Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2013

Pedro M. Sousa Bento

Plataforma E-Government para as Assembleias Nacionais dos PALOP

E-Government Platform for the Lusophone Africa's National Assemblies

Universidade de Aveiro Departamento de
Electrónica, Telecomunicações e Informática,
2013

Pedro M. Sousa Bento

Plataforma E-Government para as Assembleias Nacionais dos PALOP

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor Hélder Troca Zagalo, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e Co-orientação do Doutor Cláudio Jorge Vieira Teixeira, Equiparado a Investigador Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

agradecimentos

Aos Professores Hélder Zagalo, meu orientador, e Professor Cláudio Teixeira, meu co-orientador, pelo apoio e motivação para o desenvolvimento do trabalho. Aos amigos e amigas que me acompanharam na mais valiosa etapa da minha vida. Pela sua disponibilidade, confiança e fé, durante o processo de aprendizagem, valorização pessoal e interpessoal, que nos tornam verdadeiramente Humanos. Por fim, aos meus familiares, em especial aos meus pais, pelo seu carinho, compreensão e apoio constante que constituiu a base do meu sucesso.

Resumo

Este documento e trabalho desenvolvido, assentam na perceção que o Mundo está em constante mudança. A Internet, hoje em dia, desempenha um papel enorme na vida quotidiana e é a forma mais rápida de veícular informação que a Humanidade já presenciou. Regiões do Mundo como a Europa e América, têm grande facilidade em efetuar transições de informação entre os seus cidadãos e governos, através da Internet, (serviços de E-Goverment). Porém, esta realidade não é reflectida em todas os continentes e África é um dos que ainda apresenta muitas insuficiências. Para colmatar algumas necessidades, no que respeita a comunicação por serviços de E-Government, as Nações Unidas criaram o projecto "African I-Parliaments", que desenvolveu o Sistema Parlamentar Bungeni. Este sistema parlamentar, tem como finalidade automatizar o processo legislativo dos Países Africanos. Analisando o sistema Bungeni e desenvolvendo ferramentas, o trabalho descrito neste documento, tenta criar condições para a implementação deste sistema nas Assembleias Nacionais dos PALOP (Países Africanos de Língua Oficial Portuguesa), facilitando assim o acesso à legislação e seu processo de criação através da Internet, ajudando na criação de transparência entre governos e cidadãos.

Abstract

This document and the developed work lie on the perception that the World is always changing. Internet has taken a major role in one's daily routine and is the vehicle to spread the most quantity of information, that mankind as ever witnessed. Regions of the World such as Europe and the Americas, have great ease in exchanging information between the citizens and their governments, but this reality does not reflect in every continent, and Africa is one that still presents insufficiencies regarding this matter. To fulfil some of the needs of African countries, in which regards E-Government services communication, the United Nations developed the African I-Parliaments Project, which developed the Bungeni Parliamentary System. This system aims to help automatize the legislative process of African Parliaments. By analysing the system and developing tools, the work described in this document tries to get the Bungeni Parliamentary System to be implemented in the Lusophone Africa, therefore allowing ease of legislation access and creation, through the Internet, helping in the building of government to citizens transparency.

Contents

List of Figures

List of Tables

Acronyms

ADSL	Asymmetrical Digital Subscriber Line
API	Application Programming Interface
CGI	Common Gateway Interface
CMS	Content Management System
DOM	Document Object Model
EGDI	E-Government Development Index
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICT	Information and Communications Technology
IDE	Integrated Development Environment
INI	Initialization file
IT	Information Technology
MVCC	Multi-Version Concurrency Control
ORM	Object Relational Mapper
PAS	Pluggable Authentication Service
PDF	Portable Document Format
RELAX NG	REgular LAnguage for XML Next Generation
REST	Representational State Transfer
RSS	Real Simple Syndication
SQL	Structured Query Language

SSH	Secure Shell
UI	User Interface
UNDESA	United Nations Department of Economics and Social Affairs
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language
XSLT	eXtensible Stylesheet Language for Transformation
ZCML	Zope Configuration Markup Language
ZODB	Zope Object Data Base
ZOPE	Z Object Publishing Environment

Chapter 1

Introduction

1.1 Introduction

This document and consequent developed work lies on the perception that the World is always changing. Internet as taken a major role in one's daily routine and is the vehicle to spread the most quantity of information, at any given moment, that mankind has ever witnessed.

Internet changes, but people change with it. Different trends appear everyday, always trying to appeal to the next consumer. Governments need to appeal and relate with the citizens as stores to consumers. Therefore, Internet is the perfect ground to achieve quick interaction between government and citizens.

Regions of the World such as Europe and the Americas, have great ease in exchanging information between the citizens and their governments, but this reality does reflect in every continent. Some countries, due to aspects such as war, poverty or any other aspect that conditionate a person's lifestyle, don't have that ease of communication.

To fulfil some of the needs of African countries, in which regards E-Government, the United Nations developed the African I-Parliaments Project. This project, has developed a parliamentary system, Bungeni (literally "inside the Parliament" in Kiswahili language). Bungeni Parliamentary System, aims to help automatize the legislative process of African Parliaments.

The countries belonging to Lusophone-Africa, or countries that have Portuguese as official language, became stakeholders on the implementation of the system, which can suit Parliamentary and Congressional types of legislation.

Due to the customizations allowed by the system, the legislative reality of each of these countries need to be created inside Bungeni. This fact can lead to hours spent, around configuration files, which can create entropy to the general use of the system. By analysing the system and developing tools, the work described on this document tries to get the Bungeni Parliamentary System to be implemented in the stakeholder countries, and therefore allow ease of legislation access and creation, through the Internet, accounting for government citizen transparency.

1.2 Motivation

Motivation comes from trying to reach out and preferably ensure that every citizen can have access to its Government, without reservations. It is believed that this document can help to justify the use of the Bungeni Parliamentary System in National Assemblies, but also use the centralized Portal as one engine to accelerate Citizen to Government interaction.

The centralized portal can provide access to the legislative information in real time, therefore allowing the citizens a vision of transparency into the Parliamentary Activities. This vision is important to the citizens, but Parliamentary members and staff need to have another view and access of information. A deep analysis of the Bungeni System ensures that the system fulfils the needs of these two types of users.

Most of the Parliamentary activities, executed by Parliament members or staff, are reflected to the outside World in the form of digital information. Often Parliamentary activities, represent slots in time, in which documents transit between states. Parliaments use the concept of workflow, as a set of tasks or states that a document must undergo until its completion. On the system, workflows are defined in a much time consuming form. Providing tools to the system that help productivity increase, in the creation of workflows, can be ultimately reflected on the interaction between all citizens and their Government.

1.3 Context

The National Assemblies need to empower the general use and modernization of modern Information and Communication Systems to, in a radical manner, make the quality of services evolve, as well as providing access to Parliamentary activities and documents to allow transparency, bringing the strengthening of African Democracies. It is in this context that the Bungeni project emerged, based on an open source concept, with the goal to improve the managing of information in the African National Assemblies to provide the citizens one easy access to Parliamentary Activities.

1.4 Objectives

The Bungeni project developed until now a set of tools that can be used in Parliamentary Environments, nevertheless, its fully usage has a need of several adaptations to face the needs and realities of each country. These adaptations are namely the creation and adaptation of parliamentary figures as well as all the workflows that allow the managing of documents by the Parliament.

This work has as first objective to contribute to the adaptation and creation of means to help in the Bungeni system, having a special attention regarding the National Assemblies of the African countries that have Portuguese as official language. These adaptations, can be provided by just translating the systems language, or adapting real legislative content behaviours, that reflect a countries reality, such as modeling a workflow.

The second objective lies in the creation of tools to help the above mentioned process, and allow a more efficient deployment on countries with differentiated Parliamentary scenarios. The tools developed, have as objective to increase the productivity of workflows, using a web-based workflow creator. This workflow creator aims to substitute the actual way of creating this type of content, by providing a user friendly point and click interface.

1.5 Document Structure

This document is divided into 5 chapters. In the beginning of each chapter there is a general introduction to the theme approached, trying to supply a general overview of what can be read in the next pages, or providing background information needed to understand the chapter itself.

The first chapter is this one and gives a general approach of what the work is and what is the structure information in this thesis.

The second chapter is dedicated to e-Government, providing a brief introduction to the theme and partial history. In this chapter open source solutions of e-Government platforms are presented, as well as the e-Government situation around the globe, stating some of the milestones achieved since 2001.

The third chapter is dedicated to the Bungeni Parliamentary System, as an alternative to the National Assemblies of the African Countries with Portuguese as Offical Language. Information is provided, from general features to architectural details without ignoring the workflow system and its nuances.

The fourth chapter is dedicated to a tool developed for the Bungeni Parliamentary System. This tool is presented giving a real World scenario, giving it purpose. Detailed information is provided too, about its architecture and features.

The fifth chapter closes the document, with analysis of the developed work and gathered information, providing closure and a sense of purpose.

Chapter 2

Background

2.1 Introduction

In this chapter a general overview of E-Government since its "debut" and establishment is presented. This part presents some background in order to fulfil one's need to understand the origin and progress of the work developed along the document.

2.2 Introduction on E-Government

"Given that citizens throughout the world have come to expect twenty-four hours a day, seven days a week availability in their commercial interactions, it is only natural that they would expect the same from their government." [1] In a simple sentence Evans defined the way a government should behave when it comes to Internet interaction with the citizens in 2006. Although 2006 is a long run from now (in therms of advancement in technology), this year marks history as the first year since Facebook, currently the biggest social network, became worldwide accessible. In 2006, Facebook was firstly used as a campaign tool in the midterm elections for the United States of America, stating forever that government initiative and the Internet could benefit from this kind of relationship. This stunt demonstrated that Internet could vehicle Government ideals and information, much faster than any other [2].

The contribution of "Social Media" alongside great technology improvements and low cost Asymmetrical Digital Subscriber Line (ADSL) opposed to slower and expensive Dial-Up Connection, changed the web itself. In the nineties Internet was a niche place that average users were only starting to discover [3]. In a simple form, its contents were static pages that held some information that constant update by programmers. From the year 2000 forward Internet content started to change and a lot of more information started to be easily found thanks to the mass popularization of search engines, such as Google.

In 2001 a paper was introduced describing the prevision of the impact that E-Government would have across the world. Citing the research that was done, "Described simply, E-

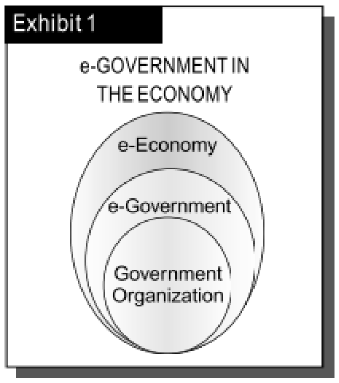


Figure 2.1: E-Government in the Economy[4]

Government is the application of the tools and techniques of e-Commerce to the work of government. These tools and techniques are intended to serve both the government and its citizens". Figure 2.1 shows the idea that the governmental organization can use the concepts of E-Government to extend its reach into the greater e-Economy. Eventually, E-Government will become "simply the way things are done" and will no longer be treated as a brand new concept, as happened with spreadsheets and fax machines [4].

The paper has also a few reasons why the E-Government should behave like e-Economy, and despite being written some years ago the given reasons are actual. One of the prominent reasons is the increasing of Internet users (as stated before), the total number of Internet users reached approximately 100 million by the end of 1998 and are projected to grow, Table 2.1.

Year	2005	2010	2013
World Population (thousands of million)	6.5	6.9	7.1
Not Using the Internet	84%	70%	61%
Using the Internet	16%	30%	39%
Users in the developing World	8%	21%	31%
Users in the developed World	51%	67%	77%

Table 2.1: World Population Internet Users [5]

Web buyers, according to the source, were expected to represent 40 percent of all users in 2002. According to new data 85 per cent of the Internet users have made at least a buy On-line, thus justifying the increase expected on the study [6]. Following the theory provided on the paper, if e-Commerce thrives, so should E-Government.

How should a government present itself on a global web? The idea emerging from Howard's report was that the information should be provided in the form of Web Portal, but then again, the paper was written in an era where Portals were the great information providers on the Internet. Every Web Site or company that wanted to give information about their services a decade ago needed to have their own Portals. Portals were so popular back then that even

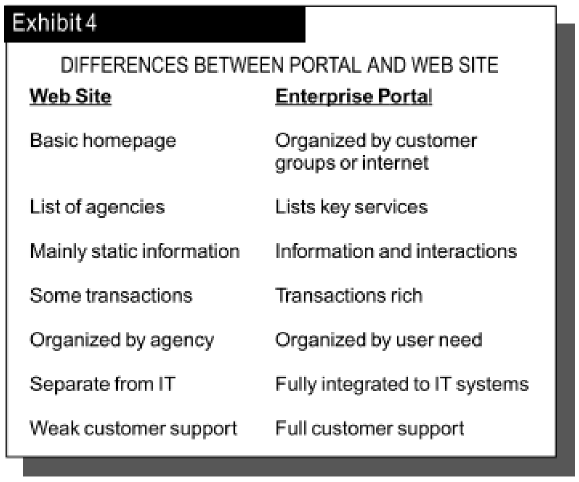


Exhibit 4

DIFFERENCES BETWEEN PORTAL AND WEB SITE

Web Site	Enterprise Portal
Basic homepage	Organized by customer groups or internet
List of agencies	Lists key services
Mainly static information	Information and interactions
Some transactions	Transactions rich
Organized by agency	Organized by user need
Separate from IT	Fully integrated to IT systems
Weak customer support	Full customer support

Figure 2.2: Services: Portals versus Web Sites[4]

some search engines evolved to them and eventually became too complex for normal users. The services provided in one simple web site are different from the experience provided on an enterprise Portal, just like suggested on Figure 2.2.

Despite the ageing of the document, the general idea did not fall too far in time and that Portals are still used in E-Government today. The main focus is that the access to the services provided by the government on its web extension should be all centred in one central site, instead of multiple sites, to maintain ease on access of information to the users that seek it.

According to the document, in 2001 Canada, Singapore and the United States were the countries with the most mature and innovative E-Government systems On-line (Figure 2.3). It is important to note that even though all the governments were making progress, at the time, even the most highly rated were showing less than 50 percent “maturity” ratings [4]. The last statement, provides the idea that despite a general positive progress line on all the systems surveyed, even the most mature ones were far away from a satisfiable response in order to represent the real Government on the Internet.

To conclude, although some technological setbacks, E-Government was quickly becoming a reality. It was expected that Citizens and businesses adopted it quickly enough so that it would eventually become simply one of the ways “things get done” rather than a novel concept, at the time. This would be achieved by using the previously referred Portals, to attract and help Citizens make governmental tasks with the services provided on such sites. In order for this situation to become a reality, two things needed to happen: the Portals should not be only listings of the services of one’s government but a short cut to help the citizens achieve goals; the services of one’s government should be all concentrated in a central site instead of agencies or services niches, separating the parts from the whole.

In the beginning of the century this was what was expected of E-Government systems.

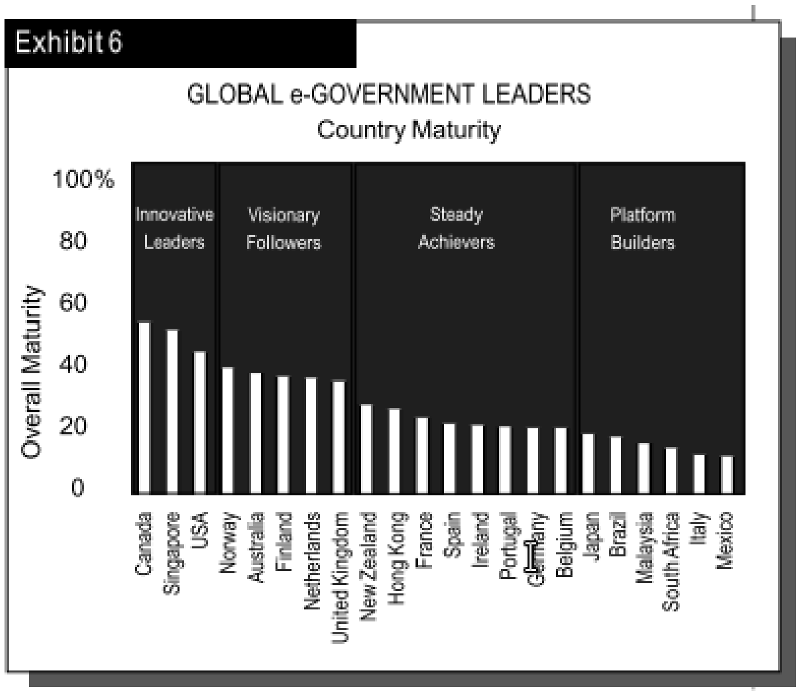


Figure 2.3: Global E-Government Leaders in 2004[4]

2.3 Types of E-Government

Though E-Government is a broad concept, related to everything that a government of one's country makes available online, it can be broke up in four major categories or models depending on the actors involved on the interaction:

- Government-to-Citizen;
- Government-to-Business;
- Government-to-Government;
- Government-to-Employees.

Government-to-Citizen

The government provides a wide variety of services to fulfil the needs of a citizen. Services like transparency to the parliament or renew a driver's license for example, ease the burden of a citizen. Not only does the citizen lose the need to have to be in loco, to perform these operations, but it also eases the bureaucratic burden.

Government-to-Business

Online non-commercial interaction between local and central government and the commercial business sector. It is useful to provide and gather information from businesses, such as taxes payment and financial benefits.

Government-to-Government

Intra governmental interaction, between government agencies or organisations. Can be seen as the cooperation from local governments or Municipalities to the central Government, or in some cases between Parliaments and Federal Government. Can be used to integrate different E-Government systems as one whole system, that on the outside World constitute a Government to Citizen or Government to Business service.

Government-to-Employees

E-Government tools supplied Online by the government for the governmental employees to be productive. The interaction is done directly between communication tools and government units. Services such as E-law (discussed further ahead) are a good example of this category of E-Government.

The number of basic E-Government services that should be implemented and are readiness measured since a decade ago is twenty [7]. They are as follow:

- income taxes;
- job searching services;
- social security benefits;
- personal documents;
- car registration;
- application for building permission;
- declaration to police;
- public libraries;
- certificates;
- enrolment in higher education;
- announcement of moving;
- health related services;
- social contributions for employees;
- corporate tax;
- VAT (value added tax);
- registration of a new company;
- submission of data to statistical offices;
- customs declaration;
- environment related permits;
- public procurement.

2.3.1 E-Government versus E-Governance

E-Government

"E-Government refers to the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government. These technologies can serve a variety of different ends: better delivery of government services to citizens, improved interactions with business and industry, citizen empowerment through access to information, or more efficient government management. The resulting benefits can be less corruption, increased transparency, greater convenience, revenue growth, and/or cost reductions." [8]

E-Governance

"Conceptually, governance can be defined as the rule of the rulers, typically within a given set of rules. One might conclude that governance is the process - by which authority is conferred on rulers, by which they make the rules, and by which those rules are enforced and modified." [9]

E-Governance can be defined as the use of ICTs as a tool to achieve better governance. This happens by introducing the citizens into e-government platforms, and make available mechanisms to integrate the citizens or businesses in governance participation. A good example is the submission of a public petition that can result in a law, by which any citizen (including the government, or the rulers) must abide.

E-Governance Applied in Law Making

A good example for implementation of E-Governance is the one given by the Spanish Legislature that started in 2008 [10].

A new law for science and Technology was developed taking into account public opinion using a web blog system. After the draft done by the committees entitled to this task, the texts were made public in a web site (composed only by open source technologies), that served as a central point for people to interact with the making of this law. A RSS feed was made available for interested users to follow the changes and debates. More than 600 contributions were received and at least 20% were taken into account for the final documents.

After this, workshops and round tables were created to further discussion and were disseminated Online. The average audience was composed by 500 physical attendees and more than 1500 users were online at some moment.

In the next phase, article creation, the texts were made available online once again, and a commenting system was used to address the right to exert opinion, about some part of a law.

Finally the law was approved and made available online, making this a perfect example of citizen cooperation on an E-Governance environment.

2.4 Methods to avaliate E-Government Platforms

In 2007, European Commission Directorate General for Information Society and Media, released a benchmarking survey that analysed the supply of online public services dictating rules and methods to measure accurately the availability of such services [7].

"The E-Government policy environment has evolved from "bringing public services online" to a concept of effective and user centric service delivery in an inclusive and competitive European society" [7].

Five priorities needed to be addressed so that these types of services were recognized and demonstrated the capability to deliver the **i2010!** E-Government Action Plan in a correct way. The priorities were:

1. No citizen left behind;
2. Making efficiency and effectiveness a reality;
3. Implementing high-impact key services for citizens and businesses;
4. Putting key enablers in place;
5. Strengthening participation and democratic decision-making.

The first priority regards the advancing inclusion through E-Government so that all citizens benefit from trusted, innovative services and easy access for all, leaving no one uninformed.

The second priority concerns the efficiency and effectiveness of a system, contributing to high user satisfaction, transparency and accountability. If a system is not effective on what it is supposed to accomplish, it has no reason to exist.

The third priority regards the implementation of services that are really needed by the citizens. If a system can supply 100% online availability but does not support operations needed by the citizens, the citizens simply will not reach out and start using it.

The fourth priority concerns the use of a secure system to protect and authenticate the citizens that demand operations of an E-Government system. On a system of this type there are collections of sensitive data being exchanged, so the public service (E-Government system) should be authenticated and provide confidentiality.

Lastly the system should provide a way for citizens to interact with the government directly. The participation can be done for example in a public debate or forum or also in a democratic decision-making, by voting proposals or submitting petitions.

The two core measurements of sophistication and fully-online availability of online services are the major metrics to measure a system's readiness.

This measurement is based upon a method that take into account new technological possibilities and insights.

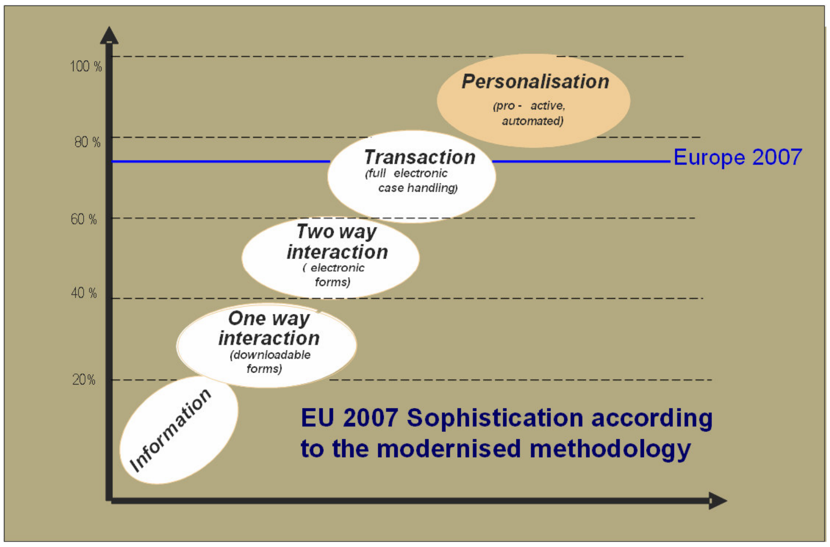


Figure 2.4: Sophistication Framework Levels [7]

2.4.1 Method to Create The Core Indicators

The scoring framework

In order to measure the indicator "availability of public services online", an e-service sophistication model was developed. This model illustrates the different degrees of sophistication of online public services going from basic information provision over one-way and two way interaction to full electronic case handling [7].

This indicator is based on the separating of the systems into two states:

- "no full online availability" : contains stages 0 to 3 of the sophistication framework.
- "full online availability" : status granted to all services that reach a stage strictly above the 3rd stage of the sophistication framework.

