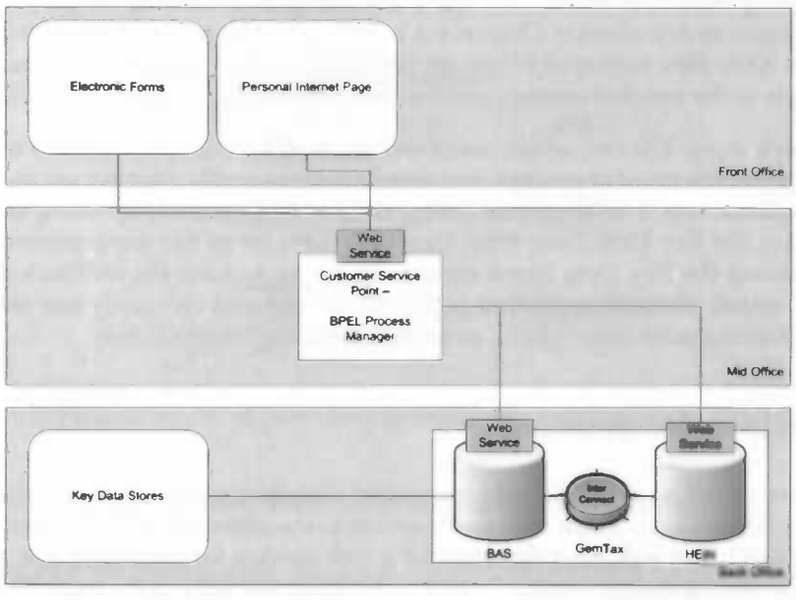

Figure 7.1: The implementation of the proof of concept.

Despite the differences, almost all the points from Chapter 6.2.1 still stand. GemTax, as a product which is in development from the very beginning of the Dutch electronic government ideas, is extended with web services such that it becomes a customer-centric product which makes use of electronic forms and has the potential to be fully incorporated in an electronic government organisation when the enabling technologies mature.

7.1.1 Differences with the suggested architecture

There are basically six main differences:

- **No Enterprise Service Bus.** The Customer Service Point talks directly to the GemTax subsystems without an Enterprise Service Bus. Ideally the Customer Service Point would have initiated an ESB instance which would communicate with the BPEL Process Manager or web services of BAS and HEIN to ensure a certain level of quality of service. However, the ESB proved very unstable, probably because it was still a preview version. Almost all the functionality of the ESB has been replaced by BPEL functionality.
- **Unchanged data synchronisation between BAS and HEIN.** The proof of concept could not make use of a complete and functional GemTax system. The functionality in the form of PL/SQL procedures which are already available in HEIN for use with InterConnect can in theory be reused in an ESB alternative. However, without a complete HEIN system these PL/SQL procedures do not compile and because of an unstable ESB, InterConnect is used in the proof of concept.
- **Unchanged data import.** The output of the Key Data Stores is a rather complex set of files which can not easily be imported using the standard ESB File adapter. A custom import adapter could be written, for which was no time in the proof of concept. Therefore BAS is placed in the back office as a system concerned with maintaining a datawarehouse for use in other back office processes.
- **No security.** The proof of concept is not intended to go in production and is deployed in a secure environment. To reduce the complexity the Web Service Manager is not deployed.
- **Processing electronic forms.** Not much has changed concerning the processing of the electronic forms as described in Chapter 6.4.1. The only change is that the proof of concept only accepts XML files as output of the electronic form. This is because the case dossier is not available in the proof of concept and the case dossier is not a part of GemTax.
- **Feedback duty.** The law which mandates the feedback duty comes into effect in 2007. At the moment the proof of concept was developed no specific information on how to perform the feedback was available. Importing data in BAS is done by using third party tools, shielding the Key Data Store from GemTax. As soon as the departments responsible for maintaining the Key Data Stores come with a way to fulfil the feedback duty this can be implemented. Possible errors are registered in BAS and can easily use stock services like e-mail functionality from a BPEL process to fulfil the feedback duty.

7.1.2 Providing information

A government portal or a web site of a municipal authority can provide its customer with personalised tax information. These web sites provide in the authentication and authorisation of the user using DigiD. The web sites then invoke a web service for providing the personalised tax information in the Customer Service Point by providing the BSN of the user. This web service is a BPEL process which requests the information from the HEIN system.

7.1.3 Interacting with GemTax

GemTax is considered a blackbox or COTS product. This means that no software development will take place in the BAS and HEIN systems of GemTax. The SOA extension can therefore be considered as separate system which communicates with GemTax.