The sophistication framework levels can be observed on Figure 2.4.

This method has been applied consistently to assess the progress of e-Europe, regarding E-Government services.

Before 2007, the idea of pro-active service delivery, i.e. the government pro-actively performing actions to enhance the service delivery quality and the user friendliness was not measured.

The idea of automatic service delivery, i.e. the government automatically providing specific services, was one to be yet measured as well. Both the pro-active service delivery and automatic service delivery, constitute the personalization level, of the sophistication framework.

Europe achieved on 2007 an average overall sophistication maturity level that was between "two- way interaction" and "fully transactional" (Figure 2.4), or more precisely 76 per cent.

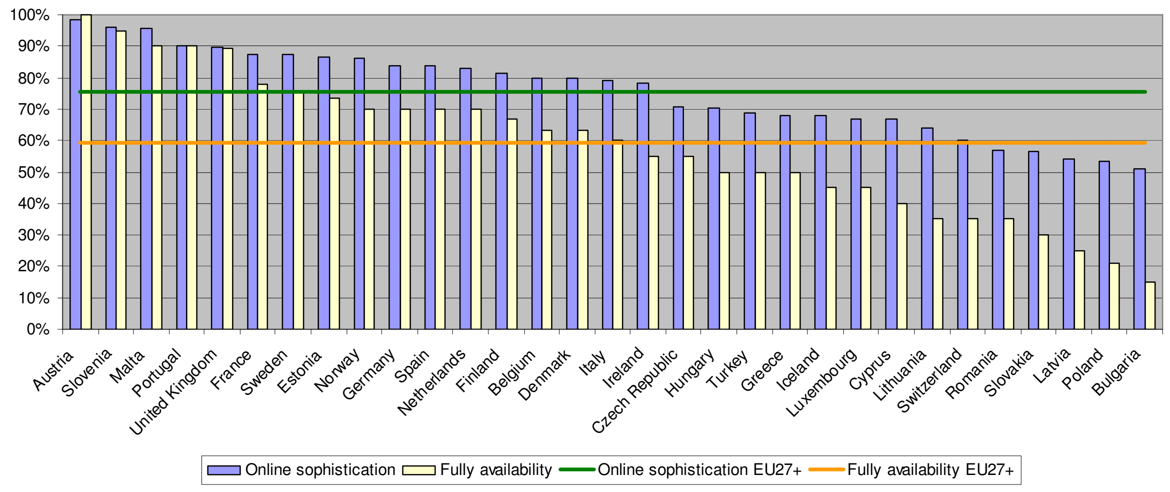


Figure 2.5: E-Government Services Readiness (Sophistication and Online Availability)[7]

By comparison to the year before there is a progression on sophistication, by almost a level, because the average had just reached a level of two way interaction (electronic forms) [7].

The status of sophistication and online availability in 2007 show that Austria had the biggest scoring. This can be seen on Figure 2.5.

Still, recent surveys use methods that are similar to this, taking in to account that technology is evolving. United Nations surveys on E-Government for example use these methods as basis and classify the development of E-Government services using a E-Government Developing Index (EGDI), that ranges from 0 to 1 being 1 the maximal score. Such score is based in several factors, but take in account that people live in a World where financial status plays a big role. One of the factors used to calculate this EGDI is the financial capability of a country related to the rest of the World. This EGDI is also dynamic and can ascend or descend in a country being evaluated from one year to another. This fact has to do with the method taking in account that technology evolves and what is a novelty in a year, may be a standard of evaluation in the next [11] [12] [13].

2.5 The Austrian Case Study - E-Law Project

According to [7], Austria was ahead of the European development regarding E-Government. It had both superior online availability and sophistication measurements on its services when compared to the rest of the Europe.

The Austrian E-Government Act [14], defined standards and security in communication between citizens and public services in 2004. Two years later the e-Law project surfaced and sent Austria ahead of Europe in development of E-Government services.

"The E-LAW project aims at a reform of legal text production, creating one continuous electronic production channel with a uniform layout prepared on the same electronic text basis from draft to publication (promulgation) on the Internet." [15] In a simple way, e-Law was a project created to manage and help the process of law making, since the draft to the publication, making it available for citizens along the progress to accompany. The specificity of the workflow produced by a legal document is due to its type and/or country in which it is created. For example, in Portugal the documents are created in the Assembleia da Républica, which is different from the Austrian workflows were the bills are created on the Nationalrat (local Parliament) and also Bundesrat (Federal House). The workflow system included in e-law includes government bills, committee reports, legal enactments of the Nationalrat and decisions of the Bundesrat.

The idea of e-Law was to make a window like system so that citizens could access the legal activity on the Internet any time and virtually anywhere. Electronic texts of legal enactments were to be delivered the Federal Chancellery, ready for publication and the authentic electronic publication on the Internet was available for everybody free of charge. The public and the Public Ministries could obtain the parliamentary business including the explanatory remarks within 24 to 72 hours on the Internet. The savings potential of the idea resembled one million Euros at the time, which was about sixty tons of paper. Unfortunatelly, there is a great opposition in using only digital copies of legal texts to work, and a major part is kept in paper, reducing the savings potential. Austria was trying to be the pioneer and innovate on this field of no paper policy.

The e-law project aimed at creating one continuous electronic production channel from the invitation to comment on draft legislation to promulgation (on the Internet). As a result, it is only required to enter amendments to the text during the legislative stages, nevertheless those amendments should be only done by people that can legally edit the documents.

The first intend result of the project, was to replace texts of laws on paper to digital copies, consolidating the idea that printed government bills, committee reports and other parliamentary matter would cease to exist. The technology would be able to assist in making it possible to draw up texts which could be queried electronically. The Online queries would refer to the original stages in which a document can attend alterations. Everything would be done in a fully transparent process.

"For the purpose of cost-cutting, the texts of legislation were to be given a uniform layout and were to be prepared on the same electronic text basis from draft to publication in the Federal Law Gazette on the Internet." [15] This means that different legal document types would have their own formatting. For example all the bills look alike, in the form if the text content is disregarded, but would be different from a committee report.

As a final result, The State Printing Office (Wiener Zeitung) would be no longer necessary. Following up the creation of government bills the Federal Chancellery would send them to

Parliament, which would return the consolidated electronic version of the legal enactment adopted by the Nationalrat once parliamentary procedures had been completed.

To translate E-LAW into reality, two projects were launched by the Administration of Parliament:

- the "Implementing E-Law" project to ensure one continuous electronic channel for the legislative procedure in the Nationalrat and the Bundesrat, as well as
- the "Roll-out Plan for laptops to be used by Members of Parliament"

Austria started to not only develop the system itself, but also provided tools in the form of laptops for the Members of Parliament to use and get used to the new electronic legal text workflow.

The re-design of the legislative procedure for the ministries was formally adopted by resolution of the Austrian Federal Government of 6 June 2001. At the same time that the implementation of the e-Law project was being advocated, the Presidents of the Nationalrat called for better IT equipment for the Members of Parliament.

The objectives to be met in the reform of the legislative process were defined as follows:

1. building up on existing databases
2. ensuring that the high quality requirements for parliamentary business will be fulfilled
3. taking into account the separation of powers between government and parliament
4. considering the principle of true costs (no passing on of costs or tasks from the government to parliament)
5. minimization of the total costs of parliamentary business
6. considering the special working conditions of parliament.

The first objective dictates that the system should use already existent databases in order to simplify the process, by not creating unnecessary redundancy.

The second objective can be seen as one of major importance in the system, because it concerns the real process of legal documents creating and their fully respective workflows being mapped on a digital world. This process has the obligation to be thorough by penance of turning the system obsolete on the failure of its successful completion.

The third objective is also a major concern in the system, and dictates the separation of powers, not only between government and parliament but also from them to the general citizens. Because it was a system to be implemented on an open world, the e-law project needed to secure the access in each stage of the creative legislative process. Only the agents that have the authority to intervene on the process at a given moment can do so.

The fourth and fifth objectives concern legal cost fees, and finally the sixth reflects the implementation of the system in the parliament as well as the special tools provided to the Members of Parliament to use it.

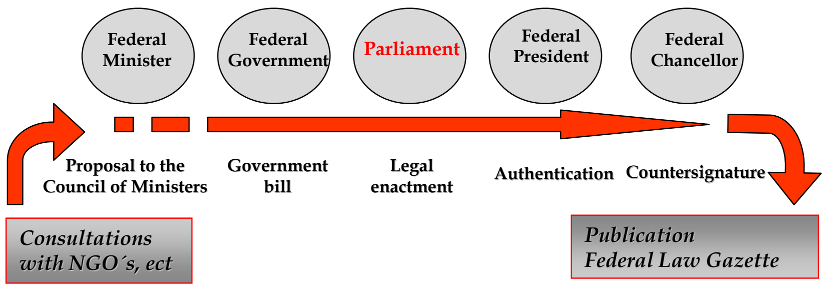


Figure 2.6: Simple Bill Workflow Example - Before E-Law[15]

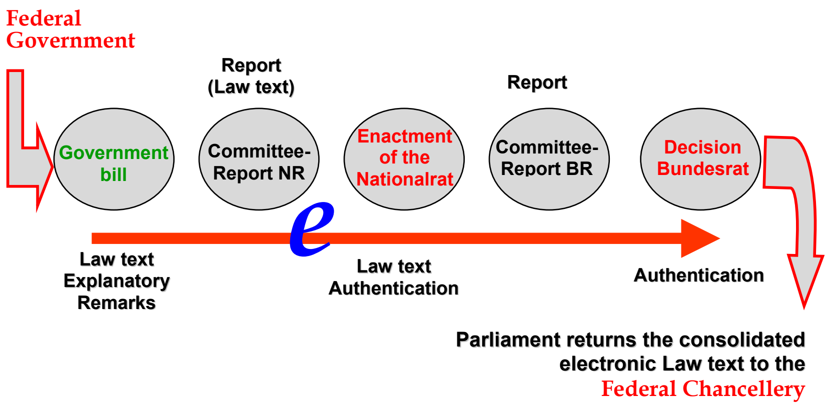


Figure 2.7: Simple Bill Workflow Example - After E-Law[15]

2.6 Separation of Duties in the Austrian e-law Process

The process and results of law making have an enormous impact on society, therefore they need to be secured against external and internal alteration, be it inadvertent or malicious. This is even more important in the electronic world, where there is already a human discomfort by working on a virtual environment. In fact, strong evidence suggests that internal alteration is far more critical than often perceived.

Separation of duties in the e-law project attends to the need to reflect organisational security and control properties that exist in the real world. E-law served as an example to the rest of the World regarding this matter.

The overall aim of this system is to replace printed law texts by digitally signed electronic documents as stated before. A brief analysis of the scenario of law making process, is provided in order, to comprehend the "Separation of Duties" imposed on the system.

A typical scenario detailing the use of e-law and its interaction with the Austrian Legal Information System (**RIS!**) is that of a change to existing law. Figure 2.8 can provide information relative to the workflow of this process.

To better understand the steps undergoing in this process they will be provided in the

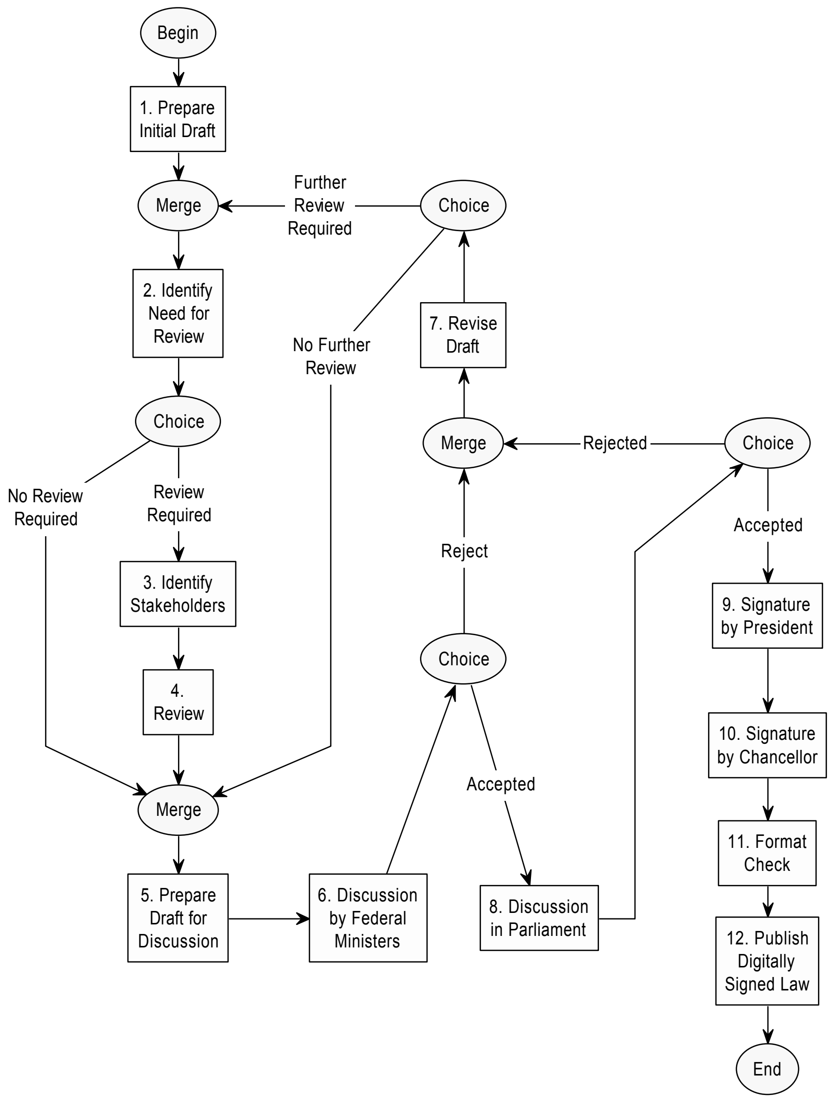


Figure 2.8: Detailed Bill Workflow For the Austrian Parliament[16]

form of a list, taking in account that a step can't be started without the previous one has been completed:

1. A law clerk working prepares a draft of the proposed change.
2. It is then decided whether the draft should be reviewed by external stakeholders.
3. The stakeholders are identified and invited to review and comment on the draft bill.
4. The stakeholders then review the draft.
5. The draft is prepared for further discussion and put on the agenda of the weekly meeting of the Austrian federal ministers.
6. If at the discussion the draft is rejected, it has to be proposed again by the initiating ministry.
7. In this case the draft is revised by a law clerk and it is decided whether further review of the revised text is required. If the federal ministers agree to accept the draft without change, it becomes a government bill.
8. The government bill is discussed by national council and then by the federal council.

   Both chambers may either agree or disagree, and possibly change the government bill. In case of an agreement, the text is passed back to the e-law system and BKA to be published in the federal law gazette. In case of a rejection or possible veto, the draft is again revised by the originating ministry, returning to step 7.
9. Prior to final publication, the president must sign and approve that the change to the law has been performed according to the constitution.
10. The president's approval must also be countersigned by the chancellor (step 10).
11. A final check of the new or changed law is performed by the constitutional service.
12. The changed law is then published in the RIS after it has been digitally signed to provide authenticity.

Security requeriments often cross paths with the separation of duties. On the case of the e-law project, the cross is made by having mechanisms that despite respecting the separation of duties, also enforce the non existence of conflict of interest done by changes on the system.

Such mechanisms prevent, that for example, a clerk initializing a legal draft should not be the principal who decides whether reviews by stakeholders are required. This prevents that a not well intentioned clerk could pass the document to the next stage in the system without consulting the stakeholders. Other example is that a clerk should not be allowed to modify a document and upload it onto the RIS at the same time. If this situation could happen, then a clerk could entry modifications and send it to the RIS without legal approval. From the examples given previously is easy to perceive that a clerk can't go through the all workflow of law making by himself.

Although the origins of separation principles can be clearly identified in the development of organisational theory, it is not always reflected on a virtual system that tries to replicate one existing organization of the real World. Separation of properties and roles that users carry on the real World should be applicable in the context of the judicial domain.

In the context of role-based access control systems, separation of duties are enforced based on the notion of mutually exclusive roles. Such roles are pairs of organisational roles which are in some way incompatible [16].

The separation of duties is proposed to solve some of the security problems stated before, that could incur in conflicts of interest.

Separation of duties can be done in a Static or Dynamic way if it is user centred. In the Static Separation of Duties a user can not be a member of any to exclusive roles [16]. The Dynamic Separation of Duties defends that a user may be a member of any two exclusive roles but must not activate them both at the same time.

Other type of separation of duties is the Object-based that has the focus on the object at work rather than the user working on it. It defines that a user may be a member of any two exclusive roles and may also activate them at the same time, but he must not act upon the same object through both.

Yet another type, focus the separation of Duties on the operation (Operational Separation of Duties) which states that a user may be a member of some exclusive roles as long as the set of authorizations acquired over these roles does not permit them to execute every step of a workflow, in law making for example.

Finally the History-based Separation of Duties says that a user may be a member of some exclusive roles and the complete set of authorizations acquired over these roles may cover an entire workflow, but a principal must not be able to perform all the workflow steps involving the same object(s). This is the category that on the end prevents a workflow from being done by just one user but allows for maximum flexibility.

2.7 Evolution of E-Government from 2008 to 2012

In order to perceive how E-Government evolved around the World, three surveys from the United Nations regarding this theme, served as basis to the overall analysis. The surveys are respectively from 2008, 2010 and 2012, made in increments of two years, in the way that they can provide equal amounts of time to the development of the E-Government services and their readiness. The rating or E-Government Developing Index as a range of 0 to 1, being 1 the maximal score, as described in 2.4.

In 2008, Sweden (0.9157) was the leader surpassing the United States, the leader of the previous survey. Three Scandinavian countries took the top three spots, with Denmark (0.9134) and Norway (0.8921) in second and third place respectively. The United States (0.8644)

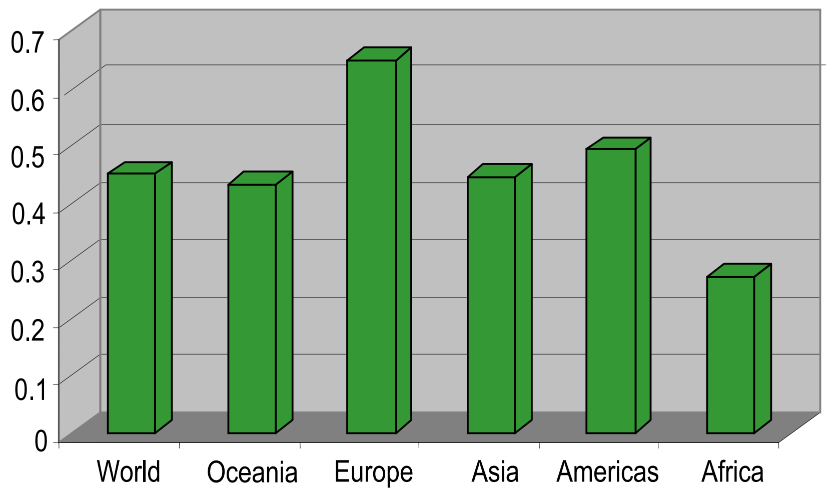


Figure 2.9: World E-Government Developing Index in 2008[11]

came in fourth. Regarding the World, there were large differences between the five regions in terms of e-government readiness. Europe (0.6490) having a clear advantage over the other regions, followed by the Americas (0.4936), Asia (0.4470), Oceania (0.4338) and Africa (0.2739). Asia and Oceania were slightly below the world average (0.4514), while Africa tallied far behind [11]. Figure 2.9.

It is possible to infer from these results that in 2008 Europe had a major increse in E-Government readiness on account that Sweden, Denmark and Norway surpassed the United States in almost 0.5 points.

Two years later in 2010, the first position in E-Government online services was held by the Republic of Korea (1.0000), followed by the United States (0.9365) and Canada (0.8825).

The results by regions of the World changed only in the values losing on average approximately 0.1 points. The ranking stayed the same with Africa (0.2733) on last place and Europe (0.6227) on the first. Second to Europe was the Americas (0.4790) followed by Asia (0.4424) and Oceania (0.4193). Figure 2.10.

Despite a major improvement by the Republic of Korea which was only in 6th place on the ranking in 2008, Asia was still with a poor development compared to Europe and even the Americas. Disparity in richness of Asian countries can explain these results [12].

According to the 2012 United Nations E-government Survey rankings, the Republic of Korea is the world leader (0.9283) followed by the Netherlands (0.9125), the United Kingdom (0.8960) and Denmark (0.8889). On a World regional level, once again Europe (0.7188) is the leader followed by the America (0.5403), Asia (0.4992), Oceania (0.4240) and Africa (0.2780) [13]. Figure 2.11.

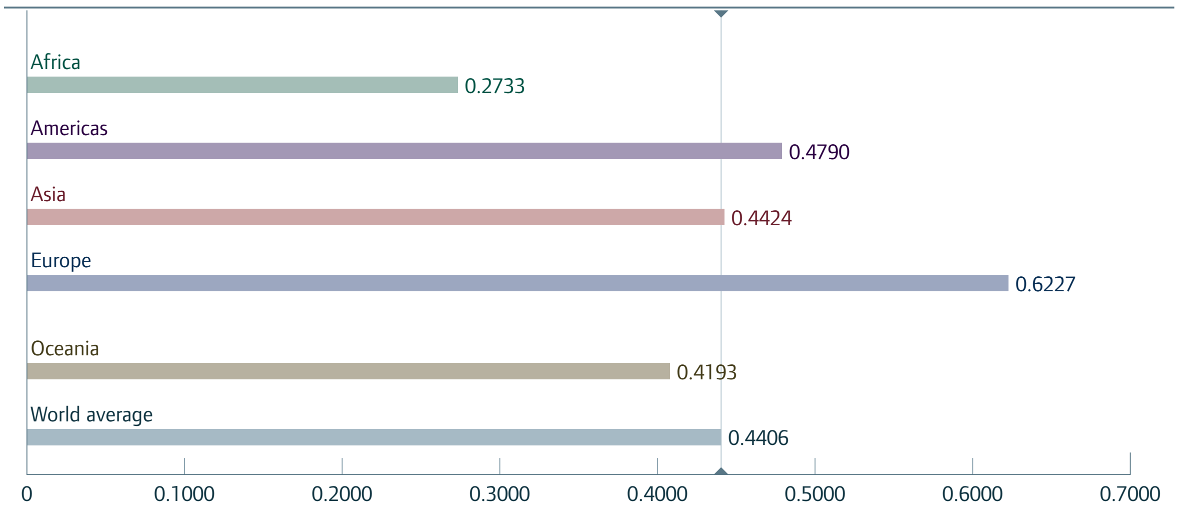


Figure 2.10: World E-Government Developing Index in 2010[12]

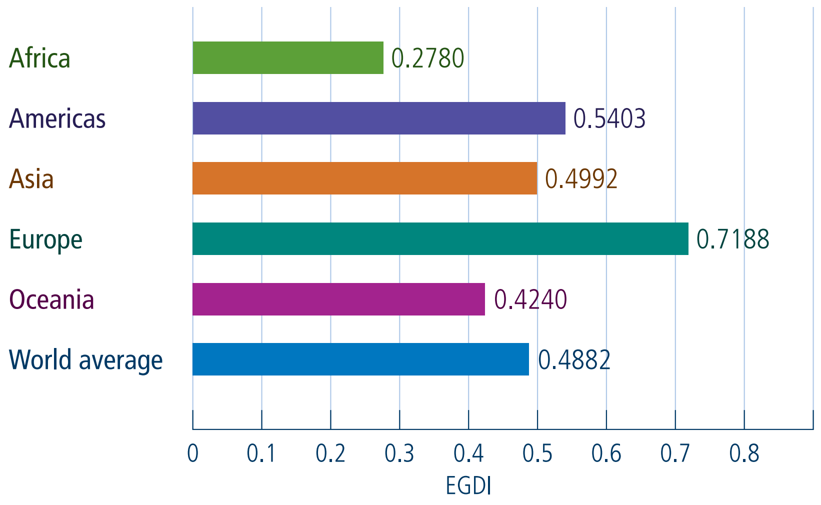


Figure 2.11: World E-Government Developing Index in 2012[13]

Although the African E-Government index of development in the last four years as been almost constant, the other regions have fluctuated values without changing the World ranking. The fact that Africa is in the tail when it concerns E-Government development can be sustained by a social economical situation that affects that region. This presents a problem for E-Government to thrive, but also presents an opportunity to develop and implement low cost or open source types of E-Government platforms, such as for example the Bungeni Parliamentary System developed by the same organization that elaborated these survey. This system, as is shown further ahead was developed by a department inside the United Nations to help giving a quicker response from African countries to the E-Government Development.

2.8 Open Source E-Government Frameworks

With the increase in technology and Internet usage and availability, developers started to contribute on open source projects in a wide variety of areas. From games to general libraries and frameworks, these types of projects intend to atone the unnecessary spending of capital by providing, free or low cost, alternatives.

E-Government as not stand astray from this tendency, frameworks and technologies started to be developed with the intent to lower the cost of ICT's by a governments country, and also facilitate the interoperability between proprietary solutions.

A brief introduction to some open source E-Government projects follows:

EGovFrame Portal [17]

The E-Government Standard Framework has an objective to increase the quality of eGovernment services, the efficiency of IT investment and the standardization and the reusability of application SWs through establishing and applying the development framework standard.

There are many problems in the existing E-Government system because each individual application system uses various kinds and versions of frameworks. This framework intents to standardize the E-Government framework used by one's country, in order to lower the costs of ICT's. Normally a E-Government system needs to be mantained by its vendor, because the modules provided are proprietary and can't be changed by a programmer. This fact develops cyclic payments to maintain the platform, as well as paid training sessions for the staff to learn about the new specifications in the system. The standardization of E-Government Framework eliminates the technical dependency on vendor's proprietary development framework and promotes the standardization. South Korea implemented some services using this framework.

FreE-Gov [18]

Free-gov is a free software system especially designed for e-government, providing Electronic Record Management with full Digital Cryptographic Signature support. It is extremely secure, when properly installed, fast and scalable. It is an ideal platform for e-government implementations of any size and purpose.

It is a modular platform, that can be built according to the government needs. Recently a module to manage municipalities complaints was made available. Supports strong user authentication, logging, record management and digital cryptographic signature, to ensure documents authenticity. It is built using open source software and it is also totally open source.

It is stated on the web page that it can provide high escalation, but it is also stated that it uses "thin clients" relegating all the process management to the server, which may infer performance problems.

ERP5 for Public Administration [19]

ERP5 provides a consistent and unified solution for governements and public administrations looking for a proven, global and scalable open source framework to cover their application needs.

ERP5 covers multiple areas of E-Government such as finance, human resources, asset management, inventory, document management and knowledge management. Successful deployments of ERP5 have been done by organisations in business fields as central bank, city, state and Governmental Agency.

Idega Solutions [20]

The idega solutions have been extensively used for self-service and E-Government type applications. It is a modular system that can accommodate modules with solutions for municipalities and government agencies managing. The solutions have been designed to have a general foundation which makes them applicable to different countries. So far the idega Solutions have been implemented in several government institutions in Iceland and Sweden.

The idega platform also has a dynamic process engine which can be used to build workflows connected to government processes which is in turn can be linked to a central case registry.

Bungeni Parliamentary and Legislative System [21]

Being an initiative of "Africa i-Parliament Action Plan" a programme of UN/DESA, Bungeni aims to increase productivity in African Parliaments at low cost. The system is generic, to attend the needs of different legislation types in different countries. It supplies tools to increase E-Governance, by means of citizen participation on the Portal, and also i-Parliamentary modules to produce efficiently legislative content.

Bungeni provides tools for the drafting, revision, amendment and publishing phases of the document life-cycle and also tools to allow citizens to provide feedback, comments on issues, submit petitions and propose amendment to legislation in discussion.

This parliamentary framework will be discussed ahead on the document.

AT4AM for All [22]

AT4AM is the web-based amendment authoring tool used with great success at the European Parliament since February 2010 by Members and their assistants, and advisors to create and table amendments on the proposals of the European Commission and the Council of the European Union.

AT4AM is the first system of the future XML-based workflow of amendments at the European Parliament. Until February 2013, 250.000 amendments were created with AT4AM.

The European Parliament decided to use the knowledge and the experience gained on the AT4AM project to develop AT4AM for All primarily to help national and regional parliaments to implement there own XML-based amendment authoring systems.

2.9 E-Government Situation on African Countries with Portuguese as Official Language

In Africa, E-Government services are below the World average index level. In the last decade, while other regions of the globe achieved good E-Government index levels, thus achieving also good E-Governance level, Africa struggled on this matter. This can be seen in Figure 2.12.

Noticing the lag of African E-Government developing index, the United Nations created a project to help the development of E-Government Services, specifically National Assemblies legislative document management and system transparency. This initiative was named African I-Parliaments project and was based on open source software.

In the context of this document, special attention should be given to the African Countries with Official Portuguese Language. These countries are Sao Tome and Principe, Cape Verde, Guinea-Bissau, Angola and Mozambique.

Cape Verde is the most active in E-Government services implementation and use of ICT's in the governmental agencies and courts, featuring the top ten of African most developed E-Government countries in 6th place. As an example, the SIIC-2013 ("Sistema de Informação de Investigação Criminal") [23] and SIPC CV ("Sistema de Informatizaç ao do Processo Civil de Cabo Verde") [24], being currently developed on the University of Aveiro.

Sao Tome and Angola have a similar level of E-Government and are above the African average.

The current situation on E-Government, Infrastructure and E-Participation Indexes, can be sum up by Table 2.2. E-Participation measures the E-Governance allowed by the country

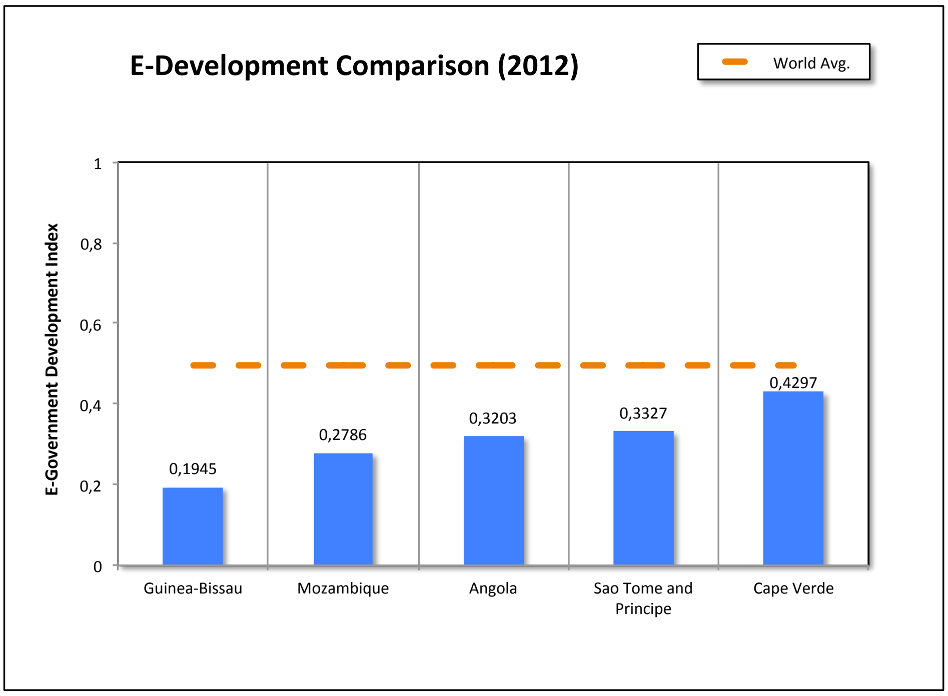


Figure 2.12: E-Government Development Indexes for the African Countries with Portuguese as Official Language

and the Infrastructure Index represents the amount of available technology to implement the services.

Countries	E-Government Index	Infrastructure Index	E-Participation Index
Cape Verde	0.430	0.227	0.237
Sao Tome and Principe	0.333	0.137	0.026
Angola	0.320	0.089	0.026
Mozambique	0.279	0.044	0.132
Guinea-Bissau	0.195	0.051	0.053
World Average	0.496	0.326	0.268

Table 2.2: E-Government Indexes on Lusophone Africa [25]

Despite the current situation each country as taken a different road to get to these positions, on the last decade [25].

Angola had an overall peak on the E-Government Index since 2008, and has maintain the score, having reached its highest index in 2008 (0.3328). This peak in growth was the third highest Worldwide [26]. Figure 2.13

Cape Verde, also suffered a peak in 2008 reaching the highest score in 2012 (0.4297) but the lowest score of this country (0.322 in 2003), was the highest that all the other countries, discussed on this part, reached. Figure 2.14

Mozambique had almost a linear growth in in the last decade, being the only exception

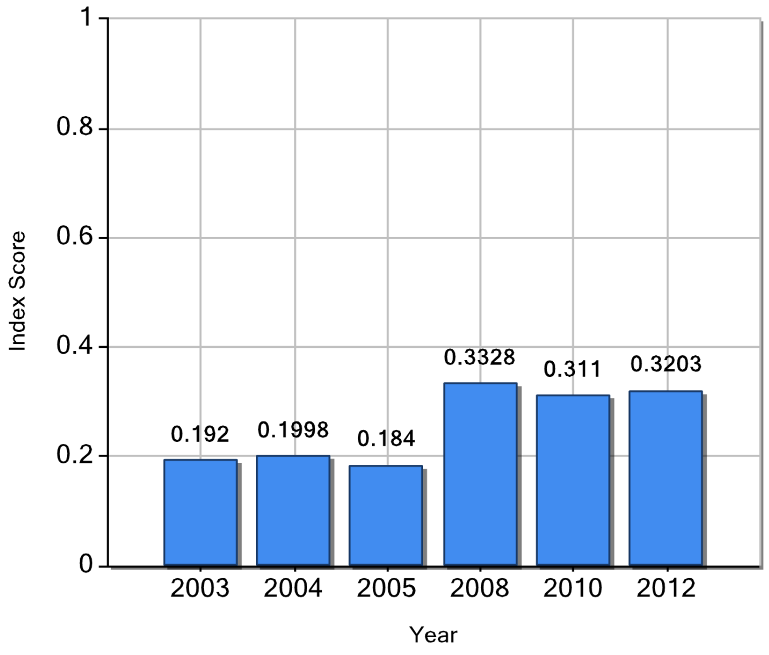


Figure 2.13: Angola E-Government Development Index from 2003 to 2012

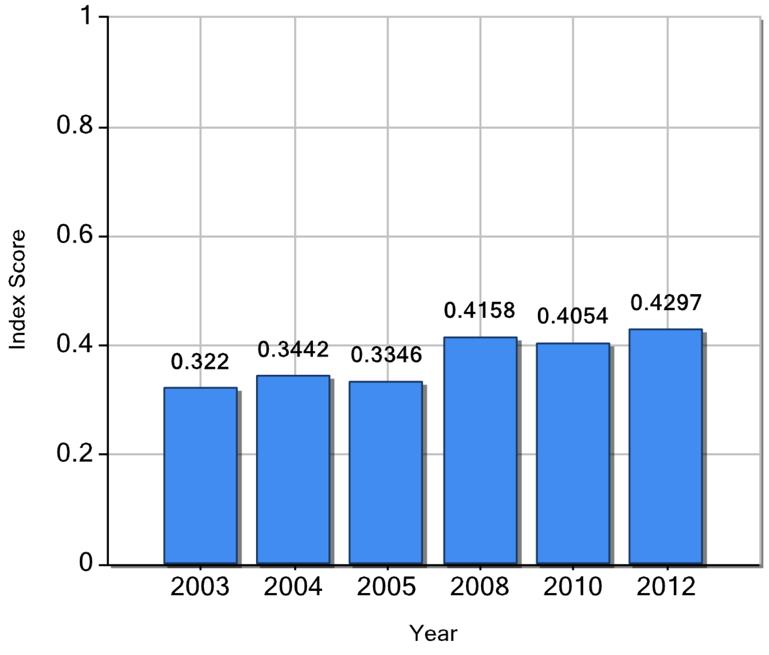


Figure 2.14: Cape Verde E-Government Development Index from 2003 to 2012

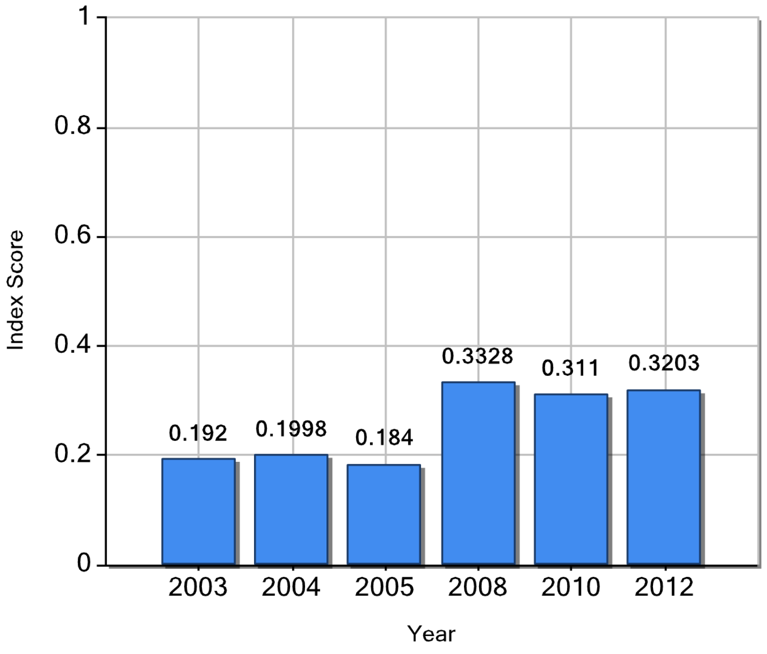


Figure 2.15: Mozambique E-Government Development Index from 2003 to 2012

the year 2010 (0.2288), in which the index score was lower then the two adjacent years (2010 - 0.2288 and 2012 - 0.2786, respectively). Figure 2.15.

Guinea-Bissau on the starting of this measurement had no E-Government services active, and though it started on the next year, its maximum index score (0.1945 in 2012) is the absolute lowest score of these countries, if Angola and Mozambique starting years are ignored. Figure 2.16.

Sao Tome and Principe, also had a peak from 2005 (0.3215) to 2008 (0.3215), but before and after these dates there was no considerable growth on the index. Figure 2.17.

The stagnation on the values indicates roughly, there are other variables to consider such, as socio-economical aspects, that E-Government has not evolved between the years. Peaks indicate that E-Government services thrived and there were innovations.

After analysing all these results it is easy to infer that these countries need better E-Government frameworks and more available services.

2.9.1 Survey on Official Country Sites

On a brief survey, three sites of each of these countries were visited.

The first site was the Official Portal of the country, the second the official Parliament or National Assembly and the third the Citizens Portal. The choice was made, having the intent that the first site is the one a local citizen should address in order to seek information about its Government.

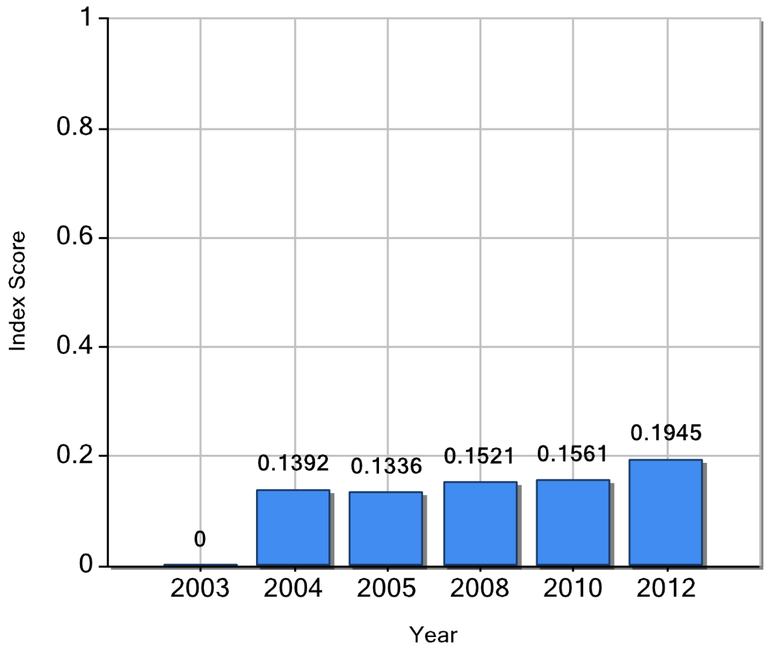


Figure 2.16: Guinea-Bissau E-Government Development Index from 2003 to 2012

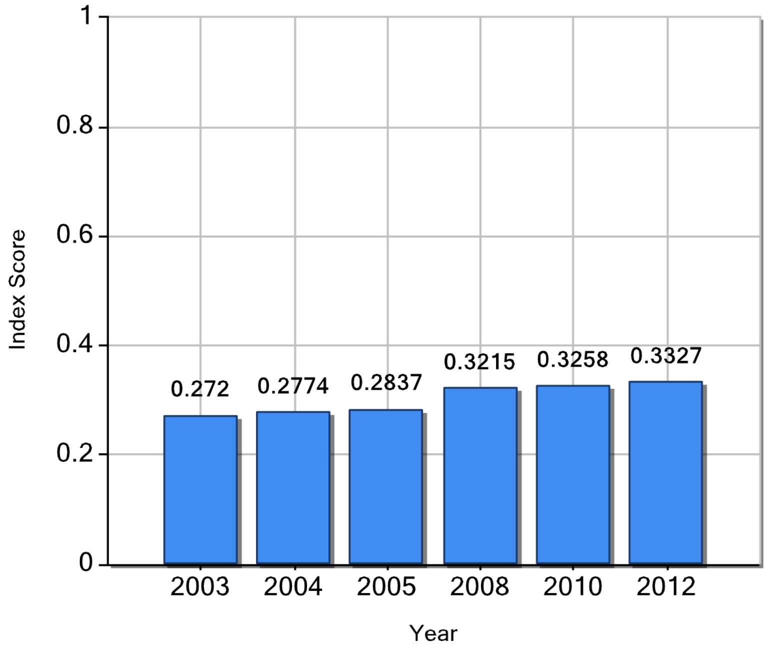


Figure 2.17: Sao Tome and Principe E-Government Development Index from 2003 to 2012

The second one was selected to analyse by respecting the principle that any citizen should be allowed to access the laws or system of rules, by which it should abide.

Finally the third site was chosen because on a general basis is where the Government creates pointers to the E-Government available services, in what concerns Government to citizen interaction.

Starting from Guinea-Bissau, the Portal pointed to no services and the National Assembly only gave information about static laws, having not a central database of legislation. A Citizen Portal could not be found.

Mozambique's Portal, on the contrary, had services accessed only through information or downloading documents for later submission. Regarding the National Assembly page there was no possible analysis, due to not being accessible. Mozambique Citizen Portal has plenty of service information, but in a static form, and provides pointers to other websites government related, that are still active in a partial form.

Sao Tome and Principe Portal was under construction and provided only little information. The National Assembly page is located on a Brazilian server, and provides only static law information, with no legislative database. A Citizen Portal to this country could not be found.

Cape Verde Portal points to other ministries and services, giving a real idea of a E-Governmental Portal. The National Assembly provided access to a database of legislation. Cape Verde's Citizen Portal, allows to submit electronic forms to financial services and makes available information to other services.

Angola's Portal, provides information about services on the site, and has pointers that redirect to E-Government services. The National Assembly site has no legislation database. Angola's Citizen Portal, besides having the most services available only allowed information on how to use the services or where.

To summarize, these countries need to address some aspects of E-Government and E-Governance.

The above stated African I-Parliaments project can help this devolpment regarding the National Assemblies and E-Governance index growth. In fact the use of modern ICT's developed and maintained open source are now being studied to be used by many European governments and even in the European Parliament [27] [28] [29] [30].

Chapter 3

Bungeni Parliamentary System

3.1 Introduction

Bungeni is an e-Government system built by a division of United Nations Department of Economics and Social Affairs (UNDESA), that aims to help to increase production on both the legislative process and ease of access related to this theme by the citizens of one's country.

The system is open source and built mainly in Python language, having been deployed through Plone 2.0 and Zope frameworks. Developed for Linux Debian Based distribution, such as Ubuntu and others [31] and is distributed via debian installation packages that must be installed in a certain order due to module dependencies. It is composed of several sub-services which act as a whole and make the system bigger than the sum of its parts. Being built on the open source philosophy of software development, which encourages everyone to work for a greater good, the system was built by a team that was previously located on Kenya, although having software contributions from people around the World like the author of this work. Everything in Bungeni was designed and developed to be as generic as this type of E-Government systems can be. Such features as language, type of legislation existent in one's country or even all the workflow of developing legislative content can be edited and appropriated by different countries, which makes Bungeni a suitable E-Government system for a great majority of places in the map.

Currently the development process has stopped due to lack of funding, but the system has a good overall capability of production, with almost all the features completed or about to be completed. For this reason it needs rigorous testing before entering real deployment phase on countries.

3.2 Used Technologies Overview

In this section, a brief overview of the technologies used to develop the Bungeni Parliamentary System is provided. The use of each of the technologies are discussed in each of the modules that implements it.

Fabric [32]

Fabric is a Python (2.5 or higher) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks.

It provides a basic suite of operations for executing local or remote shell commands (normally or via sudo) and uploading/downloading files, as well as auxiliary functionality such as prompting the running user for input, or aborting execution.

Typical use involves creating a Python module containing one or more functions, then executing them via the fab command-line tool.

Supervisord [33]

The supervisord is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. It is responsible for starting programs at its own invocation, responding to commands from clients, restarting crashed or exited subprocesses, logging its subprocesses visually or physically using file. It provides a web graphical interface to manage processes.

Paster [34]

Python Paste is a set of libraries to deploy WSGI applications, it covers all aspect of a CGI application: testing, dispatching, authentication, debugging and deployment.

It uses HTTP protocol and the WSGI application standard to find and configure applications and servers using a configuration file or a Python Egg (python compiled module).

Deliverance [35]

Deliverance is an integration tool, that allows different applications built on frameworks or languages to be integrated gracefully.

Deliverance does this by parsing the HTML code that results from web applications and applies a “theme”, granting a common look-and-feel across all the applications in the site. The transformations are done using DOM manipulation by similar to XSLT transforms.

Zope [36]

Zope is an open source web application server primarily written in the Python programming language. It features a transactional object database which can store not only content and custom data, but also dynamic HTML templates and scripts. It also features a search engine, a relational database and distributed control, allowing to create distinct roles for different types of System Administrators. Despite all these features, the ones that call for more developers are the built in Transactional Support and the strong Security Mechanisms.

Postgres SQL [37]

PostgresQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. An enterprise class database, PostgresQL boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance.