The SOA extension interacts with BAS and HEIN on the database and logic level. Oracle's SOA Suite provides Database Adapters for executing PL/SQL procedures or database queries. These database adapters are web services in their own right, and will be invoked from BPEL processes.

Using database adapters for interaction with GemTax automatically means that these adapters have to connect using a database user to the databases who has certain rights on certain database tables. This is one of two small modifications to the GemTax system that the SOA extension requires. The other modification is the addition of new ascertains in the specific database table of BAS to cope with reports of possibly incorrect data as reported by the customer.

7.2 Functional design and implementation details

Looking at the Customer Service Point three categories of web services can be discovered:

1. web services for getting personalised information from GemTax;
2. web services for changing the customer's payment method;
3. Web services for dealing with complaints.

7.2.1 Requesting personalised information

When a customer wants to request personalised information, the government first has to know who it is dealing with. This means using DigiD to authenticate the customer. GemTax will receive the BSN number and will gather the data based on this number. The process model is shown in Figure 7.2.

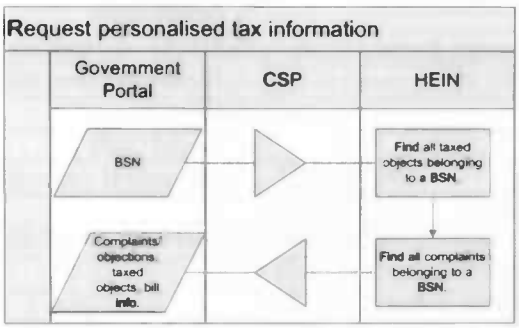

Figure 7.2: Process model of providing personalised information to the customer.

With the BSN number the following information is gathered from GemTax:

- **Name and address information.** With this a customer can see if the correct data about him is known by the tax systems.
- **Tax bill information.** Per taxed object the amount, the number of payment terms, payment deadline, payment method and the billing address is gathered.
- **Information about the taxed object.** The kind of taxes for the object, the objects address and the amount of money owed.
- **Information about lodged complaints.** For all the complaints lodged by the customer which are available in the system information plus the status is gathered.

The output is a large XML document containing all the gathered data. The data is gathered using a number of BPEL processes which all take the BSN number as input. The BPEL processes use Database Adapters to gather the information from the complex e-Business Suite data structure of HEIN.

Four BPEL processes are involved in this process. PIPINFO receives the BSN and redirects this number to three processes which run in parallel: GETCUSTOMERDATA, GETTAXBILL, GETBEZWAARSCHRIFTINFO for getting the name and address information, information about taxed objects and information about lodged complaints respectively. The output of these processes is merged and transformed in the desired output for PIPINFO by means of an XSLT transformation.

GetCustomerData

Getting information such as name and address from GemTax is done by a Database Adapter which queries the HZ_PARTIES table from the AR database schema. This table contains the information about the customers including the BSN. This process takes the BSN as input.

Gathered information from the table:

Column	Description
ATTRIBUTE2	Registered voter
ATTRIBUTE3	Bank number
ATTRIBUTE4	Giro number (Postbank)
PERSON_FIRST_NAME	First name
PERSON_MIDDLE_NAME	Lastname prefix ("tussenvoegsel")
PERSON_LAST_NAME	Lastname
COUNTRY	Country indicator (two letters)
ADDRESS1	Streetname
ADDRESS2	House number
ADDRESS3	House number suffix
CITY	City
POSTAL_CODE	Postal code
COUNTY	Municipal authority

Table 7.1: The gathered personal information

The column ATTRIBUTE1 contains the BSN, which is used in the database query to find the specific row needed.

GetTaxBill

Getting the information about the tax bills of the customer requires some more work to be done. These so called invoices are identified by the PARTY_ID from the AR.HZ_PARTIES table. This process takes the BSN as input. A helper BPEL process GETPARTY_ID is created which provides the PARTY_ID belonging to a BSN it gets as input.

With the PARTY_ID the following tax bill data is gathered from the table XXATX.XXADE_INVOICE_HEADERS, zero or more rows are returned as shown in table 7.2.