SQLAlchemy [38]

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper (ORM) that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language. SQLAlchemy considers the database to be a relational algebra engine, not just a collection of tables. Rows can be selected from not only tables but also joins and other select statements to compose larger structures. SQLAlchemy is most famous for its object-relational mapper (ORM), an optional component that provides the data mapper pattern, where classes can be mapped to the database in open ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

Xapian [39]

Xapian is an Open Source Search Engine Library highly adaptable toolkit which allows developers to easily add advanced indexing and search facilities to their own applications. It supports the Probabilistic Information Retrieval model and also supports a rich set of boolean query operators.

Plone [40]

Plone is a powerful, flexible Content Management solution that is easy to install, use and extend. Plone lets non-technical people create and maintain information using only a web browser. The main use of Plone is as base for web sites or intranets because its modular nature helps the customization of all aspects. Plone is a product for Zope2 application server, so it shares the core functionalities.

Exist-DB [41]

Exist-DB is a XML object database that is Schema-less. The high-performance native XML database engine stores textual or binary data and documents without requiring a database schema, creating flexibility for the developers. It has a browser-based Integrated Development Environment (IDE), to explore the objects visually. The querys to retrieve data are done using the XQuery standard.

3.3 Architecture

[Bungeni Services Scripts] - The Bungeni system relies on a set of services that need to be working simultaneously for the whole to behave in full potential. Some of these services rely on one another (thus requiring cascade activation), some are independent, but overall their interfaces behave like Figure 3.1 shows.

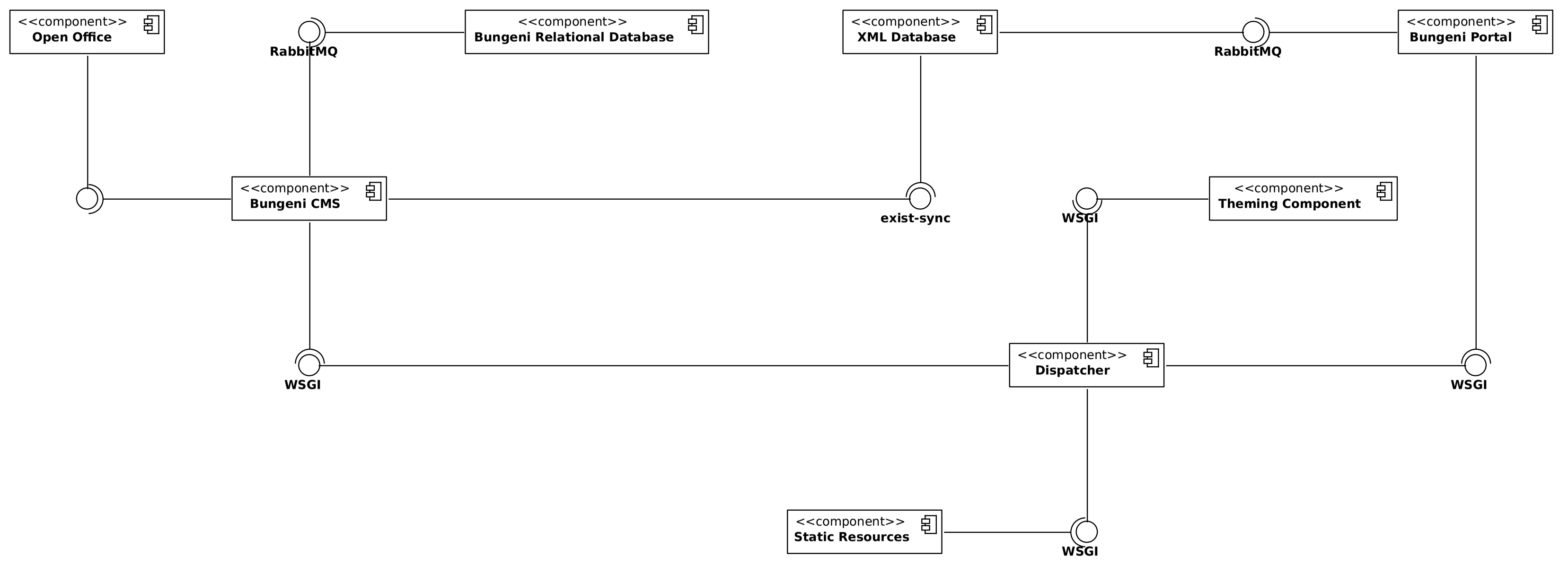
The system is activated by running a set of scripts done via Fabric which is a Python library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks. The scripts can be even run remotely due to this framework on a command line environment or via a graphical interface on the browser created using Supervisord which is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems (Figure 3.2). After activation of the Fabric scripts each module can be loaded by using a secure Fabric script, or as stated before, by running the graphical interface. Both ways of loading the scripts are protected via a user name Password security pair that can initially be configured to the system administrator preferences. Usually the systems are run inside the institution's intra-net like the diagram shows, because exposing such delicate configuration tools to the outside World can be considered a major security fault. The protection of the system will solely rely on the user name password key pair. Besides that, leaving the system exposed can be really problematic because these services run on unprotected ports and can be accessed by ill willed users.

3.3.1 Databases

Exist

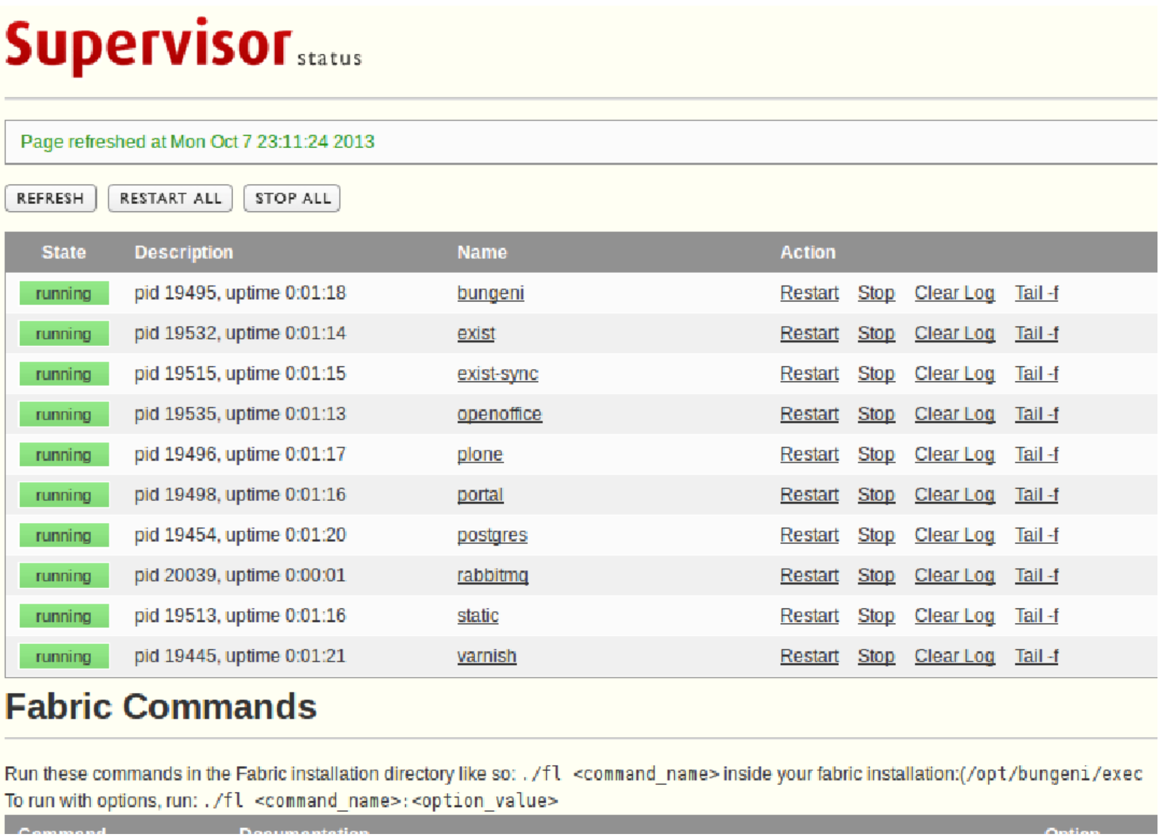
Exist is a service based on exist-db, which is no more than an open source native XML database. Basically Bungeni has 2 types of database, exist-db which is only used for serialized legislative documents or Akoma Ntoso files (after serialization only), and a relational Postgres database discussed further on the document. This XML database has the data accessed via REST API which can be quicker and easier for the system to process visually to the citizen accessing the system via Portal (discussed ahead). Usually in Bungeni, the exist-db only stores information that will be shared with citizens on the Internet, leaving internal data to the relational database. The stored information can be data that is workflowable, which is data that has a given sequence similar to a partitioned task in order to be completed. The workflowable content contains every legislative document serialized and ready to be showed Online. The serialization done before storing data is no more than categorizing documents by items and storing those items indexed in order for the system to gather them faster before the visual presentation.

The exist-db needs to be flexible because Bungeni is generic and accepts new types of workflowable documents that can be introduced in the system. In example, in a bicameral



<<component>> Open Office
RabbitMQ
<<component>> Bungeni Relational Database
<<component>> XML Database
RabbitMQ
<<component>> Bungeni Portal
<<component>> Bungeni CMS
exist-sync
WSGI
<<component>> Theming Component
WSGI
<<component>> Dispatcher
WSGI
<<component>> Static Resources
WSGI

Figure 3.1: Bungeni System Components and Interfaces



Page refreshed at Mon Oct 7 23:11:24 2013

State	Description	Name	Action
running	pid 19495, uptime 0:01:18	bungeni	Restart Stop Clear Log Tail -f
running	pid 19532, uptime 0:01:14	exist	Restart Stop Clear Log Tail -f
running	pid 19515, uptime 0:01:15	exist-sync	Restart Stop Clear Log Tail -f
running	pid 19535, uptime 0:01:13	openoffice	Restart Stop Clear Log Tail -f
running	pid 19496, uptime 0:01:17	plone	Restart Stop Clear Log Tail -f
running	pid 19498, uptime 0:01:16	portal	Restart Stop Clear Log Tail -f
running	pid 19454, uptime 0:01:20	postgres	Restart Stop Clear Log Tail -f
running	pid 20039, uptime 0:00:01	rabbitmq	Restart Stop Clear Log Tail -f
running	pid 19513, uptime 0:01:16	static	Restart Stop Clear Log Tail -f
running	pid 19445, uptime 0:01:21	varnish	Restart Stop Clear Log Tail -f

Fabric Commands

Run these commands in the Fabric installation directory like so: `./fl <command_name>` inside your fabric installation:(`/opt/bungeni/exec`
To run with options, run: `./fl <command_name>:<option_value>`

Figure 3.2: Supervisord Interface for Bungeni Services Management

type of legislation there can be information changed between the chambers. Although Bungeni supports this type of information exchange with Bicameral Documents, there may be the need of adding a more specific one like Bicameral Motions that has different document items as the previously stated. If exist-db was not flexible enough it would compromise the generic approach of Bungeni and thus not allow for specificity of that kind.

Exist-Sync

This service acts like a bridge in putting the Documents on the exist-db and also getting them to display on the visual interface. The serialization is implemented in this module, and it is as simple as itemize documents stored on the Bungeni relational Postgres database and storing them in categories on the XML exist-db. It is a simple module that processes each file individually or as a batch, before storing the data on exist-db. Can be accessed using the exist-db interface connected to the Bungeni core pressing the "serialization manager" button. Every time the service is restarted it will check for differences between the relational Postgres database and the exist-db and as the name suggests, begin the synchronization of the data between the two.

Bungeni DB

This component is the main database used in the Bungeni Parliamentary System. This is a relational database and uses Postgres [37] open-source database Management System. Data stored on this database comes from the Bungeni Portal or BungeniCMS by the way of RabbitMQ module, which is used to provide better response times and also data serialization. The data exchanged is done by SQL Query and Retrieve to the system modules, that ask for information. This database can contain a lot of information comparing it to the exist-db. It stores from user data (with security on the password) to legislative content (including status of workflowing documents), as well as every object that as a representation on the Bungeni Parliamentary System. Before data is serialized to exist-db it comes from the Postgres relational database.

Like the exist-db needs to be flexible, because Bungeni is generic and accepts new types of workflowable documents that can be introduced in the system, so does need the Postgres database. As a previous example given in the description of exist-db, a bicameral document used in the type of bicameral legislation is always stored first in this database than in exist-db. If Postgres was not flexible it would compromise the generic approach of Bungeni and thus not allowing its deployment across multiple countries with different legislative types. This flexibility is simulated by using tables that have generic content. Each stored object has a collection of attributes that are the same as newly created objects. This way, objects like different types of workflows, for instance, are created but have the same basic info.

RabbitMQ

This component provides an interface between the Postgres relational database and the BungeniCMS or Bungeni Portal. It is based on the RabbitMQ [42] messaging system that is a robust platform for intra applications messaging, taking in account concurrency and performance. As a service in Bungeni it acts as a SQL Query and Retrieve messaging carrier, between the database and the system core. In early versions of the Bungeni Parliamentary system, RabbitMQ was not in the mandatory stack of packages needed by the system, but some of the performance increased since its use.

3.3.2 Theming and Styling

Theme Delivering

This component provides a coherent look-and-feel across all applications on the site. The HTML coming from a source (e.g. BungeniPortal or BungeniCMS) is re-written based on a "theme". The "theme" is written in a static HTML page that has metadata embedded, but also uses css files and images to give a better look. On a a reply, the HTML coming from the applications is appended to the page metadata, creating new content which is then displayed to the end user on the browser. This operation is done using a set of rules based on a XPath syntax and altering the DOM. The main technology associated with this module is the one provided by Deliverance [35]. "Themes" for some African countries are already provided within the Bungeni Parliamentary system and can be activated using the Fabric scripts.

Dispatcher

This component simply call the application using a mapping between URL's and applications. Basically each application belonging to the Bungeni Parliamentary System has a set of URL's, (like a web service REST API) that are deployed and ready. This module uses paste script or "Paster" [34] to map the URL's coming from the server WSGI to the corresponding application and ensures the way back on the reply.

Static Resources

Static Resources are images and styling content provided via WSGI on the return of HTTP requests. Static [43] is a python library that provides an easy way to include static content in the WSGI applications. The Theme Delivering service accesses the static resources using the Dispatcher service before sending the HTTP styled request to the browser. The content is accessed via URL's, like the applications in the Bungeni Parliamentary System, so the interface used to retrieve information is the same provided inside the Dispatcher.

3.3.3 General Components

Open Office

Bungeni Parliamentary System has bindings with Open Office built in the Operating System interfaces. Open Office [44] is a Microsoft Office like set of tools created for office production purposes. It is open-source as well as multi platform. Nowadays the production team of Open Office as forked the project, and another set of tools using the same interfaces was created, namely Libre Office [45]. Because the interfaces are the same, Libre Office and Open Office both work flawlessly with the Bungeni Parliamentary System, but only Open Office will referredto, from this point on. The Bungeni Parliamentary System uses the interfaces for operations, such as buildind .doc or .odt formated files based on the Akoma Ntoso standard. Open Office was also used to produce downloadable content, such as PDF files related to reports or documents that contained legislative content, as well as agenda file types. The format or look for the downloadable content was provided using Open Office documents and a set of XML files. Now the support is provided by html files, and the Open Office styling was deprecated.

WSGI

WSGI [46] is an acronym for Web Server Gateway Interface. It is a specification for web servers and application servers to communicate with web applications (though it can also be used for more than that). It is a Python standard. It is itself basically a type of request and dispatching mechanism framework based on URL processing for server sided applications. On the Bungeni Parliamentary System it implements part of the "Dispatcher" service used on the bottom of the Zope3 [36] stack.

3.3.4 Core Application

BungeniPortal

This is the application that provides specific parliament features. Through the use of this application, one can access legislative content and data regarding various sections of information. BungeniPortal was developed in order to use a Bicameral Type of Parliament (Higher House and Lower House). The information type displayed on each of the Houses is the same. To exclude unnecessary repetition only the Lower House will be used in this example.

The information accessible in both the Lower and Higher Houses can be break up in the following sections:

- National Assembly: information regarding the active Legislature in the Lower House;

- Business: in this area there are the daily operations of the various activities of the House:
  - What's on: overview of the daily operations of the House, basically a news feed that as a Real Simple Syndication (RSS) [47] interface;
  - Committees: list of committees, for each one there is metadata about the matter of discussion, membership and sittings regarding the House Legislation;
  - Bills: list of bills and associated metadata. Actions are provided to version the bill and access the workflow associated with the bill, as well as download the documents in various digital formats;
  - Questions: list of questions and associated metadata. Workflow and versioning actions are provided, as well as download operations for the documents in various digital formats;
  - Motions: list of motions and associated metadata. Workflow and versioning actions are provided, as well as download operations for the documents in various digital formats;
  - Tabled documents: list of tabled documents and metadata. Workflow and versioning actions are provided, as well as download operations for the documents in various digital formats
  - Agenda items: list of agenda items and metadata;
  - Calendar: calendar showing the sittings of the plenary and the committees. The information is also accessible via RSS interface;
  - Publications: list of publications and informations, these publications are the reports of sittings and general information regarding publicized documents, such as legislative content related data;
- Members: here one can search for information about members of the House:
  - Member: trivia information about a member of the House;
  - Biographical Data: a complete biography of the member;
  - Offices held: informations about offices in which the member has a title;
  - Parliament activities: a list of legislative content the member has Participated in. e.g. submitted Questions or become a bill signatory;
  - Contacts: information regarding the member list of contacts;
  - Archive: access to current and historical activities of the parliament, the categories are:
    - Parliaments;
    - Political groups;
    - Committees;
    - Governments.

Graphical Interfaces that need access through means of a secure login:

- Workspace: This is available for members of parliament and for clerks. It is a graphical interface to manage legislative content. It simulates a real World workspace environment, by filing the documents under each state folder name, in e.g. “Draft”, “In Progress”. There is also provided information relative to the role groups that the clerk or member of parliament is associated to;
- Administration: This is an administration section provided to the Administrator. This is used for adding parliaments, new users, closing parliaments, entering preliminary metadata, and general manage of the information that appears on BungeniPortal.

The Figure 3.3 can give a more illustrated and perception depth on how the contents are related in BungeniPortal. For the graphical information to be more ease to the view, a simplification is made and only the Lower House was used for example purposes, but the system as also an Higher House. To understand the full BungeniPortal both Higher and Lower House behave the same way, so the diagram would be an exact repetition only with the change of words from Lower to Higher.

The versions and workflow functionalities provide traversals into the content; for example in a motion there are the links to past workflow states and older versions of the motion - allowing the user to browse not just the current state of the motion but also the full audited history of the motion.

Information is also related to itself, and can be accessed in many ways like the connected components show. To give an example, it is possible to access bills moved by a member using the “Parliamentary Activities” interface or just browsing the member and discover the content associated with him.

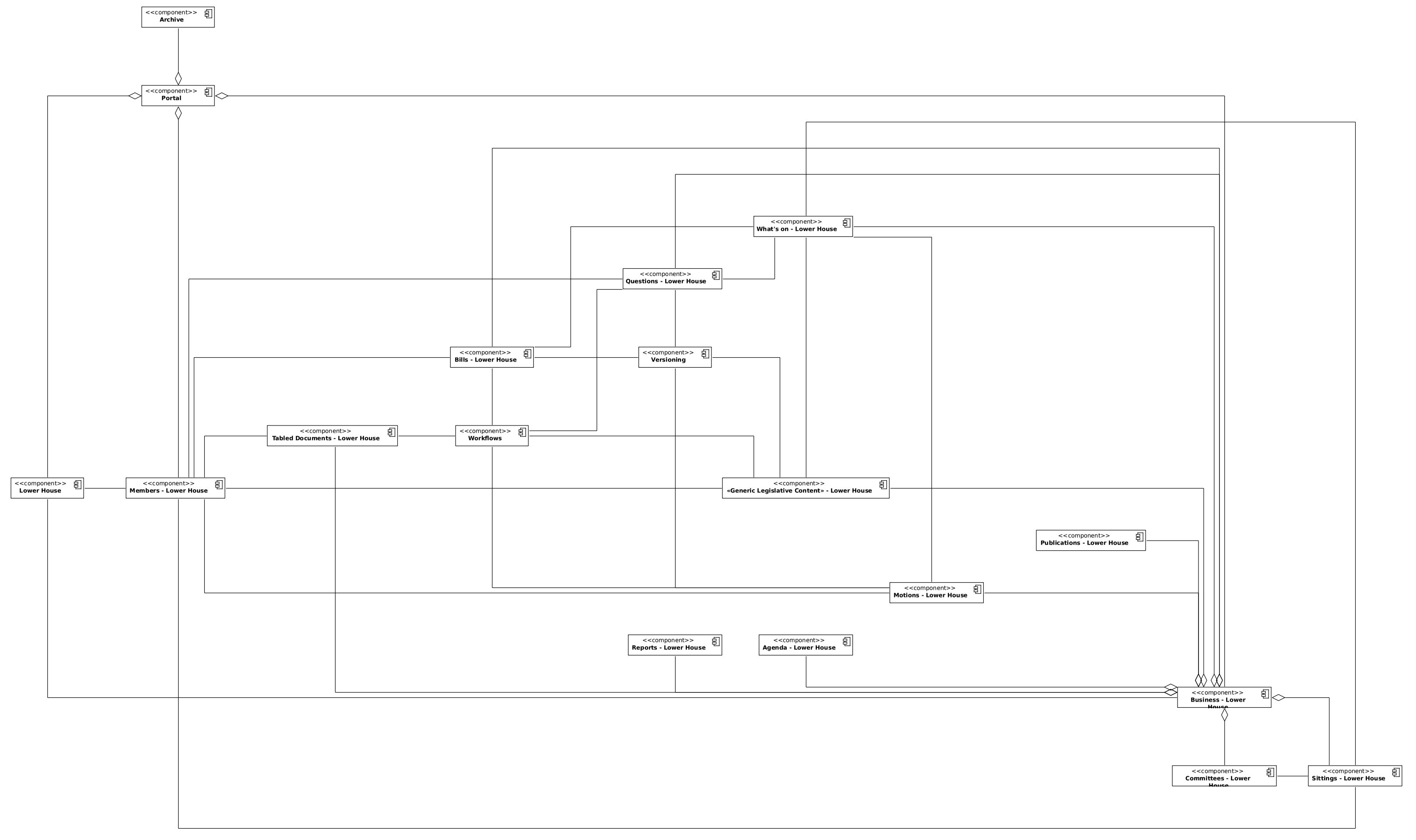
Though the BungeniPortal is itself a component, it can be taken apart to smaller components like the Figure 3.4 shows.

Inside the BungeniPortal there are 7 packages, or components namely server, portal, core, models and UI. A brief description of each one follows:

Server

Application based on the Zope3 application server. The main package contains a file with the configuration of the libraries that are loaded on the deployment of the Bungeni Parliamentary System, reducing the overhead time of loading all the libraries of the Zope3 platform. It also contains the “application_factory” function and header so that the “Dispatcher” service, based on “Paster” can launch the server. It uses ore.wsgiapp to provide access to the ”application_factory” function, otherwise “Paster” couldn’t address the request.

The package also contains SMTPMailer utility used for sending e-mail from within Zope3.