Besides this tax bill data the following data about the taxed object is returned and merged with the data above

Column	Description
TRX_NUMBER	Billing number/identifier
TRX_DATE	Billing date
BILL_TO_CUSTOMER_NAME	Full name of the billed customer
BILL_TO_ADDRESS1	Streetname of the billing address
BILL_TO_ADDRESS2	House number of the billing address
BILL_TO_ADDRESS3	House number suffix of the billing address
BILL_TO_POSTAL	Postal code of the billing address
BILL_TO_CITY	City of the billing address
PAYMENT_TERMS	Number of payment terms
DUE_DATE	Payment due date
BILL_TO_BANK_ACCOUNT_NUM	Bank account for billing
TOTAL_AMOUNT	Total amount of the bill
INCASSO	Payment method

Table 7.2: Tax bill data

Column	Description
LINE_ITEM_DESCRIPTION	Object name
LINE_EXTENDED_AMOUNT	Amount of money the object is taxed for
ITEM_NAME	The kind of tax

Table 7.3: Data of taxed objects

GetBezwaarschriftInfo

This process is responsible for getting information about lodged complaints by the customer and also gets the BSN as input. It gathers the following data from the XXATX.BEZWAARSCHRIFTEN of HEIN:

Column	Description
AANSLAG_ID	Identifier for the corresponding tax bill
STATUS_ID	State of the complaint
STATUSDATUM	Date of the latest state change
ONTVANKELIJK	Shows whether or not the complaint is granted
ONVANGSTDATUM	Date the complaint was received

Table 7.4: Information on lodged complaints

This data is appended with the number of appendices the specific complaint contained from the XXATX.BEZWAARSCHRIFTENDOCS table.

7.2.2 Changing the payment method

There are two ways a customer can pay taxes. The first is by receiving a tax bill and manually pay this bill. The other way is that the money owed is automatically withdrawn from the customer's bank account. If the customer wants to switch between these two it can be done by requesting it using an electronic form. The process model is depicted in Figure 7.3.

The proof of concept features two web services for this. One for changing to the automatic withdrawal, the other for changing to manual payment method (AUTOMATICINCASSO and ACCEPTGIRO respectively). Again these web services are BPEL processes which use Database Adapters for making the required changes in HEIN.

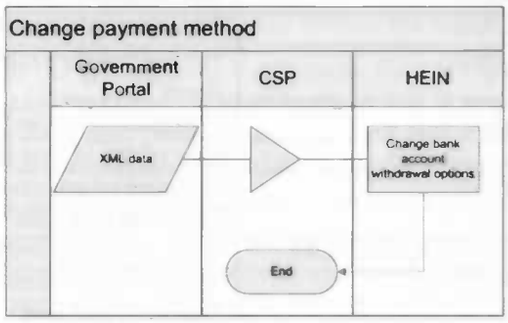

Figure 7.3: Process model of changing the payment method.

The payment methods are defined in AR.AR_RECEIPT_METHODS in HEIN. In the table AR.RA_CUST_RECEIPT_METHODS the payment method of the customer is defined by linking here to the AR.AR_RECEIPT_METHODS table. However, finding the specific customer in AR.RA_CUST_RECEIPT_METHODS needs some work. Remember that the BPEL processes get the BSN as input, and the identifier in the AR.RA_CUST_RECEIPT_METHODS is CUSTOMER_ID. The helper service GETCUSTOMERID gets the CUSTOMER_ID which belongs to a BSN. This process links the BSN with the PARTY_ID from AR.HZ_PARTIES which is a foreign key in AR.HZ_CUST_ACCOUNTS. In this table CUST_ACCOUNT_ID key can then be used to find the corresponding CUSTOMER_ID in AR.RA_RECEIPT_METHODS.

With the BSN now linked to a CUSTOMER_ID the BPEL processes AUTOMATICINCASSO and ACCEPTGIRO can change the payment method to and from automatic withdrawal by getting the BSN as input.

7.2.3 Lodging complaints

The previous tasks were, apart from the very complex database structure of the e-Business Suite HEIN, not very complex. Lodging complaints by electronic forms is a different story. Here BAS comes also in the picture. A complaint about the tax bill is caused by a tax bill based on faulty information. For example when the value of an object is incorrect, or the party in question has just moved so the tax bill arrives at the wrong customer. The process model can been seen in Figure 7.4 on page 59, the rectangle represents automatic actions, the trapezium manual (human) actions done by public servants.

Refering to Figure 7.4 on page 59 the following can be said:

- The XML document the electronic form gives as output contains all information needed for the complaint. This means information about the customer, the object in question, and what the reason is for the complaint.
- Based on the reason for the complaint the first decision is made whether or not the complaint can automatically be administered in HEIN and BAS. For example, the complaint that the property value of the object is incorrect can automatically be administered, but when the customer specifies its own "custom" reason this is not the case. The complaint can automatically be denied (rejected) if, for example, the complaint is out of date.