<<component>> Archive
<<component>> Portal
<<component>> What's on - Lower House
<<component>> Questions - Lower House
<<component>> Bills - Lower House
<<component>> Versioning
<<component>> Tabled Documents - Lower House
<<component>> Workflows
<<component>> Lower House
<<component>> Members - Lower House
<<component>> «Generic Legislative Content» - Lower House
<<component>> Publications - Lower House
<<component>> Motions - Lower House
<<component>> Reports - Lower House
<<component>> Agenda - Lower House
<<component>> Business - Lower House
<<component>> Committees - Lower House
<<component>> Sittings - Lower House

Figure 3.3: Logical Components of Bungeni Portal

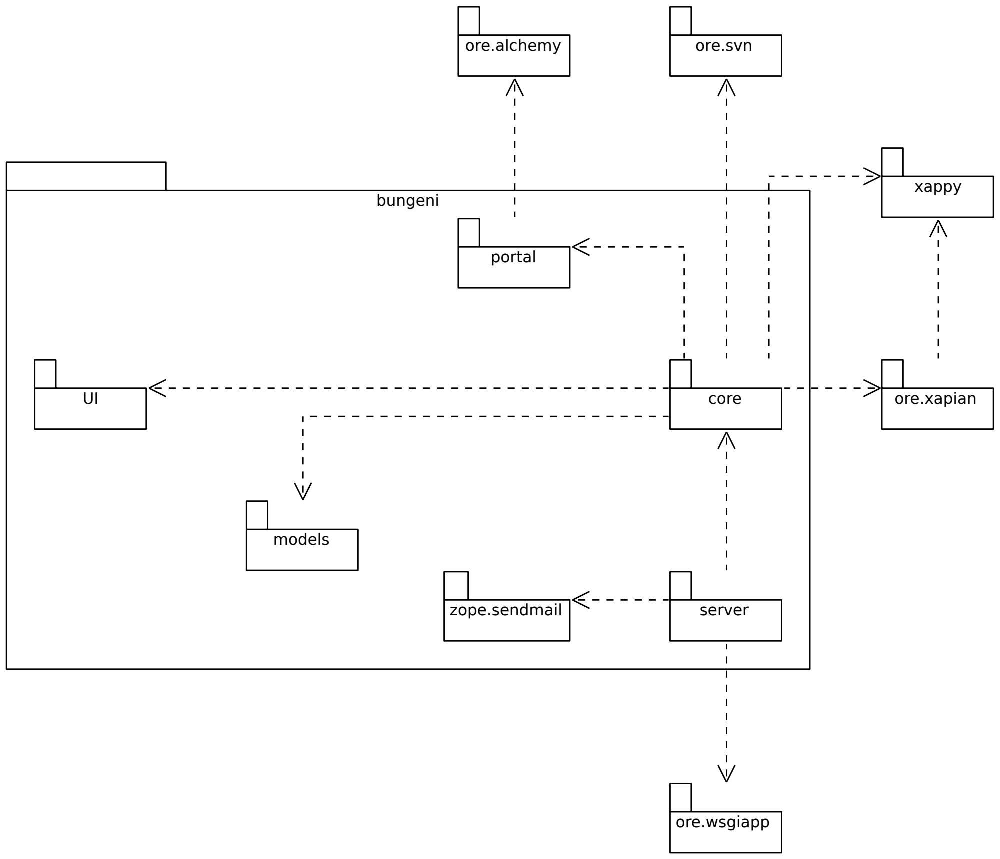


Figure 3.4: Package Diagram of Bungeni Portal

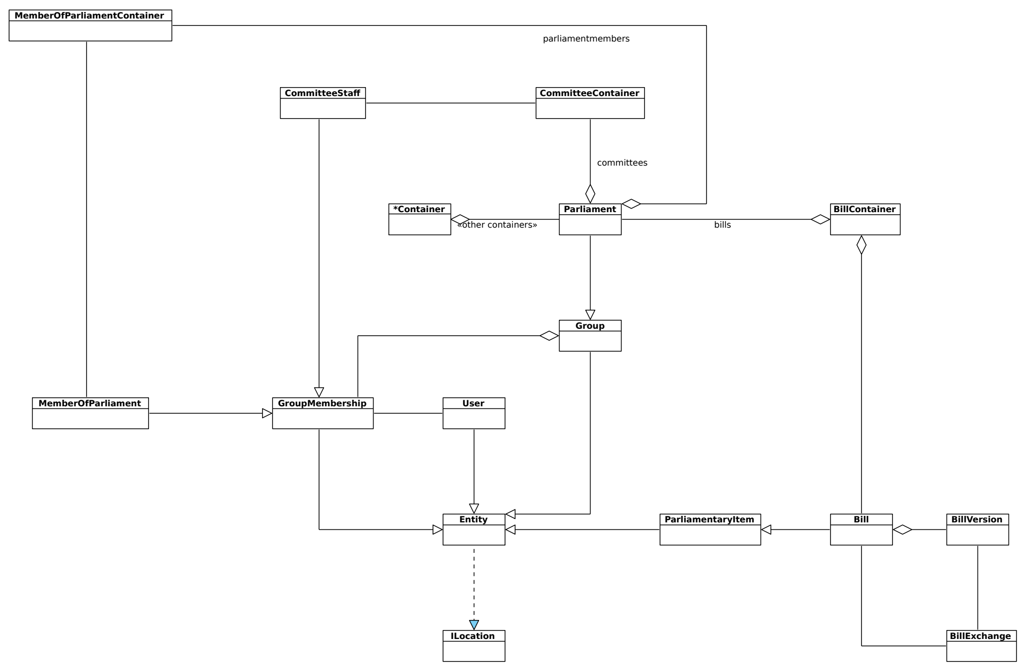


Figure 3.5: Bungeni.models Classes Diagram

Portal

Contains configurations for the Relational database, using “ore.alchemist” (a wrapper for SQLAlchemy) and also the declaration of the visual base template and functionalities. It also defines the “make_static_serving_app” function invoked by the “Dispatcher” service using “Paster” for static styling objects, contained within the “static” folder and sub-folders. Lastly, also contains the rules and themes for “Deliverance” that will be applied using “Theme Delivering” service in every HTML reply.

Models

This is the part of the system that matches and accesses data on the Relational database, using ore.alchemist on top of ore.Alchemy.

This package is constituted by the classes and containers that represent a Parliament be it either a Lower House or a Higher House with all its members and content. Each model is treated as an Entity to be located on the system.

As scenario of implementation, and to better understand some of the relations on the Figure 3.5, an example is provided relating one object of the system to its counterparts on the Relational database.

Being “bills” the name of a “BillContainer” object on the system, containing Bill objects and if each Bill object is a ParlamentaryItem, in terms of Relational Data Base, this means that for each row in Bill table it is created a row in the ParlamentaryItem table.

Versioning on Bill objects works by accessing and changing the BillVersion if a change occurs in a Bill object. When a change is detected, automatically a ParlamentaryItem row in the Relational database is also changed and the old values are stored in a new BillVersion table row. Finally a new BillChange is generated.

UI

This package is composed by a collection of packages responsible for part of the look and feel of the Bungeni Parliamentary System. They contain the objects used in the user interface as well as the structure of the web pages. Despite the existence of a minor styling being present on this module, the responsibility for theming the applications is of the "Theme Delivering" service, via Deliverance.

The collection of packages includes:

- forms package: content and container browser views (web pages structure);
- calendar package: interfaces and code to implements the operations involving scheduling, in e.g sittings agenda;
- workspaces.py: manage the access to items of a federated user in the system that is a member of parliament or a clerk;
- workflow.py: manage the interface to access the documents workflow functionalities, such as advancing states and checking permissions on each state;
- versions.py: provide the functionalities to access the documents versions. Useful when one wants to see the changes that occurred in a document during a timespan.

Core

Contains the application started from the "application_factory" entry point of Server and the contents creating the sections of the Portal. Basically unifies the Portal and Server packages in a Core service. Contains the main functions that are applied and invoked by "Paster", when the BungeniPortal service is used, including the ones that create each section of the Portal and also the workflowable documents management code.

Because the package is too complex, Figure 3.6 is an attempt to create a better understanding of it.

Analizing the Figure 3.6 it is possible to infer some understanding about this part of the Bungeni Parliamentary System.

Breaking apart the diagram a brief description of the interactors, follow:

bungeni.core.app

The main class is of this component (or interactor in this case) is AppSetup. This class is the factory adapter for the BungeniApp (IBungeniApplication) and setups

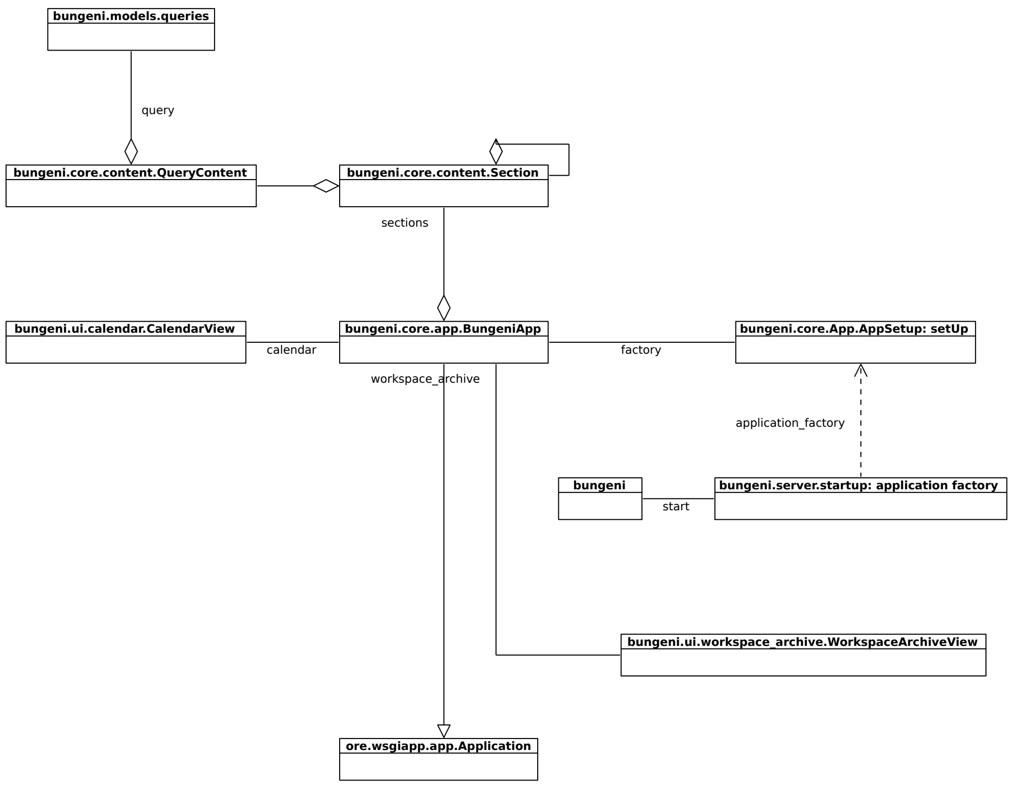

bungeni.models.queries
query
bungeni.core.content.QueryContent
bungeni.core.content.Section
sections
bungeni.ui.calendar.CalendarView
calendar
bungeni.core.app.BungeniApp
factory
bungeni.core.App.AppSetup: setUp
workspace_archive
application_factory
bungeni
start
bungeni.server.startup: application factory
bungeni.ui.workspace_archive.WorkspaceArchiveView
ore.wsgiapp.app.Application

Figure 3.6: Bungeni.core interaction Diagram

the the application, by creating indexes for each content and add the previously created to the "indexer" object that wraps Xapian [39]. It also creates the names that will relate to the URL's used by the "Dispatcher" service, implemented by Paster.

Portal is composed of sections, that were listed previously. These sections are based on four classes, depending on the content or role of a user in the system. Namely they are:

bungeni.core.content.Section
It is an OrderedContainer [48], a Zope3 folder like class in which the contents are kept ordered. For example "National Assembly", "Business", "Members" are Section contents. Usually the OrderedContainers are Persistent objects (in Zope sense) but in this case they are not stored at all.

bungeni.core.content.QueryContent
It is an object at which is attached a function performing SQL queries. In a basic sense it acts and retrieves the content of a query.

bungeni.ui.calendar.CalendarView
Browser type of view page that provides calendar functionalities, by the means of a special date time control.

bungeni.ui.workspace.archive.WorkspaceArchiveView
Browser type of federated view page that provides access to a set of functionalities related to the work of Clerks.

BungeniCMS

The BungeniCMS contains the general materials of the Bungeni Parliamentary System. It is the content management part of the system. This part of the system is developed in Plone2 [40], which is a rich a Content Management System, built on top of Zope3 Web Application. According to the site: "Plone is among the top 2 per cent of all open source projects worldwide, and has the best security track record of any major CMS.", thus proving it to be suitable for an E-Government Platform. Since Plone is flexible, new features can be easily added to BungeniCMS, like for example a Blog system for the members of the Parliament. This module is of vital importance, because as the name implies it is where the information is managed, and an E-Government platform is only needed and used when there is information to be shared with the citizens. The BungeniCMS synchronizes the relational and XML object database and is responsible for the information that shows on BungeniPortal.

BungeniCMS also provides a set of functionalities which are designed to:

- Allow a large number of people to contribute to and share stored contents, via RSS and sharing digital content;

- Control access to contents, based on permissions or (Roles);
- User roles or group membership define what each user can do (not only edit and view);
- Improve communication between users using comments on contents;
- Publication workflow and versioning support;

The CMS can hold various contents: documents, events, news, pictures, legislative content, attached files, etc. The information architecture is organized in a tree structure:

- How we work;
- Rules and regulations;
- How parliament works;
- Seating plan;
- Administrative;
- Reference material;
- History of parliament;
- Online resources;
- Useful links;
- Picture gallery;
- Have your say;
- Vote in the election;
- Become an member of parliament;
- Present a petition;
- Visit parliament.

3.3.5 Workflow System

Workflows Introduction

In the Bungeni Parliamentary System, there is the existence of certain legislative documents that use workflows. A workflow can be seen as a set of small tasks that need to be accomplished before a document reaches its final state.

To facilitate reader's understanding, an example is provided in the form of a workflow that this document has passed. In Figure 3.7 a roughly workflow is demonstrated as a collection of (small tasks) states that must be done for the approval of a University de Aveiro Dissertation. The candidate must write a document similiar to this in structure, send it to revision by the Professor in charge before Submission. After that the candidate can be Approved or Fail, but notice that to be Approved a candidate cannot go from Writing to this final state.

A workflow for legislative content works the same way, with a difference on the number of states and the type of document.

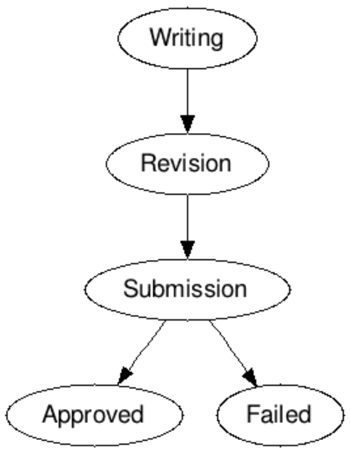


Figure 3.7: Workflow Example

Bungeni Parliamentary System Workflows

Workflowable Documents or commonly named workflows, are defined in the system by using XML files accordingly to certain specifications.

The modules that provide management of workflows on the system are 3.

First the bungeni.core.workflows component where there are configurations, factories and definitions of workflows used in the site.

The second module is bungeni.capi and basically is a "parsing" module, for new workflowable documents structure that can be submitted on the system. Inside this module there is a schema file written in RELAX NG [49] language. This schema file is used by a parser program to compare if the grammatic contained on a workflow file is valid.

Lastly the bungeni.models, where the structure modelling, access to the documents and integration with the relational database is done. Figure 3.8 follows to grant a visual aid and comprehension.

The system is already packed with some workflows that despite being fully functional are only usable to a certain degree of granularity, due to different countries and policies around the legislative process. These workflows are more suited to be used as examples and a good starting point to begin the process of more detailed workflow production, following a set of instructions [50]. Namely the ones present on the default system are:

- motion;
- bill;
- question;
- tabled documents;
- bicameral document.

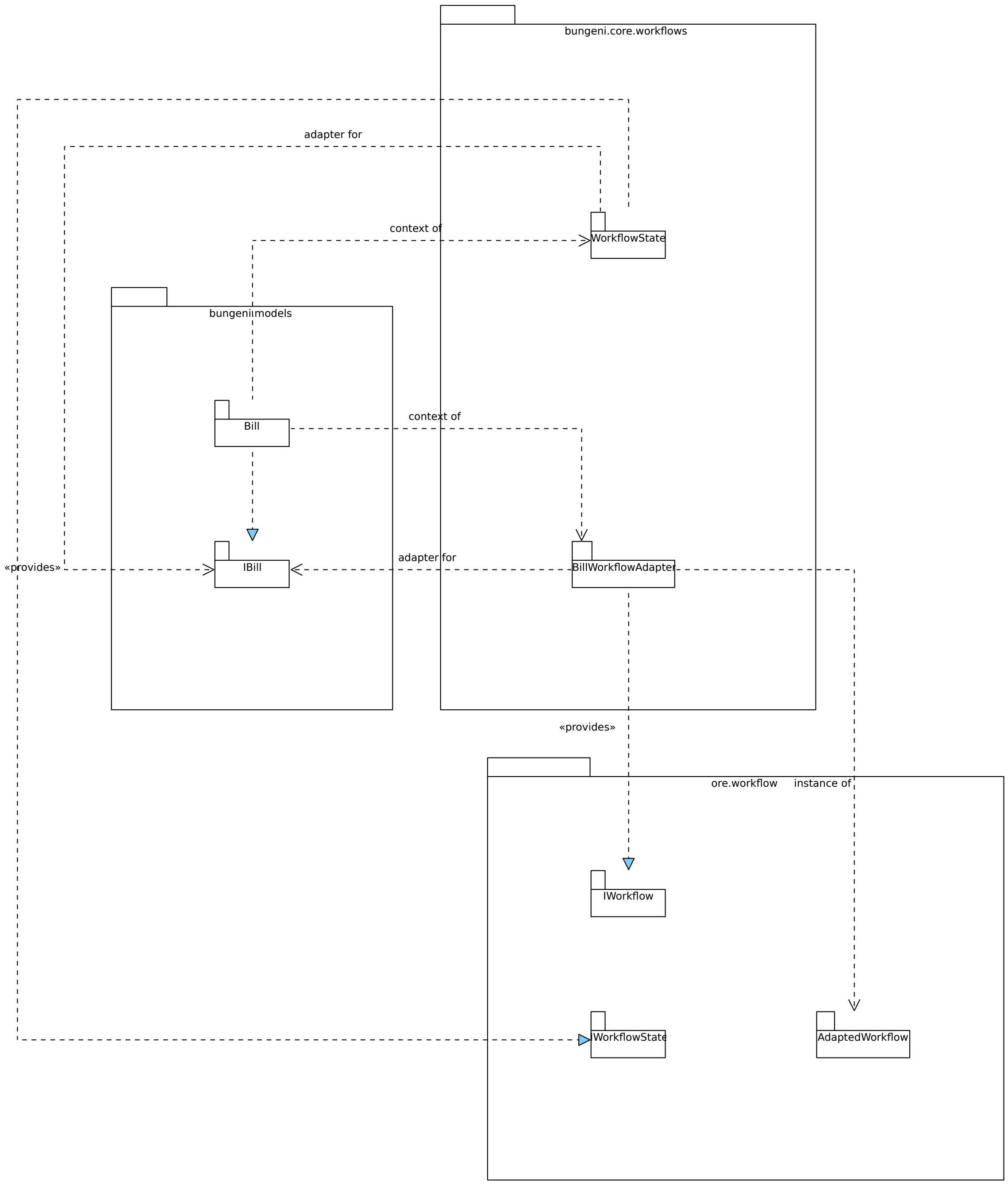


Figure 3.8: Workflows on bungeni.core

All these types of documents are available in both Lower and Higher House, without any change in the structure, hence the simplification opposed to the repetition.

The Bungeni Parliamentary System is really flexible on this part, by allowing the change or creation of new parliamentary documents to be workflowed.

When a new workflow is inserted into the system it needs to be restarted in order to be able to use it, otherwise it will not appear and thus is as it never existed.

Workflow Structure

A workflow definition file can be broke up into 3 major blocks:

1. states;
2. transitions;
3. permissions.

States

A state is a stage like a moment that a document must undergo. On the example provided on Figure 3.7: Writing, Revision, Submission, Approved and Fail are the states.

A transition is the connection between states. On the example, transitions are the arrows that construct the path that a document must undergo between states. Transitions, imply on a lighter note a restriction, by not allowing a document to follow states as desired but following an obligatory set of rules before completion.

To formalize, if a workflow is treated as a digraph, then edges are transitions and nodes are states.

Finally, permissions are defined by facets and allows. A facet is a permission mechanism that informs the system of which users have access to a document on certain state. Facets can be created on the system and have some level of complexity by allowing a description of several users accesses to states. Some facets are created on the system with the principle that they will be used in more than on type of document, thus taking away the need to repeat configuration code. Allows, are the general permission instructions and can be found in either the workflow body or inside facets. On the workflow body, they describe the permissions by the users of the system on the management of a document that uses that workflow. For example an allow can grant general permissions for a clerk to be able to generate new documents of that workflow, but only allow a member of parliament to view and not edit it. Inside facets, allows work in the same way, by preventing or allowing changes on the document that can happen in the state where the facet was declared.

3.3.6 Flow of Information

Using the component diagram presented on Figure 3.1, a normal flow of information can be established and represented in Figure 3.9 for a better understanding of the Bungeni Parliamentary System information flowing.

Giving the example of a citizen that browses the system to have information about a “Motion” and wants a digital copy of the document, an explanation about the data flow through the different components of the system, from the browser to the final information, is provided.

A regular citizen opens the browser and types in the URL of his local Bungeni Parliamentary System. The browser communicates with the Web Server of the Bungeni Parliamentary System (WSGI) which will ask for a “Theme” for the Website in BungeniCMS. The calls are made via URL and for BungeniCMS its application is called as are the “Static Resources” to theme the content that comes from BungeniCMS. When all the rules are applied in the “Theme Delivering Module” the Website appears on the browser, and the user may start his research.

After that, the citizen wants to have information about the agenda of a specific day. Using the graphical interface, the citizen selects the “Business” option and the browser sends the request using the previously reffered mechanism of “Theme Delivering”, but this time the URL’s contain another destination, the “BungeniPortal”. The “BungeniPortal”, will send a SQL Query to the “Bungeni DB”, to find out which events are happening on the chosen date. Data is delivered from “Bungeni DB” to the “BungeniPortal”, which is then converted to HTML and themed until it arrives again on the browser.

After finding out what is being discussed on that specific date, the citizen wants to download a digital copy of a document that is scheduled for discussion. The user accesses the information provided in the last flow, and this time requests a download operation on “BungeniPortal”. The data coming to download is stored on “Exist”, so after the “Dispatcher” points to the “BungeniPortal” the later will send an XML Exist query to the “Exist” module and waits for the data retrieval. When the data is retrieved, inside “BungeniPortal” an operation to transform the data into a PDF file is executed, and after completion the file is sent to the browser, for citizen viewing.

3.3.7 System Deployment

Like stated before on the document, the deployment of the components is made via supervisord, either using the graphical interface or the Fabric commands on a Terminal. The starter supervisord configuration lies on the file supervisord.conf, a file with structure similar to Microsoft Windows INI files. Using this file one can see what deployment is gonna be done in therms of Bungeni Parliamentary System components, like the Figure 3.10 shows.