- If the decision is made to manually investigate the claim, the BPEL workflow initiates a so called Human task. This standard Oracle BPEL functionality assigns the task to a user who has manually to asses the claim using a web based interface.
- Looking at HEIN, it automatically administers the complaint and stops the tax collection in question. This is the only automated part of HEIN. The rest of the tasks in HEIN have to be done by the people of the back office and are supported by already available standard functionality of HEIN.
- When a complaint is caused by faulty data, this particular data from the operational data store –BAS that is– should be flagged as possibly incorrect. This is done by automatically moving the specific data from the Data Core of BAS to the Front Portal where it is flagged as possibly incorrect.
- Depending on the source of the data (a Key Data Store or not) the feedback duty has to be initiated. If the data did not originate from a Key Data Store standard BAS functionality is used by a user to asses and correct the data in question.
- After the data is updated the complaint can be processed further by HEIN.

Lodging complaints makes use of the complex workflow of the mid office. Not only does the workflow contain automated process steps, but also manual (human) process steps. The BPEL Process Manager of Oracle supports this out of the box. The human tasks are standard web services which are generated and deployed together with the BPEL process. It can even send SMS messages or e-mails in the same way. This functionality can be very useful with coordinating complex workflows between different back office processes, as seen here with BAS and HEIN.

Looking at Figure 7.4 we basically see three steps in BAS and HEIN:

1. Routing/intake.
2. Create/administer complaint and stop the specific tax collection.
3. Flag the corresponding data as possibly incorrect in BAS.

The routing BPEL process features a Human Workflow which enables a public servant to assess whether or not the complaint should be automatically processed or manually.

For the automatic creation and administration of the BPEL process a Database Adapter is build on top of an existing PL/SQL procedure in HEIN to administer and stop the specific tax collection. This procedure is called "ambtshalve_bezwaren" procedure and belongs to the XXATX_BEZWAAR package from the APPS database schema.

Depending on the lodged complaint certain data has to be marked as possibly incorrect in BAS. This means that an ascertain on the specific data item needs to be made. The proof of concept contains a BPEL process which can make an ascertain on the WOZ data, when the customer lodges a complaint if the valuation of its property is not correct. The specific data item is moved from the Data Core to the Front Portal and marked. In this way the other back office systems will not have access to this (possibly) faulty data until the investigation is finished. This is all done by Database Adapters which execute SQL queries on the data in BAS.

7.3 Summary

Implementing GemTax as an electronic government client system as introduced in Chapter 6 proved difficult. At the time of implementation, the brand new Oracle SOA Suite was still a

Developers Preview edition resulting in an unstable ESB. Fitting GemTax in the electronic government means SOA-enabling it and make it work together with the solutions the electronic government has in place (or will have). SOA-enabling on database level is not the best way to do that. The complex data structure of the e-Business Suite makes even simple tasks rather complex. Preparing GemTax by creating specific PL/SQL procedures in the logic layer will make the creation of BPEL processes and web services on top of GemTax easier and more maintainable.

It seems also that the rest of the government is not ready for a complete implementation of the electronic government. The feedback duty is an example of this, it was not possible to implement it, because the law on the Key Data Stores is not in effect and therefore no information was available on how to feedback possibly incorrect data to the data source was known.

However, despite the differences between the actual implementation and the architecture from Chapter 6, the proof of concept implementation of the SOA-extension of GemTax makes a customer-centric communal tax application from an otherwise rigid and closed application.