The main components to be deployed are:

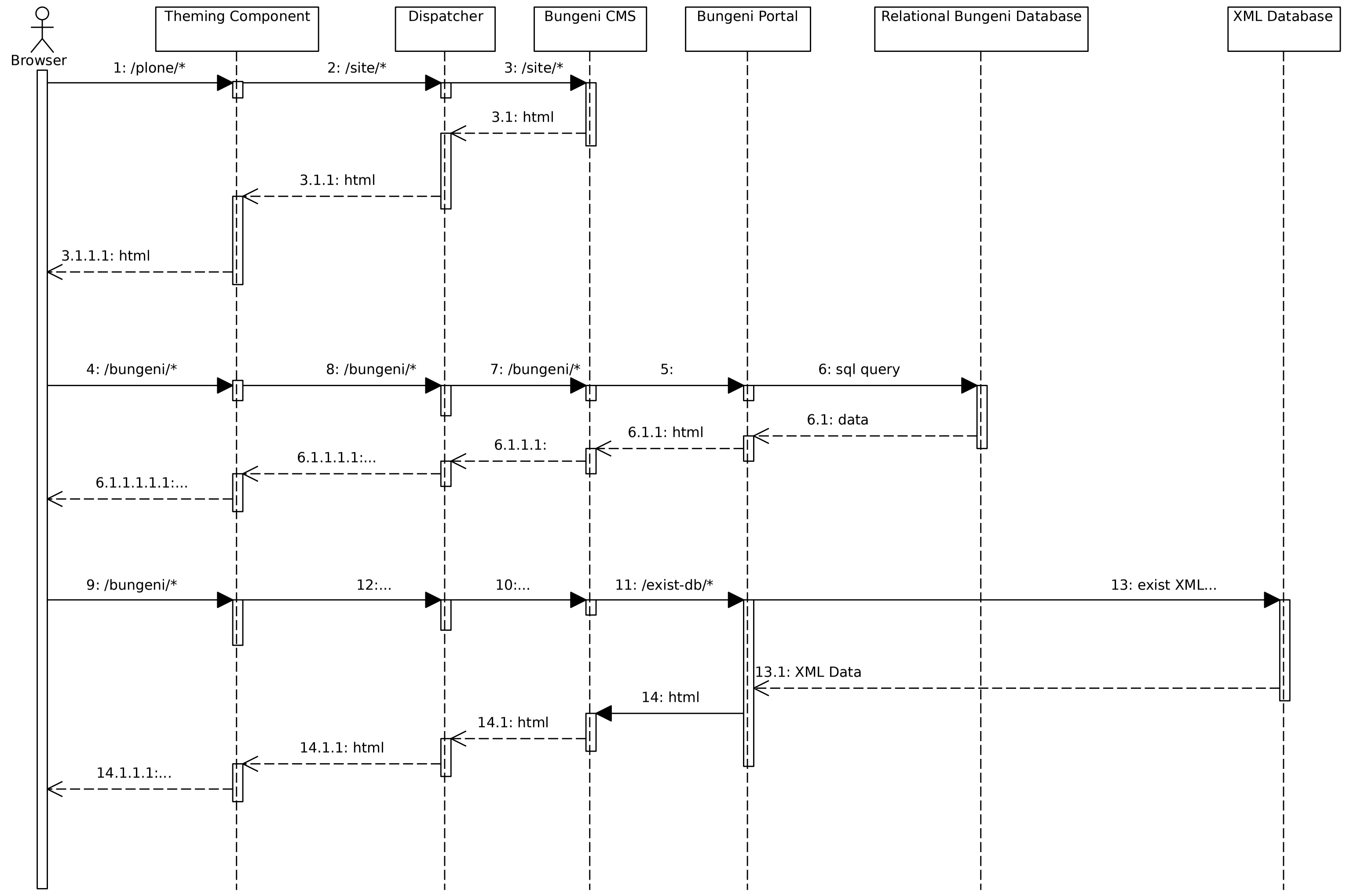


Figure 3.9: Data Flowing In Bungeni Parliamentary System

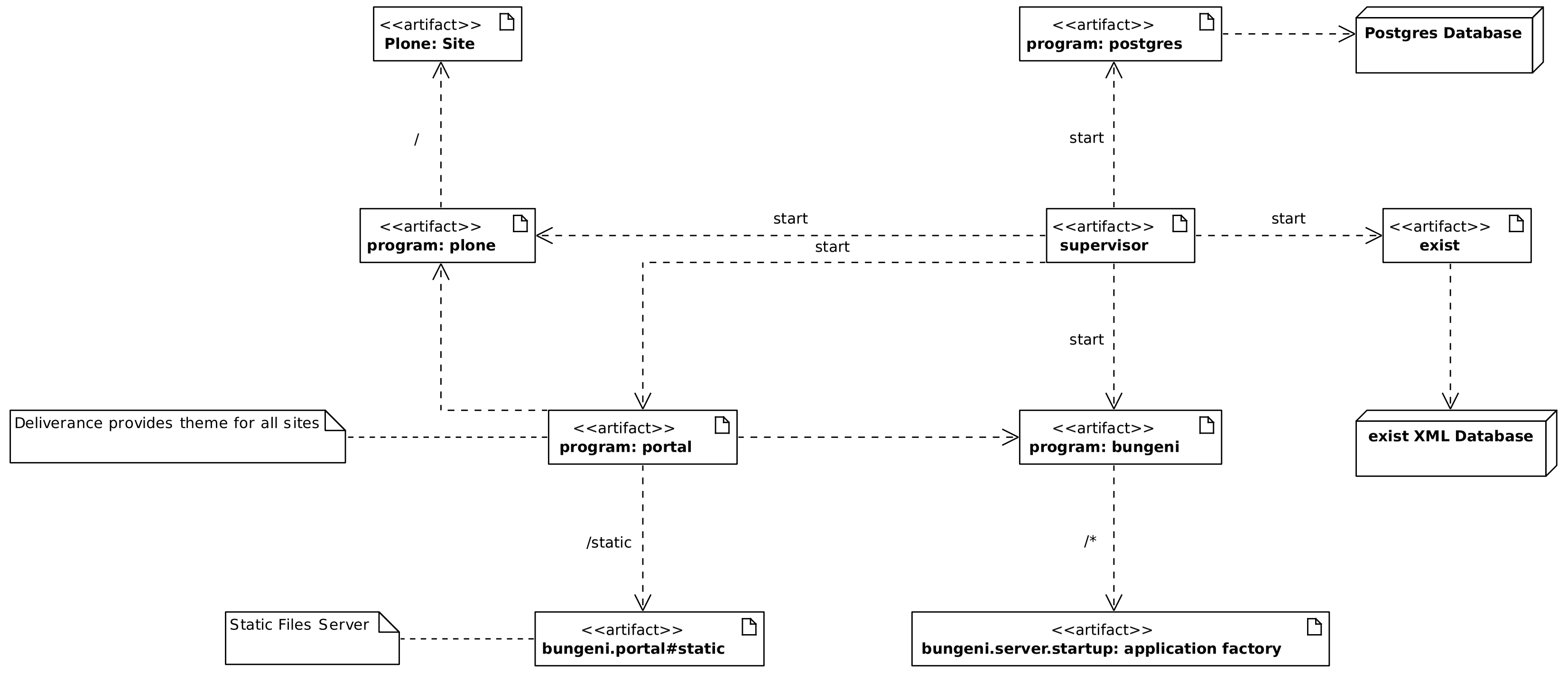


Figure 3.10: System Components Deployment in the Bungeni Parliamentary System

- program:portal;
- program:plone;
- program:bungeni;
- program:postgres;
- program:openoffice.

Some other components were briefly stated previously on the document and include:

- program:exist;
- program:exist-sync;
- program:rabbitmq.

One final component that wasn't discussed is varnish [51], which though it helps the Bungeni Parliamentary System to have a better performance it is not necessary for the good behavior of the system. Varnish is a library to implement a cache mechanism on the system. It speeds up query retrieval by previously indexing results to recent accessed content and also by elevated number of requests to an object.

Paster Deployment

Applications are reachable through Paster, that implements the "Dispatcher" module as previously stated in 3.3.2. Paster serves applications based on the WSGI interface using the HTTP protocol.

Configuration files are spread in the system, and some have a higher complexity, by interacting with not only the component being deployed, but also with other components that relate to it.

For instance, the configuration file for program "portal" has a "main" section that defines a pipeline which filters requests through deliverance (3.3.2) and serves a great amount of specific application related URL's. This configuration file despite relating mainly to "program:portal" also makes changes on the deployment of the "Dispatcher", due to URL's mapping, and also "Theme Delivering".

By default in the configuration files, the URL's used by Paster to call the applications are defined as in the list below. Because paster works over HTTP and replies to requests using WSGI, this means that the same interfaces can be accessed by a regular browser and are not fully protected.

List of application URL's:

- / = plone;
- /plone = plone;
- /bungeni = bungeni;
- /members = members;

- /business = business;
- /business/whats-on = whatson;
- /archive = archive;
- /calendar = calendar;
- /admin = admin;
- /workspace_archive = workspace_archive;
- /static = static.

program:bungeni

Bungeni Portal is accessible through paster with the deploy.ini configuration file. The mapping is made using two URL's, '/' and '/cache'. The first is the real access to the "Bungeni Portal" on '/', a two way connector to a repoze.who [52] based middleware for authentication and the server execution application. The server application uses ore.wsgiapp.

Ore.wsgiapp according to the documentation provided on the development site "allows one to bootstrap a zope3 environment as a wsgi application without a ZODB and site.zcml is the zope 3 instance configuration". This means in a simple way that ore.wsgiapp will provide a place for indexing the application headers mapped on URL's so that they can be accessed via Paster.

Configuration for the access configuration and mapping are done via Zope Configuration Markup Language files (ZCML).

program:plone

As stated on 3.3.4, BungeniCMS is based on Plone and it is accessed through paster by means of a deployment file. WSGI works by dispatching requests in the Bungeni Parliamentary System, part of which are replied to Plone. Plone "per se" is not capable of replying web requests, so attached to it there is a ZOPE instance. The BungeniCMS is the "site" instance of Plone in the root of Zope.

To clarify, ZOPE (Z Object Publishing Environment) is an application server and Plone is a Content Management System which is built on top of Zope that on its way relies on WSGI to feed requests.

3.3.8 Security and Access Control

Technologies Used

repoze.who

repoze.who is an identification and authentication framework for arbitrary WSGI applications. repoze.who can be configured either as WSGI middleware or as an API for use by an application. repoze.who is inspired by Zope2's Pluggable Authentication Service

(PAS) (but repoze.who is not dependent on Zope in any way; it is useful for any WSGI application). It provides no facility for authorization (ensuring whether a user can or cannot perform the operation implied by the request). This is considered to be the domain of the WSGI application [46].

OAuth Standard

The OAuth authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. In a way it simulates the Single Sign On Principle.

Access Control - Roles

Roles are virtual titles given to users on Information Systems. The access control is controlled by the use of these virtual titles, and its rules define what users have access to which object or resource. Roles are a metaphor to what happens in real life with the granted titles. For example, if a citizen as the title of janitor in the bank, it is only normal that he gets reported to security if he tries to be inside the banks safe. On the other hand, if a citizen has the title of bank manager the security will most probably not be alarmed.

This metaphor is used throughout Internet and local applications has a mean to restrict and grant correct access control.

In the Bungeni Parliamentary System almost everything in flexible and can be created or modified. Access control is no exception. The Default System is already packed with a few roles. To state a few of the most important:

- Administrator;
- Clerk;
- Speaker;
- Member of Parliament;
- Member of Government.

The roles relate how the system works and how it is presented. For example, a user with the Administrator role, after a successful login is redirected to a completely different graphical interface from users that have roles such as Member of Parliament or Clerk. This is access control in action on the system.

Other type of access control can happen on the access to workflows (3.3.5). Despite sharing the same graphical interface (workspace) upon successful login, both Clerk and Member of Parliament roles can have different access control when it regards the behavior of workflows.

As stated before, workflows have configuration files in which access control can be managed through facets and allows. This way, for example, a Member of Parliament may be only able to view a workflow that a Clerk has permissions to edit or vice versa, depending on the workflow configurations.

Roles creating is flexible and dynamic, but is done on the file system, by adding Roles and the direct description that will be visible on the graphical interface. On the graphical interface an Administrator user can only associate already created users with already created Roles profiles. An Administrator user has not the ability to create new Roles on the graphical interface. When a new role is inserted into the system it needs to be restarted in order to be able to use it, otherwise it will not appear and thus is as it never existed.

3.4 Customizations Folder

The customizations folder, built in the system, is the core of dynamic customizations on the Bungeni Parliamentary System. It is the location where all the files and templates to accommodate different realities of countries are represented. It is the folder where configuration files to document workflows, roles, forms and layout themes are stored.

Thanks to this only folder the system can be deployed, in the National Assemblies of all countries, using the same deployment steps (which now consist in using Debian based packages). After the installation a country's reality can be modelled according to workflows and roles that apply locally in a dynamic fashion. This eases, for instance, migrating through different servers.

In a scenario where a server develops connection problems, this folder can be copied and used on a new deployed system, without having to make internal system configurations. This also means that a country's National Assembly reality needs to be only modelled once.

Chapter 4

Documents Workflow Edition

4.1 Introduction

On the course of over one year, since the entrance on the project and starting the development of the work, the Bungeni Parliamentary System suffered an enormous process of changes. On a daily basis, implementation would change sometimes hourly. Though the base of the system was already built, there were minor problems to fix and be thoroughly tested. Often communication with the development team was done to address some troubles that were not expected at first. The workflow management system was one of the troublesome parts of the system, because when it was built it was meant to have just a set of static documents. After some iterations of testing and report to the developing team, a few fixes to the code were provided to them and also some locations where the code had problems, so that the pinpoint and resolution could be more effective and efficient.

In the middle of February, beginning of March of 2013, the system changed back to its roots to accommodate, countries that have two types of chambers (Lower House and Higher House). Once again, participation as one active tester was performed, providing feedback, problem locations and also fixes. Though the implementation was not changed directly during the work, some installation problems were detected and solved during it, as well as some advice on the substitution of a python package server. On the system repository there is also documentation relative to the process of workflows and legislation mapping, that was done with the supervision of the Professors involved.

To summarize, given the facts, contribution to parts of the system was provided and as well as a "first costumer" feedback, to the developers team.

In this chapter the development of a tool, to help increase productivity on the production of workflows is presented. Other content related to the developed work is also explained.

Since the configurations are written in XML files, a prototype idea to create a user friendly tool that could create the configurations without typing errors and wrong logic emerged. The need to create such a tool appeared and is explained in the next section of the Document, using a real World workflow, allowing the demonstrate the complex process behind the production of this type of content.

4.2 A Workflow Creation

Part of the Bungeni project that involved the University of Aveiro, was to modelate real country legislation workflows on the Bungeni Parliamentary System. One of the countries that showed great interest in this platform was Sao Tome and Principe. Sao Tome, provided us with the data needed to be able to understand the workflow process behind a great majority of Parliamentary documents. Based on the data provided, and consulting official documents such as the Constitution [53], and also in the regiment of the National Assembly [54] the team was able to start the creation of the workflow for the bill project - Projeto de Lei (PDL).

The process of producing the workflow was not direct and had some iterations between the Aveiro team and the "Assembleia da República". Aveiro provided the knowledge of the Bungeni System and the "Assembleia" shared the legislation knowledge in what concerns the steps that a PDL can undergo before completion.

Often iterations lasted for weeks. This was due to the fact that workflows are mapped by using XML files, following a schema. On the time the team started developing the workflow, there was no available tool to speed up the process. Just the use of Kate text editor which can help by storing previously used words on cache, making the writing less prone to error. Everything else was written by hand, and took several hours.

The team started the process using the "bill" workflow (embedded in the Bungeni System), that is intended to be used as an example or starting point to develop custom workflows. By using a tool developed by the development team and enhanced by me, results of the workflow were provided on PNG (Portable Network Graphics) files.

The task list to create or modify a workflow was divided in the following steps:

1. understand and map the states and transitions on paper
2. write the XML configurations
3. compile the result, if the compiling was not successful return to point 2 was obligatory
4. visualize the flow of information provided on the PNG files, could lead to point 2
5. analyse legislation and workflow accordance, could lead to point 1 or 2
6. test the workflow on the system
7. present the final result to the "Assembleia da República", and wait for feedback. If there were alterations to be done, start at point 1 again

Following this process and with several alterations along the creation of the workflow, many hours were spent between points 2 and 4, and later also 5. The team took almost two months to develop this workflow that had 91 states, opposed to the the original bill which had

35 states. More than often, states needed to be decomposed in smaller states after received feedback to augment the granularity of the workflow. These alterations usually meant that states needed to be disconnected in the middle of the workflow and connected again to new states, before every change was complete. This meant skipping from state to state manually, erasing configurations, creating new ones always keeping track on what the final result should look like.

It is in this context that the idea of a tool that could easily automate the mapping of a document workflow, to the Bungeni system, emerged. This tool would allow ease on the writing of the XML files used in the configurations, by the system.

4.3 Developed Tool Overview

The developed tool or workflow editor, intends to increase productivity on the creation and management of workflows inside the Bungeni Parliamentary System. As seen before the creation of a workflow is a task prone to errors and so, a higher degree of automation of code could reduce this fact. To part of the Bungeni Parliamentary System, this tool should be developed using only open source software mantaining the philosophy of the system, which is completely developed under open source licensing.

The workflow editor should address a set of functionalities described further in the document, but on a general rule, the existence of such a tool should save a great number of hours spent between modelling a workflow and testing it. This is achieved by automation of the XML code that it will produce, which is already in concordance with the XML Schemas implemented on the system, minimizing writing and syntax errors. The edition will be done in a graphical interface with minimum keyboard input.

4.3.1 Server Side Overview

The server side of the application is where all the logic and functionalities are implemented. The workflow editor was written using the approach of a “thin client”, relegating all the processing of information to this part of the tool. Thus making the server busy, it is a better approach in this case, because it saves every change produced on the workflow on the server and not on javascript variables which have the tendency to be more volatile. This part of the tool was developed in python.

4.3.2 Client Side Overview

The client side of the application, can be seen as the graphical interface (provided in the form of a web application), with which the user or actors will interact. It has minimal logic implemented, regarding only graphical information.

4.4 Requirements Analysis

4.4.1 Stakeholders

Stakeholders of the developed tool can be seen as all the countries that are interested in implementing the Bungeni Parliamentary System in their National Assemblies, namely the African Countries. The collection of stakeholders can be seen in a broad way as all the National Assemblies clerks that interact with the Bungeni System, because the tool will allow to speed the modelling of workflows. In a specific way, the actors that model the workflows are the direct stakeholders on the development of this tool.

4.4.2 Architectural Requirements

The workflow editor, should be developed in a modular form, being independent from the code of the Bungeni Parliamentary System. The Bungeni Parliamentary System, should not have dependencies on the implementation of the workflow editor. Both tools should exist in a "symbiotic" relationship, not depending from each other, but taking advantage of the functionalities provided from one another. The modular approach will help in the embedding of the workflow editor in a debian package.

4.4.3 Functional Requirements

The workflow editor needs to address a collection of functionalities to be able to be considered a valuable tool:

- it should provide access to previously created system workflows for edition only purposes;
- the tool should allow creation and edition of properties or constraints that provide access control;
- a document having a scheme and properties of a document, should be created and provided in PDF form.

Use Cases

Actors
Only one actor was identified to describe the possible uses cases of the workflow editor. It will be named User henceforward. The User is the person whose task is to create a new document workflow using the tool, or also import and modify already existent workflows.

The Figure 4.1 represents the Use Cases of the workflow Editor.

Import Workflow

Allows the user to import an already created workflow from the Bungeni Parliamentary

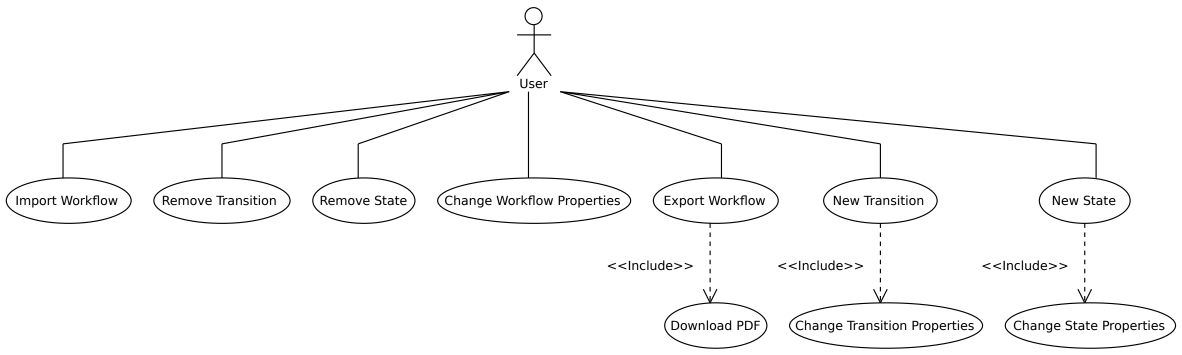


Figure 4.1: Workflow Editor: Use Cases Diagram

System, in order to use it as a starting point to develop another or just to edit the properties of it.

Actor: User

Workflow:

1. The user clicks the button assigned to that functionality
2. The user is prompted with a dialog asking the name of the workflow which should be imported
3. After introduction the graphical interface communicates with the backend application asking for the workflow
4. The workflow is loaded from XML, parsed to JSON and sent to the graphical interface
5. The JSON object is parsed and a diagram of the workflow is created as well as the collection of its properties in the user interface

Expected Results:

- Fully formed information retrieved from a previously created workflow
- In case of the BackEnd not finding the workflow, the interface produces an error

New State

Allows the user to create a new state on the workflow, as well as editing the properties and access control of such state.

Actor: User

Workflow:

1. The user clicks the button assigned to that functionality
2. The user is prompted with a dialog asking the name of the state that should be created, alongside with its properties

3. Inside the logic of the interface the state is added to the workflow diagram and controls to access its properties are created

Expected Results:

- New workflow state produced

New Transition

Allows the user to create a new transition between states on the workflow, as well as editing the properties and access control of such transition.

Actor: User

Workflow:

1. The user clicks the button assigned to that functionality
2. The user is prompted with a dialog asking the name of the transition that should be created, alongside with its properties
3. The user clicks the states (first the source state and secondly the destination state)
4. Inside the logic of the interface the transition is added to the workflow diagram, which is updated, and controls to access its properties are created

Expected Results:

- New workflow transition produced

Remove Transition

Allows the user to remove a transition between states on the workflow, as well as deleting the properties and access control of such transition.

Actor: User

Workflow:

1. The user clicks a state which, is connected to the transition which he wants to delete, on the list of states
2. The list of states connected to that state appears
3. The user clicks on the button that deletes the transition
4. Inside the logic of the interface the transition is deleted in the workflow diagram, which is updated, and controls to access its properties are also deleted

Expected Results:

- Workflow transition deleted, on the diagram, as well as controls associated with it

Remove State

Allows the user to remove a state from the workflow, as well as deleting the properties and access control of such state.

Actor: User

Workflow:

1. The user clicks the button associated with that functionality
2. The user clicks the state to be removed on the diagram
3. Inside the logic of the interface the state is deleted in the workflow diagram, which is updated, and controls to access its properties are also deleted

Expected Results:

- Workflow state deleted, on the diagram, as well as controls associated with it

Export Workflow

Allows the user to export the workflow to the backend or server side, in order to produce the XML configurations of a workflow according to the Bungeni Parliamentary System specifications.

Actor: User

Workflow:

1. After being satisfied with a workflow, the user clicks on the button assigned to that functionality
2. The user is prompted with a dialog box, asking the name for which the created workflow will be referred to
3. The graphical interface communicates with the backend sending the information needed to produce the workflow in JSON format
4. The backend produces the XML files needed to implement the workflow on the Bungeni Parliamentary System.
5. The backend produces and retrieves a link from which the user can access the PDF file containing the documentation

Expected Results:

- Retrieving of a PDF file with the documentation of the workflow
- Creation of the XML files needed to implement a workflow in the Bungeni Parliamentary System
- If the communication with the backend fails, or an error occurs in the processing of the workflow the interface produces an error

4.4.4 Non-functional Requirements

Non-functional requirements are types of requirements that don't have a direct reflection on the tasks or functionalities that the tool can provide. A list of these types of requirements, follows:

- the tool should be developed in the programmatic language on which the system was developed, for future integration, being it python;
- it should have a user interface, created as a web based application;
- the tool should provide, on real time, a scheme or diagram, presenting the states and corresponding transitions of the being edited workflow.