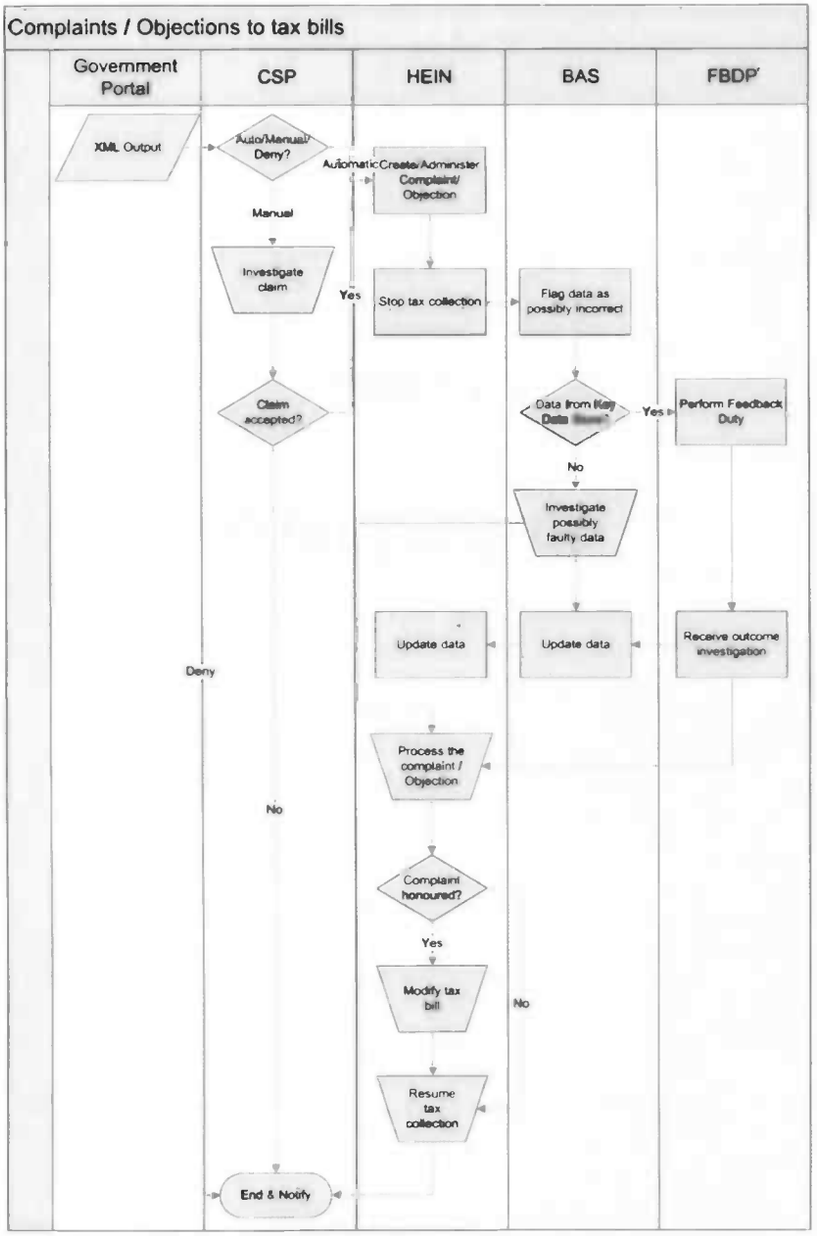

Figure 7.4: Process model lodging a complaint.

Part IV

Wrapping up

Chapter 8

Conclusion

This thesis is about the electronic government and its client systems. The electronic government is the result of the action plan "Andere Overheid" launched by the Dutch government in 2003. At the moment of writing it is the latest attempt to improve the service of the government towards their customers. It calls for a more efficient government, customer-centric, which acts as a single entity and supports "one stop shopping".

The plan "Andere Overheid" stood at the cradle of several initiatives to implement the electronic government. Most notably the plans to lower the administrative burden by centralising the data needed for government processes in the so called Key Data Stores and the Dutch electronic government reference architecture NORA.

The NORA is in essence a recommendation for government and public agencies which defines the recommended architecture of the electronic government. However, due to the subsidiary principle government agencies have the right to implement their own "electronic government" in their own way. That is why the NORA should be viewed as a recommendation defining a layer on top of a government agency to implement the customer-centric electronic government acting as a single entity.

This thesis centers around the following research question:

> In what way should the electronic government architecture be implemented such that client systems can use the provided data and functionality in the way Dutch laws and regulations prescribe?

To answer this question the following was investigated:

1. What are the underlying principles of the electronic government?
2. What is the electronic government?
3. What is a viable solution for implementing the electronic government?
4. How should client systems fit in this electronic government?

8.1 The electronic government

The electronic government is based on a set of fundamental principles from five stakeholders: the Dutch government, the European Union, the Dutch citizens and the Dutch business community. These stakeholders work within the boundaries defined by the fifth stakeholder: the Dutch

judiciary. The principles of these stakeholders form the basis of the Dutch electronic government reference architecture NORA. The idea is to implement the electronic government as one big service oriented architecture.

Implementing the electronic government three main aspects of the electronic government are identified which needs to be implemented:

1. A customer-centric approach;
2. one-stop-shopping;
3. act as a single entity.

8.1.1 Customer-centric approach

To implement the customer-centric electronic government, a government organisation is split in three levels: the front office, the mid office and the back office. The front office is concerned with the communication with the customer. It can do so via a multi-channel approach, such as mail, e-mail, telephone and internet.

The back office of a government organisation is the place where the actual products the customer requests are being processed. Because products can span across several back office departments the IT-landscape of the back office must be integrated and opened up in the SOA of the electronic government.