4.4.5 Design Requirements

The use of python language is justifiable by being easier to incorporate and embedded this tool on the system using the WSGI standard provided by the dispatcher service. To be also platform independent, the tool should have a graphical interface provided in the form of a web application.

4.5 Architecture

From the beginning of the idea and formalization of the concept to develop a tool to help and support the creation of a workflow in an automated form, it was intended to be supplied as part of the Bungeni Parliamentary System. The tool should be incorporated on the system using the WSGI standar provided by the dispatcher service.

The tool can be broken down in two major blocks:

1. a BackEnd application:
    - developed in python;
2. a FrontEnd or Web application:
    - developed using simple web frameworks based on javascript.

Communication between the two components is made by the use of web services specially created to implement the features of the tool. These web services have a REST like interface, provided by Bottle and WSGI. Figure 4.2.

4.5.1 Flow of Information

The flow of information between FrontEnd and BackEnd follows the same pattern along all the interactions with one another, and can be seen in Figure 4.3. As seen on the figure,

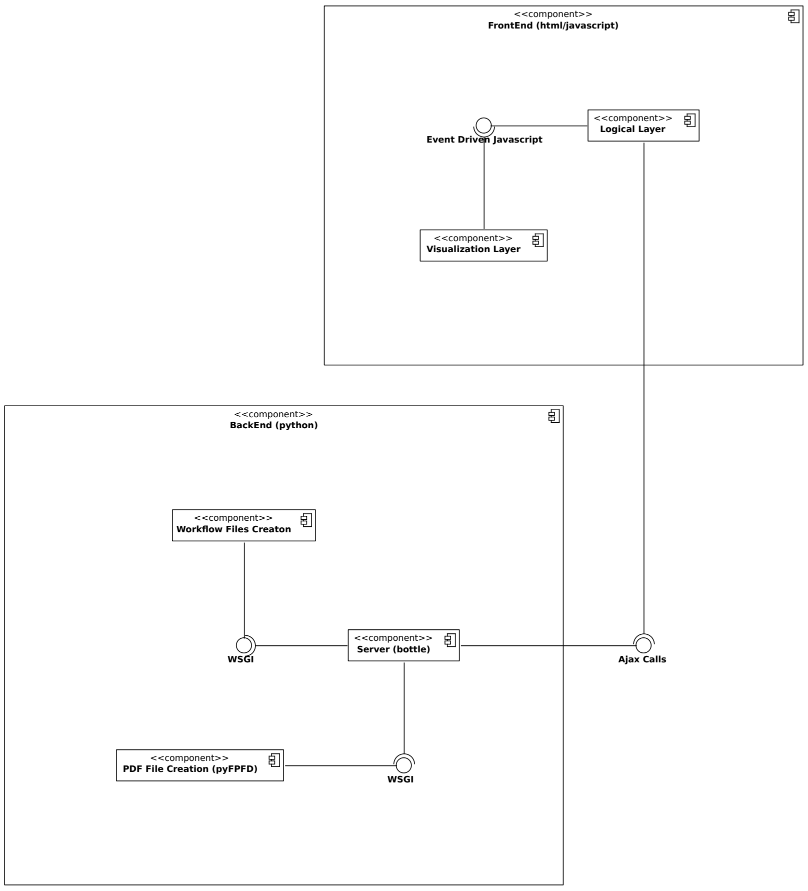


Figure 4.2: Components Diagram of the Workflow Development Tool

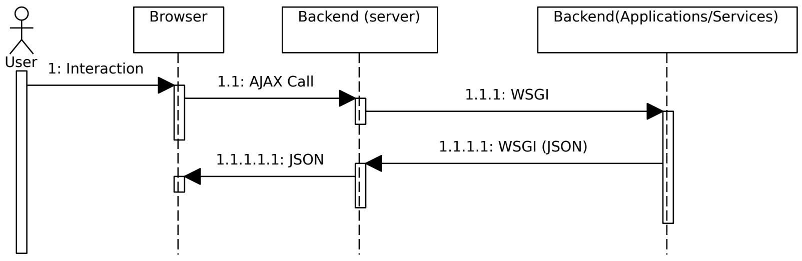
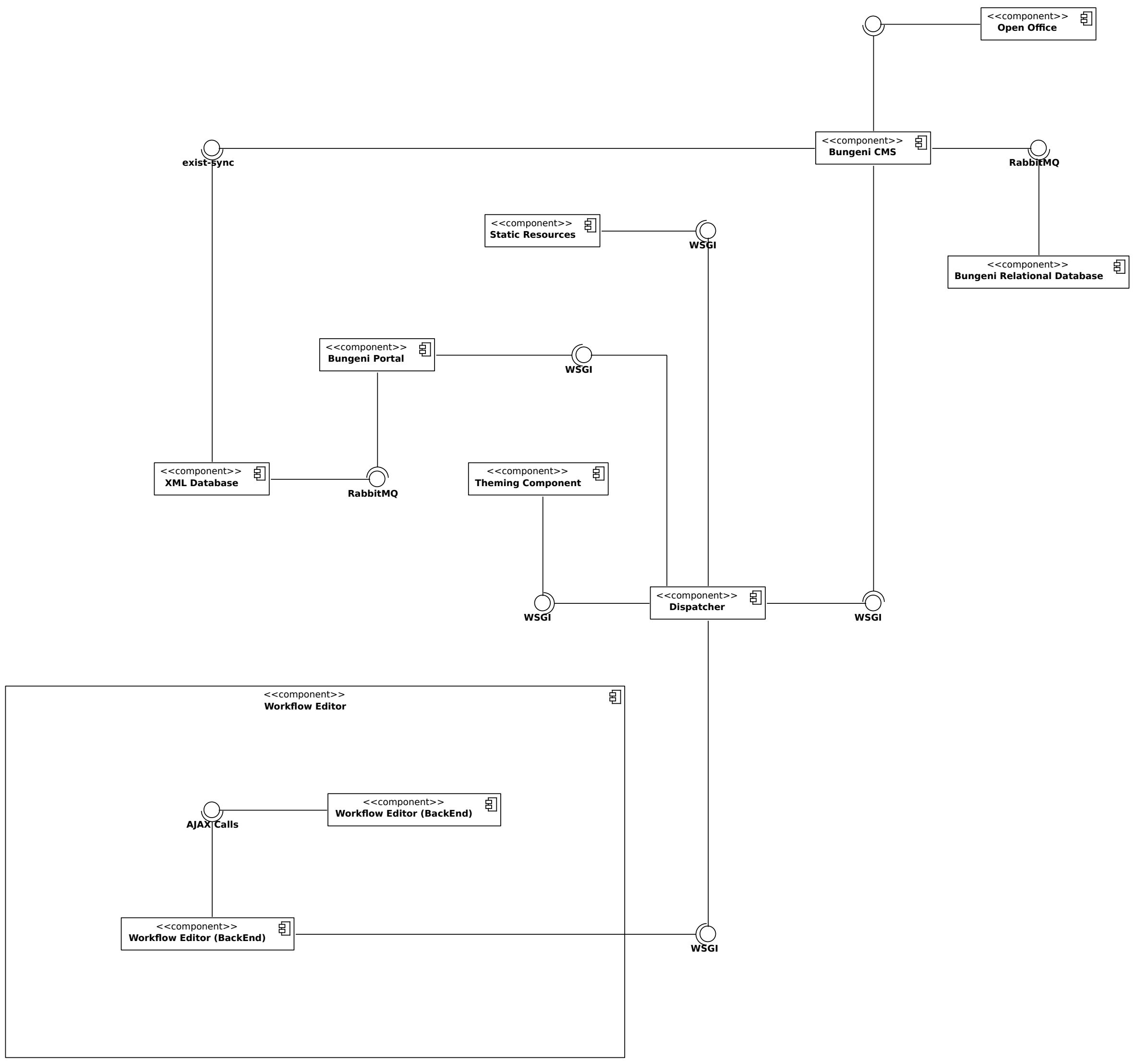


Figure 4.3: Flow of Information Between the Workflow Editor Components



<<component>> Open Office
<<component>> Bungeni CMS
exist-sync
RabbitMQ
<<component>> Static Resources
WSGI
<<component>> Bungeni Relational Database
<<component>> Bungeni Portal
WSGI
<<component>> XML Database
RabbitMQ
<<component>> Theming Component
WSGI
<<component>> Dispatcher
WSGI
<<component>> Workflow Editor
AJAX Calls
<<component>> Workflow Editor (BackEnd)
<<component>> Workflow Editor (BackEnd)
WSGI

Figure 4.4: Workflow Editor Integration in the Bungeni Parliamentary System

every connection from the user interface (Brontend) is made using AJAX calls and every return from the BacKEnd to the interface is done using JSON objects. This happens because the bottlePy API for python provides ease on the exchange of information in this type of serialized object.

4.5.2 Deployment

The workflow editor will be integrated with the Bungeni Parliamentary System, following the principle already applied in other services that are present in the system. To provide an ease on the perception, the Figure 4.4 presents the components of the workflow editor embedded on the Bungeni Parliamentary System.

It is easy to assume from the figure that, the BackEnd will only be accessible after being activated by the dispatcher service.

4.6 Development

4.6.1 Technologies Overview

Over the course of the development of the tool, the used technologies suffered a trial before usage. Although not all the technologies are used on the final product, it makes sense to name them on the cause that they provided points on understanding difficulties and features that the tool needed to address.

Twitter Bootstrap [55]

Described as "Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development." the twitter bootstrap is a collection of pre-made sets of objects in HTML5 and their corresponding applicable styles defined on CSS (Cascading Style Sheet) files. It is fully responsive.

jsPlumb [56]

jsPlumb provides means for a developer to visually connect elements on their web pages. It uses SVG or Canvas in modern browsers, and VML on IE 8 and below.

Liviz.js [57]

Liviz.js is a DHTML based "dot" tool of GraphViz (Graph Visualization Software [58]. It was previously named JSVIZ, and basically tries to implement GraphViz visualization of objects on browser. The objects are described in "dot" language and presented as SVG (Scalable Vector Graphics).

jQuery [59]

Query is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

raphael-pan-zoom [60]

This plugin enables pan and zoom functionality to a Raphael [61] paper. It works by applying linear transformations to the SVG objects, creating the illusion that pan and zoom are real.

pydot [62]

Python interface to Graphviz's [58] "dot". This module provides a full interface to create handle modify and process graphs in Graphviz's dot language. Because pydot is only an interface, Graphviz must be installed on the host system for it to succeed.

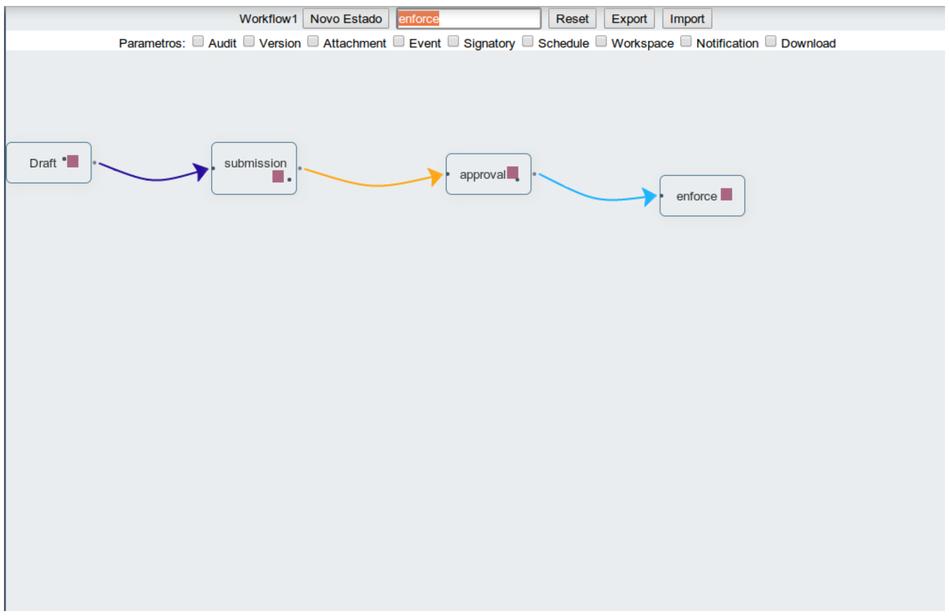


Figure 4.5: First Prototype: Small Workflow Example

Bottle [63]

Bottle is a fast, simple and lightweight WSGI micro web-framework for Python. It is distributed as a single file module and has no dependencies other than the Python Standard Library. This makes it easier to understand and also lightweight on the operations. Provides convenient access to form data, file uploads, cookies, headers and other HTTP-related metadata. Makes routing possible by using the WSGI standard.

PyFPDF [64]

PyFPDF is a library for PDF document generation under Python. Its main features are the code simplicity to produce quality PDF files, and also provide the possibility to configure the styling output of the files via HTML.

4.6.2 First Prototype

A prototype started being developed using a "backend" programmed on Python so that it could be later introduced as part as the Bungeni Parliamentary System. The "frontend" or graphical interface, used jsPlumb to display the objects on a SVG canvas within a browser. Interaction and location of objects was done using drag and drop and the design was intuitive. To use with workflows that had just a few states this implementation would be sufficient, Figure 4.5.

The big problem arose later. When trying to escalate to more complex workflows (such as the "motion" workflow provided on the Bungeni Parliamentary System). This approach became unusable, because the the jsPlumb framework did not have an "auto layout" method to

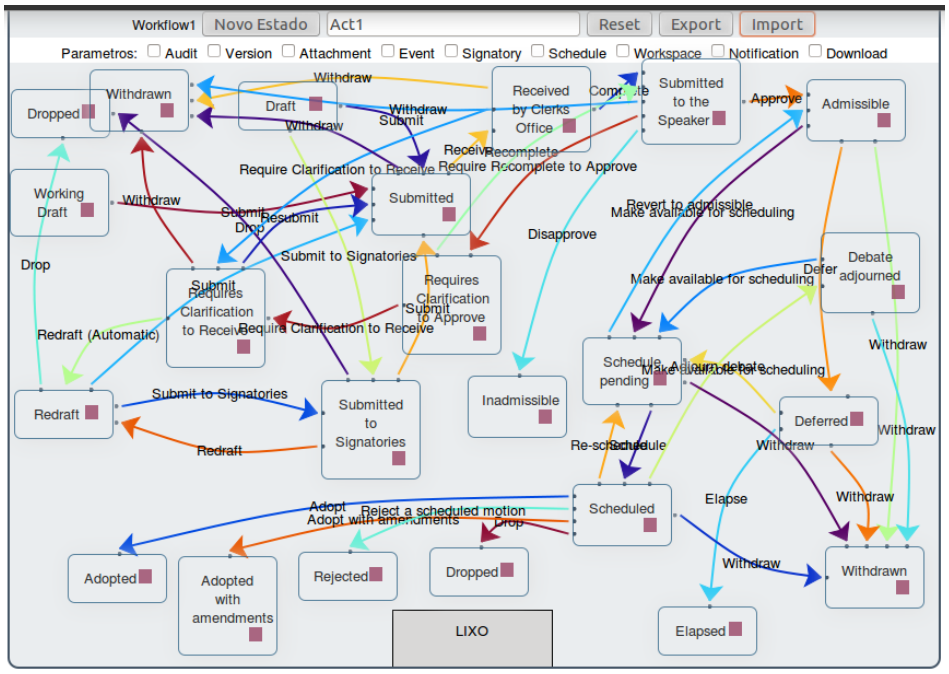


Figure 4.6: First Prototype: Complex Workflow Example

balance the position of the states and optimize the paths made by the arrows of the transitions to not cross other objects. The tool became inoperable on these conditions, Figure 4.6.

In order to solve this problem a web service was developed to calculate the optimal positions of the states, based on orthogonal layouts [65]. The web service would receive the list of states and transitions and use a interface to Graphviz to acquire optimal location coordinates, returning them to the browser so that the objects could be drawn on the canvas. Although it made a great increase on the perception of the drawn workflow it still did not satisfy the minimum requisites of visualization, because the arrows would go across the states, Figure 4.7.

The failure of the first attempt developed the need to think in therms of the graph itself being drawn and not only the usability. For experience, Graphviz had supplied quality graphs with good perception by calculating the optimal positions of the stages and the bending on the transition arrows. This bending was not allowed in jsPlumb, so the prototype was abandoned to create a new one.

4.6.3 Second Prototype

The second prototype was developed focusing on a tool that was able to emulate the results of Graphviz on a browser canvas. Liviz-js was used to achieve this behaviour successfully. Although the canvas was also a SVG object on this framework, the objects on its content were static. There was the option to make them draggable like in the first prototype, but

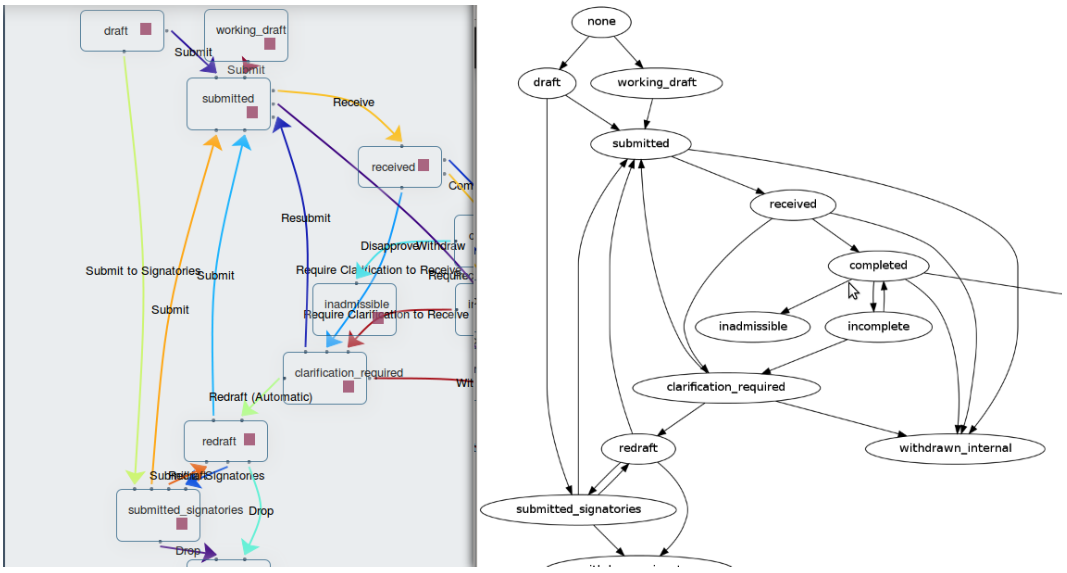


Figure 4.7: First Prototype: After Coordinates Optimization

after studying the consequences a conclusion that it would rise complexity problems, and even difficult the usability of the tool, was obvious. Instead, a new interface was studied. This new graphical approach would be almost completely point and click, opposed to the drag and drop approach of the first prototype.

The upper menu would have buttons to inform the tool of the operation being handled at the moment. The left menu would be an accordion with the information of the transitions mapped on the canvas, and the rest of the screen would be the canvas to work. The canvas would be interactive, making for example the creation of a new transition as simple as clicking two or more states.

The new approach was tested, and once again as similar to the first prototype once the workflows produced got more complex the tool became unusable. The undesired behaviour was being forced by our solution for the problems in the first prototype, the introduction of Liviz-js. Liviz-js, being a pure Graphviz interpretor receives the data (provided in "dot" format) and builds the graphs with absolute coordinates, before returning them to the canvas for displaying. In more complex workflows their corresponding graphs would not fit the screen, and would only be accessible using the native browser zoom. There was a need for a library that allowed the use of pan and zoom in SVG objects. This need was filled using the raphael-pan-zoom library that allows to zoom and pan in raphaelis [61] SVG objects. After the conversion of the Liviz canvas to a raphaelis SVG object the tool was scalable to any type of workflows not depending on their complexity.

4.6.4 Final Product

The final product, was developed based on the second prototype. A general overview can be seen in Figure 4.8. It kept the left menu in which the states of the workflow are located. Upon click, the menu, which is in fact an accordion, opens its node and reveals the connections to which the selected state is linked, or has transitions with.

On the right side of the screen a graph is shown representing each step and transition of the modelled workflow. It is possible to see that when there is the existence of a great quantity of states and transitions, the tool becomes unusable. To treat this problem a mouse scroll wheel, and top controls signalized by plus and minus characters, allow a direct control of the zoom feature.

The buttons on the top row of the page control the edition methods.

The button "Propriedades" (*Properties*) allow the user to insert general properties of the workflow, like the title of the document, or the system roles that have a access to the workflow. It also controls the workflow features, which are actions that can be done with a document of the type of the workflow being created.

The button "Novo Estado" (*New State*), creates a new state of the workflow.

The button "Nova Ligação" (*New Connection*), allows to create a new workflow transition of state, by clicking in to states on the graph, to create a visual connection.

The button "Importação" (*Import*), allows the user to import directly a workflow from the Bungeni System, defined in xml format. This feature also imports states and connection properties.

The button "Exportar Documento" (*Export Document*), sends the drawn workflow to the python backend, via JSon object and ajax calls. Once on the backend it produces a PDF sent to the user and stores an XML file with the configuration of a Bungeni workflow.

The button "Remove Estado" (*Removes State*), waits for the user to click in a state and then erases it from the graph or workflow representation.

The button "Reset" (*Reset*), as the name implies, resets the drawn of the workflow, back to a white canvas.

On the bottom of the drawing canvas, there are two buttons. These buttons allow for history tracking.

The button "Desfazer" (*Undo*) allow the user to cancel the previous action performed on the drawing, whilst the button "Refazer" (*Redo*) allow the user to do the contrary. This contrary in this case is to redo a previously cancelled drawing action.

As is possible to see in Figure 4.9, the zoom feature was used successfully to access only the part of the workflow that was relevant at the moment. Due to the existence of nearby states, that can get the user lost, the state highlights in a yellow color if its corresponding node on the accordion is hovered. This feature allows for better human eye location when states are located to nearby of each other.

Regarding the connections or transitions of state, the same help is provided through the accordion. When a node is clicked, the list of connected states to that node is displayed. Upon the hovering of "Opções", of a state that is connected to the opened node, the transition highlights on the graph and the corresponding label, in text form, is augmented. This behaviour can be seen on Figure 4.10. This hovering element also allows the user to access the edition of the connection properties.

Producing the XML file and visual aid, through a PDF file, both displayed in the appendix, this tool proves to be time saving in the production of Bungeni workflows.