The mid office integrates the back office systems, and the back office with the front office. It does so by providing an enterprise service bus message broker and an operational datastore for providing aggregated data from the Key Data Stores and the back office systems themselves. In the mid office service orchestration takes place to direct the product realisation between the different back office processes.

Not only the back office system should be customer-centric, also the workers from the back offices should see their work as part of a (bigger) product realisation. Therefore, the mid office implementation implements also human workflow instead of only process orchestration of web services.

8.1.2 One-stop-shopping

One-stop-shopping means that the customer should be able to requests products, which spread over multiple back office departments or even government or public organisations, at one place as one unique request. Digital counters should be created to give to customer a single place it can do its business with the government. These counters aggregate available services which relate to eachother by searching the government-wide product portfolio.

8.1.3 Act as a single entity

The electronic government must provide centralised common solutions to act as a single entity towards the customer. This means that the customer must have a common electronic way to request products from the government by using electronic forms. The government should have a central government portal side which shields the customer from all the underlying different government and public agencies.

Every customer must have a digital identity which must be used throughout the whole electronic government. DigiD is used for this.

For other ways of communicating with the government than the internet, the front office shields the customer from all back office departments they used to do their business with directly. A government or public agency acts as a single entity because the front office which "a single face" towards the customer. The government as a whole can be seen as a single entity by providing a single front office for the complete government that integrates the front offices of all the organisations in the way the digital counters do that for internet communication channel.

8.2 Client systems and the electronic government

For a client system to fit into the IT-landscape of the electronic government, the client should know its position within the front-mid-back office solution of the specific domain it is deployed in. Depending on the position of the client system it has to integrate with the solutions already available or implement a common solution.

Client systems in the back office must be open enough to be integrated with the mid office solutions and in that way with the other back office systems. Front office systems should be compatible with the multi-channelled communication channels and mid office solutions.

Client systems need to be open and SOA-compliant. Functionality needed for providing a specific deliverable in product realisation needs to be accessible by web services so that it can be orchestrated by the process orchestrator and used by the case dossier.

Once the client system has been positioned in the front-mid-back office solution the three aspects of the electronic government come into view again:

1. **Is the system customer-centric?** What does the customer wants with the system? Does the system need interaction with the customer?
2. **Is the system part of a chain of services?** Does the system supply a deliverable in a product realisation process? So does it need integration with mid office solutions such as the case dossier to be part of the one-stop-shopping product realisation?
3. **Should the client system use common components?** Is the client system dependent on common components, such as the electronic forms, the product portfolio etcetera, of the electronic government? Should it hide itself behind the single common "face" of the government?

SOA-enabling existing applications remains a difficult task. The systems implement their own processes which not always can be exposed. The systems can also be difficult to create web services upon. It all depends on the application itself whether or not it can be absorbed in the electronic government.

In the end the subsidiary principle gives client systems almost a carte blanche on how they will fit in the electronic government.

8.3 The Proof of Concept

The Proof of Concept implementation is concerned with the GemTax product for billing and collecting communal taxes. It is for use by municipal authorities. It consists of an operational datastore, a message broker and the actual system for billing and collecting taxes.

Because GemTax is based on Oracle software the SOA-extension for GemTax is build with Oracle's SOA Suite 10^g 10.1.3.1 Developers Preview. The architecture for this extension splits the GemTax

system in a front-mid-back office solution. The operational datastore and Oracle's ESB plus BPEL Process Manager inhabit the mid office, whereas the actual system for billing and collecting taxes resides in the back office. The front office connects the Key Data Stores with the operational datastore, but also gives access to the electronic forms for customer interaction and the already present GemTax helpdesk application.

The customer-centric aspect of GemTax lies in the electronic forms interaction, the helpdesk application and the information services. A customer can request a change in payment method or lodge a complaint using electronic forms. GemTax SOA-extension provides an information web service which gathers tax bill information based on the BSN number of the customer. This information can be used on the Personal Internet Page of the government portal web site.

By placing the GemTax helpdesk application in the front office and the use of the common component as the electronic form helps the municipal authority act as a single entity.

The actual implementation differs from the proposed architecture because of the use of the pre-production version of the Oracle SOA Suite. Fact is, and remains, that because of the subsidiary principle the actual Proof of Concept implementation is not all of a sudden invalid, but not-optimal at most.

8.4 Future research

The Dutch government is not known for its fast response to market changes or adequate reforms to service their customers in an optimal way. Bureaucracy always seems to be associated with slow and inefficient service. The IT sector should keep a close eye on the electronic government development. The development of the electronic government will continue in the years to come. Maybe it is wise for the IT sector to "guide" the government to the desired implementation of the electronic government, because it would not be the first time a government initiative for electronic service fails [4]. The Centric software company is doing so with its Centric Melodies. Many more software companies should provide standard COTS products to speed up the implementation of the electronic government at all government agencies.

A great deal of research is still necessary on the Common Index, new common solutions and process integration. With the continuing integration and collaboration of the nations of the European Union the different governments will continue to seek integration of the systems. Here a great deal of research still needs to be done. Crime fighting, disaster prevention, anti-terrorism for example require collaboration and fast data sharing between government agencies from different nations.

Key Data Stores overview

The complete overview of the Key Data Stores and their relation to eachother. It is in Dutch and taken from the NORA.

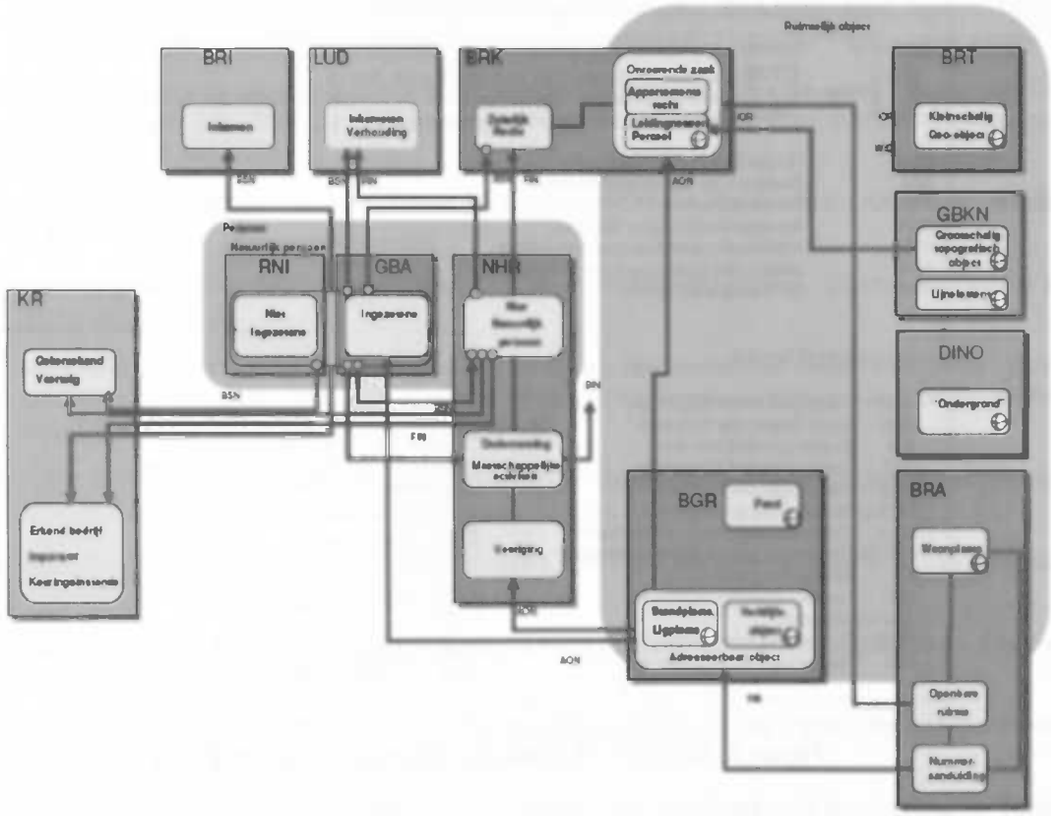

Figure 1: The complete overview of the Key Data Stores that be implemented by 2009 [7] (as of May 2006).

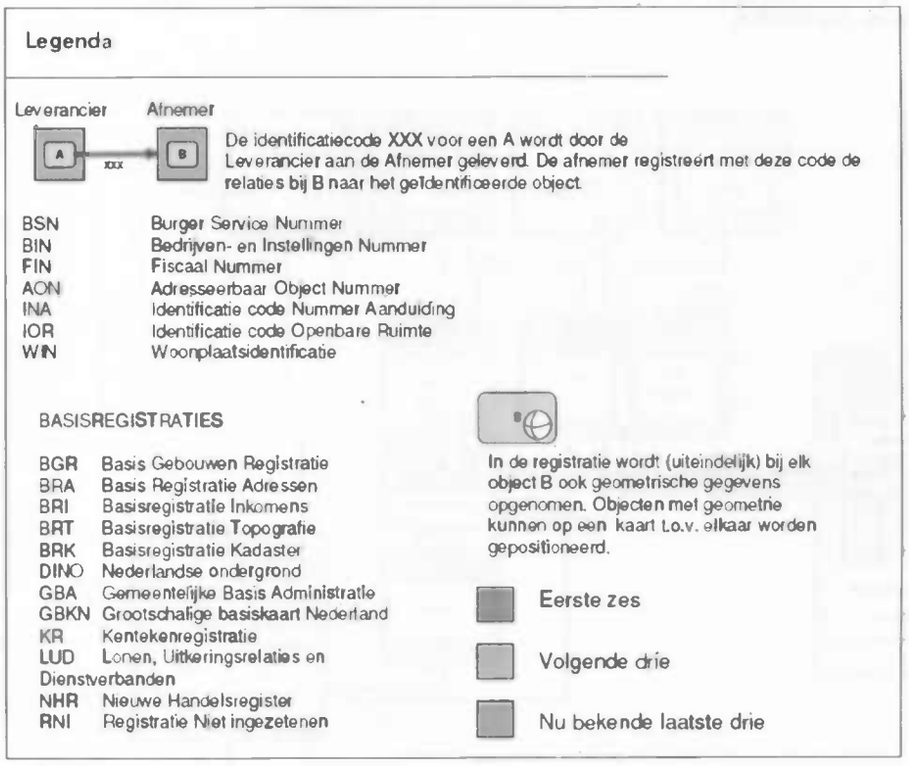

Figure 2: Explaining legend for Figure 1, taken from [7].

Bibliography

[1] P. Oaks A. Barros, M. Dumas. A Critical Overview of the Web Service Choreography Description Language (WS-CDL). *BPTrends*, (March), 2005.

[2] Ali Arsanjani. Service-oriented modeling and architectures. IBM, november 2004. `http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/`.

[3] Ila Bujar. Technische architectuur samenwerkende catalogi. Project Samenwerkende Catalogi, version 2.0, May 31 2006, `http://samenwerkendecatalogi.overheid.nl/home/documentatie/`.

[4] Anja Timmer Martijn van Dam. *Tussen het Kastje en de Muur*. Partij van de Arbeid, September 2006.

[5] D. Georgakopoulos M.P. papazoglou. Service-oriented computing. *Communications of the ACM*, 46(10), 2003.

[6] mr. J.A.M. Hendrix mr. J.G.E. Gieskes. WOZ-registratie binnen het stelsel van basisregistraties. Letter to drs. A. Pechtold, Minister Bestuurlijke Vernieuwing en Koninkrijksrelaties, May 11 2005, Letter identification: 05.2437 RK.

[7] ICTU Programma Architectuur Electronische Overheid. Nora: Nederlandse Overheids Referentie Architectuur. Version 1.0 September 27 2006 `http://www.e-overheid.nl`.

[8] Matt Poelmans. The Dutch e-Citizen Service Code ("BurgerServiceCode"). ICTU, February 2005 `http://www.burger.overheid.nl`.

[9] Paul Robertson. Integrating Legacy Systems with Modern Corporate Applications. *Communications of the ACM*, 40(5), 1997.

[10] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties. Actieprogramma "Andere Overheid". `http://www.andereoverheid.nl`.

[11] Robert-Jan van Es, Nick van Gerwen, Art Ligthart, Ron van Rooi, and Jeroen Graave. *Service Oriented Architecture. Een praktische leidraad voor invoering: Socrates*. Ordina Systems Integration & Development B.V. and Sdu Uitgevers B.V., 2005.

[12] Mr. M.J. van Pomeren. Het gebruik van persoonsgegevens in de gemeentepraktijk. *Bestuursrechtelijk tijdschrift: de Gemeentestem*, (June 10), 2006.

[13] Ministerie van VROM. Handreiking implementatie Basisregistratie Adressen en Gebouwen. Version 2.0 `http://bag.vrom.nl`.

[14] Bart Strum Wim van der Sanden. *Informatie-architectuur, de infrastructurele benadering*. Panfox, Ordina Public B.V., 2000.

[15] M. Roovers W.J. Keller, R. van Erkel. Marktverkenning mid office systemen. Programmabureau EGEM, 26 november 2004.

[16] U.O. Pijpker R. Schoemaker D. Schravendeel J.C.J. Takkenberg X. van der Linde, S.B. Luitjens. Upstream! Chronicle of the Streamlining Key Data Programme. 1st Edition, May 2003 `http://www.stroonlijningbasisgegevens.nl`.

[17] J.A. Zachman. Enterprise architecture: A framework. The Zachman institute for Framework Advancement. `http://www.zifa.com`.

[18] J.A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.

Index