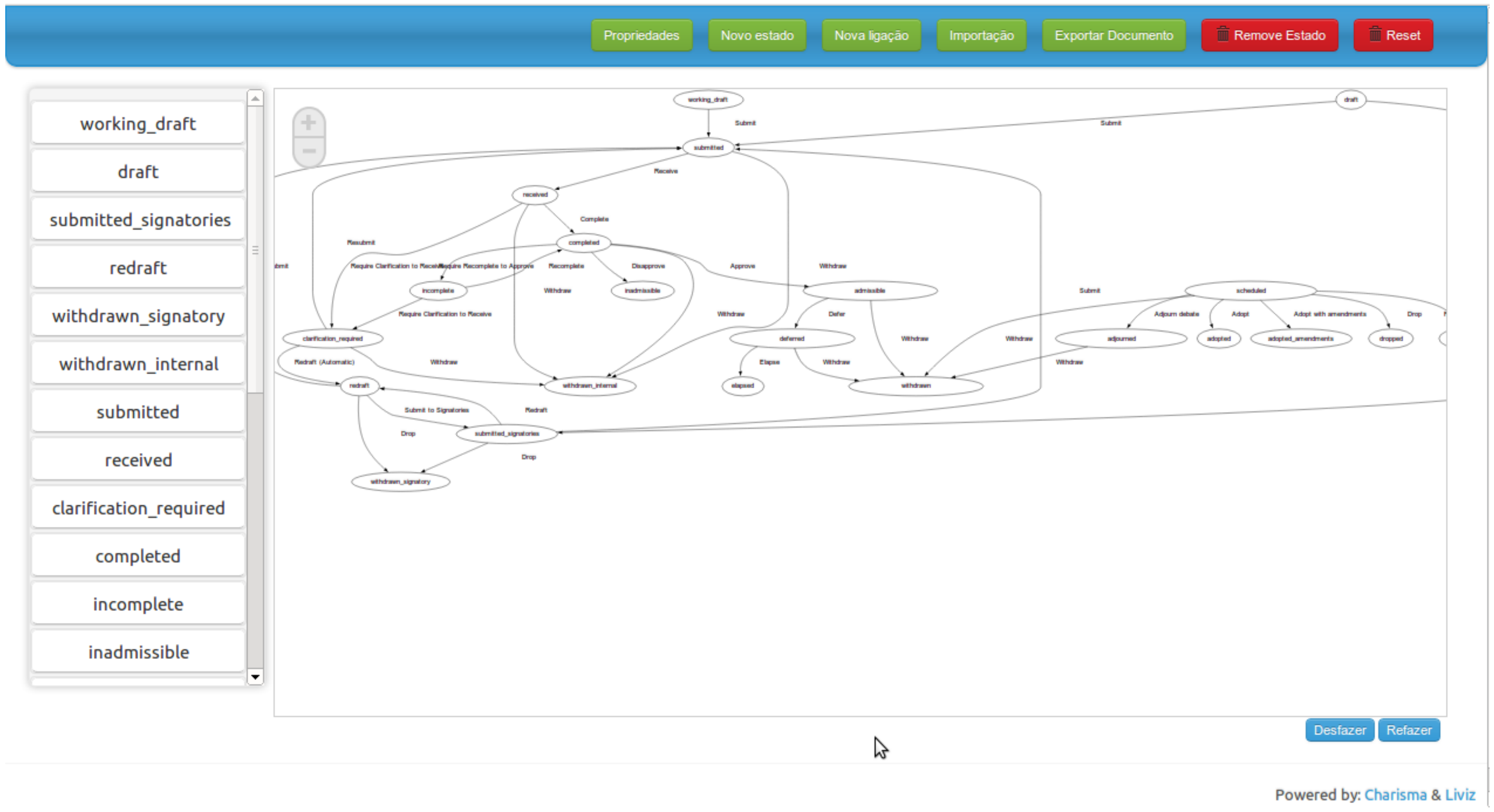


Figure 4.8: Workflow Editor General Overview

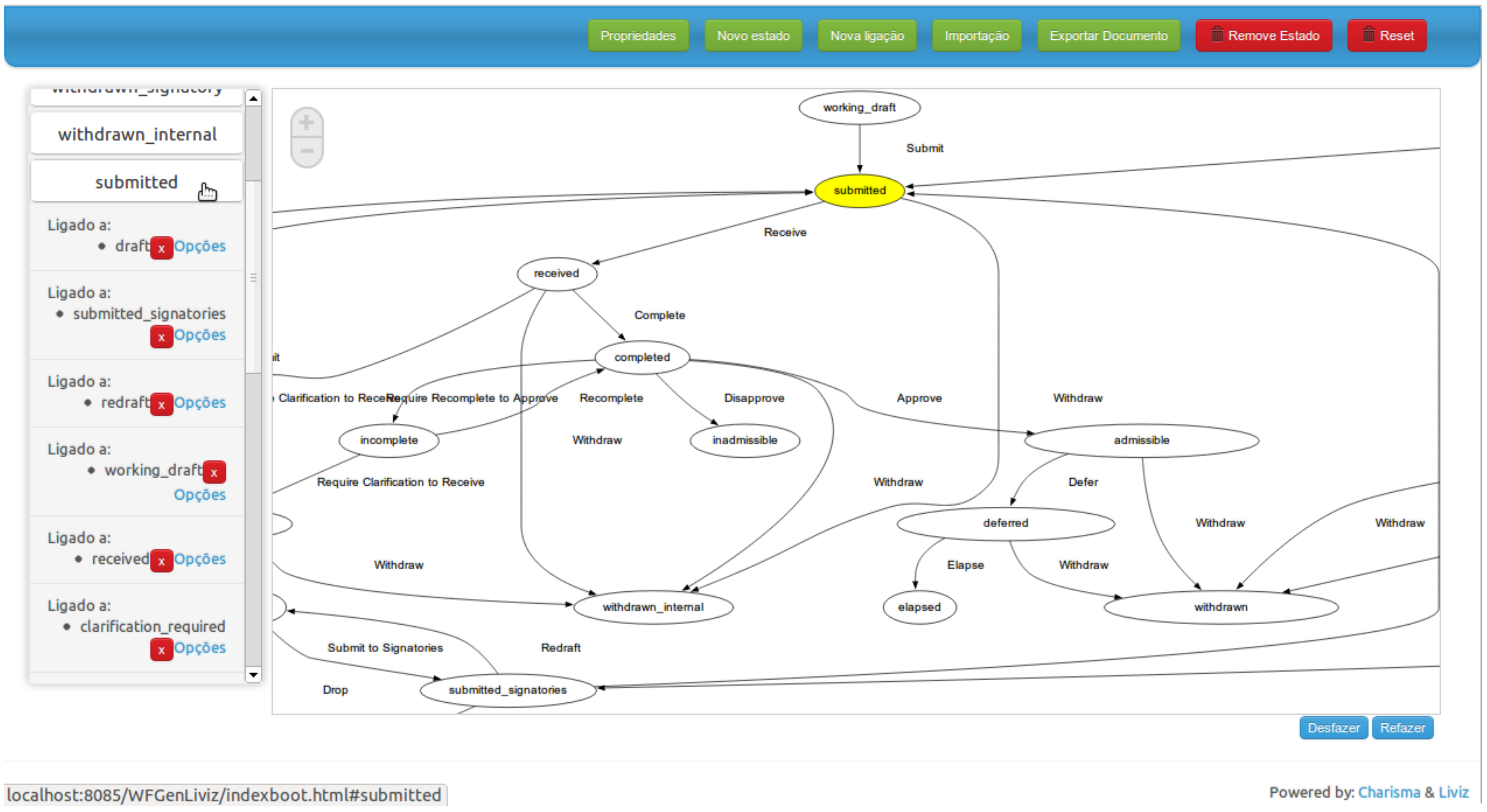


Figure 4.9: State Highlight and Zoom

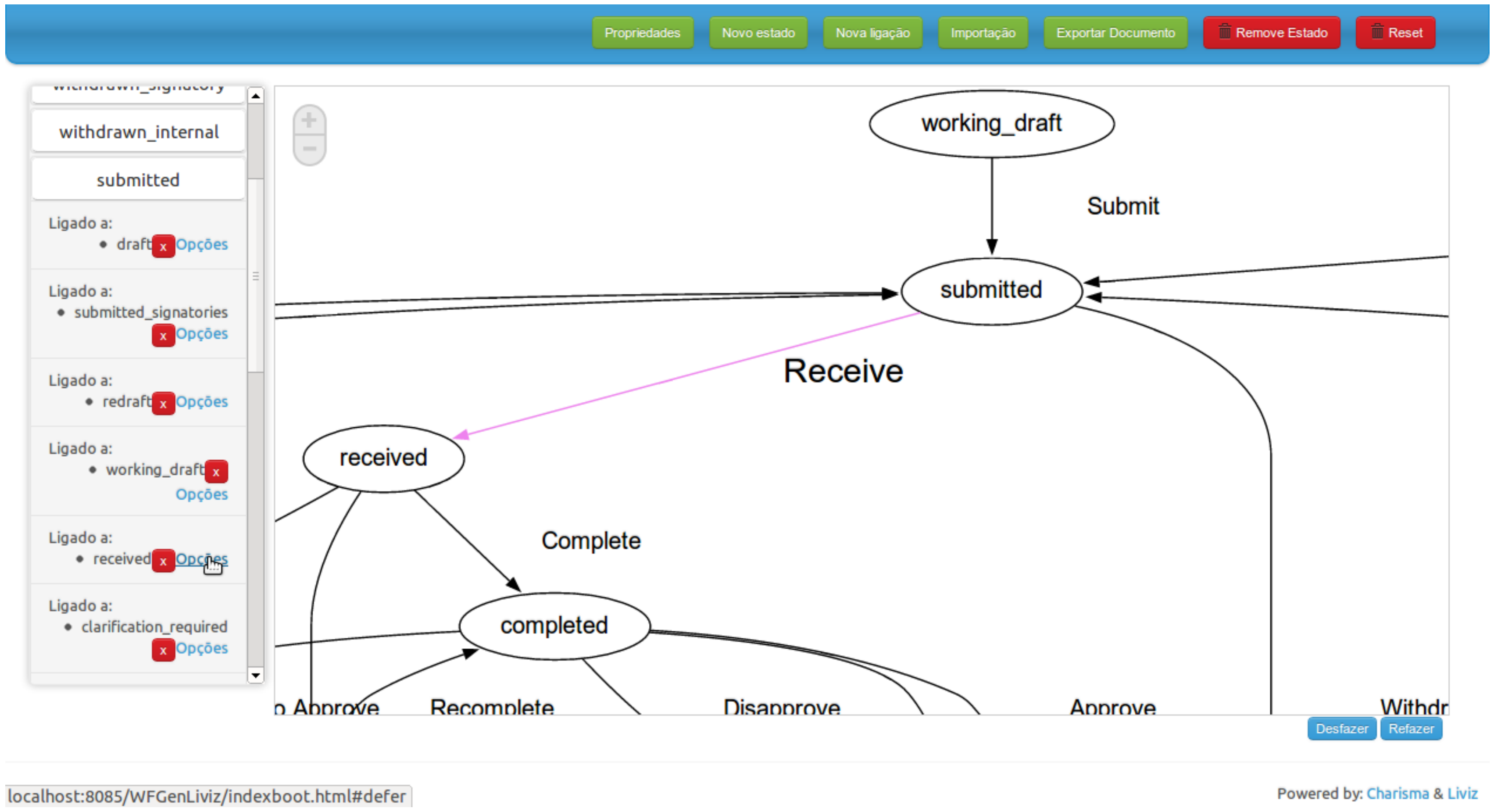


Figure 4.10: Connection Highlight and Zoom

Chapter 5

Results and Conclusion

5.1 Introduction

This chapter sums up the results of the developed work and the analysis that was made relative to the Bungeni Parliament System.

5.2 Bungeni Parliamentary System in the context of African e-Government

The Bungeni Parliamentary System as analyzed before, is a tool to support legislative documents management. The system is developed using open-source tools and is open-code and free. As seen on chapter one, Africa struggles to implement a full framework of e-Government services due to social economical problems. In the last decade, its e-Government developing index was always bellow 0.3. Since the problem is mostly economical, e-Government services such as the Bungeni Parliamentary System are a good method of improving at the minimum citizen relationship with local parliaments. Due to its open source nature it can be appropriated and located for virtually every African country.

African countries as those whose official language is Portuguese, are bellow and over the general average of the continent when the e-Government Development Index is referred to. As seen before, Mozambique (0.23) is located lower than the general average (0.27), while Angola and Sao Tome and Principe are well over, having approximately 0.33. Cape Verde and Guinea-Bissau are both on the opposite sides of the spectrum, the first having an index of 0.42 and the last 0.19. These countries don't have a support to electronic law making or legislature, and some of them have their sites located on Brazil servers. The Bungeni Parliamentary System has a Portal, that can be fully customized and have embedded links that refer other e-Government services simultaneously to their development. So other services already available in these countries would not be affected and would even be centralized and easier to find.

To summarize, due to its generic capabilities and configurations, the Bungeni Parliamentary System would be a low cost asset to improve the E-Government Development Index of these countries. Even Cape Verde could benefit from the usage of this system, despite having a great development index when compared to the others because of the e-law making platform.

5.3 Document Workflow Edition

Nowadays developing a new workflow can be time consuming and also complex, due to the different configurations that must be done in the XML files. The tasks to follow in order to create a workflow were already reffered in Chapter 3, and can be very time consuming, especially when developing a document with many states and transitions.

The workflow editor, aims and helps to produce workflows in an amount of time that is not comparable. On average a workflow implementation can take up to one month or two, depending on the complexity. Although great part of the time is spent mapping the workflow before implementing it in the system, many hours and even days are spent writing the XML files. Using the workflow editor same results can be accomplished in a matter of hours or even minutes, depending on the complexity of the workflow, once the proper modelling is accorded between the members of the legislative team. This time reduction in the modelling of the workflow can be explained due to the user interface, because only clicks are performed, instead of a repetitive process in which a visual result was only possible at the end.

5.4 Future Work

The future work in this case is regarding the Bungeni Parliamentary System, that should have a complete set of tests done to its functionalities, in order to understand the fragilities and problems of the services. Only with a complete set of tests the system can be exposed and rewritten, if that is the case, in seldom modules.

Some future work, would also be expand the Bungeni System to accommodate other types of E-Government services, centralizing the information and turning it into a country Portal.

Regarding the Workflow Editor, it could be redesigned to maximize productivity and to allow only the creation of workflows by registered system users, using the O-Auth service of Bungeni. It could also evolve and become a full configuration tool for the System, since the majority of the configurations is done by XML files editing. Since the Bungeni Parliamentary System has dictionaries and tools to translate the system between different languages, this tool should also follow that pattern.

Bibliography

[1] Donna Evans and David C. Yen. “E-Government: Evolving relationship of citizens and government, domestic, and international development”. In: *Government Information Quarterly* 23.2 (Jan. 2006), pp. 207–235. ISSN: 0740624X. DOI: `10.1016/j.giq.2005.11.004`. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0740624X05000936`.

[2] Christine B Williams and Girish J Gulati. “Social networks in political campaigns: Facebook and the 2006 midterm elections”. In: *Annual Meeting of the American Political Science Association*. 2007.

[3] The Nielsen Company. *The Social Media Report*. The Nielsen Company Website, 2012.

[4] M Howard. “e-Government Across the Globe: How Will “e” Change Government?” In: *e-Government* (2001). URL: `https://www.gfoa.org/downloads/eGovGFRAug01.pdf`.

[5] International Telecommunications Unions (ITU). *Key ICT indicators for developed and developing countries and the world (totals and penetration rates)*. (Visited on 03/08/2013).

[6] The Nielsen Company. *Global Trends in Online Shopping A Nielsen Global Consumer Report Online Shopping Around the World*. Tech. rep. June. 2010.

[7] Information Society and Public Sector. “The User Challenge Benchmarking The Supply Of Online Public Services”. In: September (2007).

[8] The World Bank. *Definition of E-Government*. URL: `http://web.worldbank.org/WBSITE/EXTERNAL/INFORMATIONANDCOMMUNICATIONANDTECHNOLOGIES/EXTEGOVERNMENT.html` (visited on 01/10/2013).

[9] The World Bank. *What is Governance?* URL: `http://web.worldbank.org/WBSITE/EXTERNAL/COUNTRIES/MENAEXT/EXTMNAREGTOPGOVERNANCE.html` (visited on 01/10/2013).

[10] Juan José Moreno-Navarro et al. “Modern development of a Law”. In: *ICT Law 2013*. 2013. ISBN: 9789899824133.

[11] United Nations. *E-Government Survey 2008*. Tech. rep. United Nations, 2008, p. 226. URL: `http://unpan1.un.org/intradoc/groups/public/documents/un/unpan028607.pdf`.

[12] United Nations. *E-Government Survey 2010*. 2010, p. 125. ISBN: 9789211231830. URL: `http://s3.amazonaws.com/connected\_republic/attachments/31/E\_Gov\_2010\_Complete.pdf`.

[13] United Nations. *E-Government Survey 2012*. 2012. ISBN: 9789211231908. URL: `http://unpan1.un.org/intradoc/groups/public/documents/un/unpan048065.pdf`.

[14] Republic of Austria. "The Austrian E-Government Act". In: *Federal Act on Provisions Facilitating Electronic Communications with Public Bodies* 10 (2004).

[15] Republic of Austria. "The E-LAW Project in Austria". In: *Republik Osterreich Parlament* (2006).

[16] Andreas Schaad, Pascal Spadone, and H Weichsel. "A case study of separation of duty properties in the context of the Austrian eLaw process." In: *2005 ACM symposium on Applied computing* (2005). URL: `http://dl.acm.org/citation.cfm?id=1066677.1066976`.

[17] Ministry of Security and Public Administrator of Korea. *EgovFrame Portal*. URL: `http://www.egovframe.go.kr/EgovIntro\_Eng.jsp` (visited on 09/02/2013).

[18] Iris Montes de Oca and Eduardo Glez. *Free e-gov Software System*. URL: `http://free-gov.org/` (visited on 09/02/2013).

[19] Nexedi & ERP5 Community. *ERP5 for Public Administration*. URL: `http://www.erp5.com/feature/egov` (visited on 10/02/2013).

[20] *Idega Open Source*. URL: `http://idega.github.io/egov.html` (visited on 10/02/2013).

[21] Africa i Parliaments. *Bungeni Parliamentary and Legislative System*. URL: `http://code.google.com/p/bungeni-portal/` (visited on 09/20/2012).

[22] European Parliament. *AT4AM for All*. URL: `http://www.at4am.org/` (visited on 10/03/2013).

[23] *Sistema de Informação de Investigação Criminal*. URL: `http://wiki.ieeta.pt/wiki/index.php/SIIC-2013` (visited on 09/05/2013).

[24] *Sistema de Informatização do Processo Civil de Cabo Verde*. URL: `http://wiki.ieeta.pt/wiki/index.php/SIPC\_CV` (visited on 09/03/2013).

[25] United Nations. *UN Public Administration Programme*. URL: `http://unpan3.un.org/egovkb/` (visited on 09/12/2013).

[26] Leadership Business Group. *Angola boosts ranking in e-Government*. URL: `http://www.leadership-bg.com/index.php?option=com\_content\&view=article\&id=321\&catid=1\&Itemid=18\&lang=en` (visited on 09/04/2013).

[27] *German Government Advised to Opt for Open Source.* URL: http://www.technolous.com/resources/articles/german-government-advised-to-opt-for-open-source (visited on 10/02/2013).

[28] Gijs Hillenius. *European Parliament expects open source usage report.* URL: http://opensource.com/government/12/6/european-parliament-expects-open-source-usage-reportOctober2013 (visited on 10/10/2013).

[29] *U.K. Parliamentarians Urge Government to Reshape Push for Open-Access Publishing.* URL: http://news.sciencemag.org/europe/2013/09/u.k.-parliamentarians-urge-government-reshape-push-open-access-publishing (visited on 10/10/2013).

[30] Portugal News. *Portuguese Government Gives Preference on Open Source Software to 2014.* URL: http://www.portugalnews.pt/tecnologia/governo-da-preferencia-ao-software-open-source-no-orcamento-de-estado-para-2014/ (visited on 10/10/2013).

[31] *Debian Based Linux Distibutions.* URL: http://www.debian.org/misc/children-distros (visited on 10/09/2013).

[32] Christian Vest Hansen. *Fabric.* URL: http://docs.fabfile.org/en/1.8/ (visited on 08/12/2012).

[33] Agendaless Consulting. *SupervisorD: A process control system.* URL: http://supervisord.org/ (visited on 08/12/2013).

[34] Ian Bicking. *Python Paste.* URL: http://pythonpaste.org/ (visited on 08/12/2013).

[35] Ian Bicking. *Deliverance transforms HTML to theme pages.* URL: http://pythonhosted.org/Deliverance/ (visited on 08/11/2013).

[36] Zope Foundation. *Zope Web Server.* URL: http://www.zope.org/ (visited on 05/02/2013).

[37] The PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open source database.* URL: http://www.postgresql.org/ (visited on 08/11/2013).

[38] Michael Bayer. *SQLAlchemy: The Python SQL Toolkit and Object Relational Mapper.* URL: http://www.sqlalchemy.org/ (visited on 05/02/2013).

[39] Olly Betts. *Xapian: Open Source Search Engine Library.* URL: http://xapian.org/ (visited on 08/12/2013).

[40] Plone Foundation and friends. *Plone CMS: Open Source Content Management.* URL: http://plone.org (visited on 11/13/2012).

[41] eXist Solutions. *eXist-db: Open Source Native XML Database.* URL: http://exist-db.org/ (visited on 04/12/2013).

[42] Pivotal Software. *RabbitMQ - Messaging that just works.* URL: http://www.rabbitmq.com/ (visited on 04/12/2013).

[43] Luke Arno. *Static: A really simple WSGI way to serve static (or mixed) content.* URL: https://pypi.python.org/pypi/static/0.4 (visited on 05/03/2013).

[44] The Apache Software Foundation. *OpenOffice: The Free and Open Productivity Suite.* URL: http://www.openoffice.org/en/ (visited on 03/10/2013).

[45] The Document Foundation. *Libre Office the Document Foundation.* URL: http://www.libreoffice.org/ (visited on 03/10/2013).

[46] *Web Server Gateway Protocol — WSGI.org.* URL: http://wsgi.org (visited on 04/10/2013).

[47] World Wide Web Consortium. *Real Simple Syndication.* URL: http://www.w3.org/WAI/highlights/about-rss (visited on 06/14/2013).

[48] The Zope Foundation. *Zope Object: OrderedContainer.* URL: http://apidoc.eea.europa.eu/zope.container.ordered.OrderedContainer-class.html (visited on 09/08/2013).

[49] Makoto Murata. *Relax NG: Parsing Framework.* URL: http://relaxng.org/ (visited on 09/10/2013).

[50] Flavio Zeni. *Bungeni Customization Guide.* URL: https://docs.google.com/document/d/1qyyQuJpOKdnyFYTgpnwrouPoc24IAik-KvY78Id_xAo/edit.

[51] Varnish Software. *Varnish cache.* URL: https://www.varnish-cache.org/ (visited on 05/10/2013).

[52] Agendaless Consulting. *repoze.who – WSGI Authentication Middleware.* URL: http://docs.repoze.org/who/1.0/ (visited on 09/09/2013).

[53] República Democrática de São Tomé e Príncipe. *Constituição da República Democrática de São Tomé e Príncipe.* URL: http://www2.camara.gov.br/saotomeeprincipe/constituicao/constituicao-da-republica-democratica-de-s.tome-e (visited on 03/10/2013).

[54] República Democrática de São Tomé e Príncipe. *Regimento Da Assembleia Nacional.*

[55] Twitter. *Twitter Bootstrap.* URL: http://getbootstrap.com/ (visited on 02/10/2013).

[56] Porrit, Simon. *jsPlumb.* URL: https://github.com/sporritt/jsplumb/ (visited on 12/14/2012).

[57] Ueyama Satoshi. *Liviz.js.* URL: http://ushiroad.com/jsviz/ (visited on 02/20/2013).

[58] AT&T Labs Research. *GraphViz - Graph Visualization Software.* URL: http://www.graphviz.org/ (visited on 11/13/2012).

[59] John Resig. *jQuery Javascript Framework.* URL: http://jquery.com/ (visited on 09/10/2013).

[60] Juan Escobar. *Raphael Zoom and Pan Plugin.* URL: https://github.com/escobar5/raphael-pan-zoom (visited on 02/05/2013).

[61] Dmitry Baranovskiy. *Raphael Javascript Library.* URL: http://raphaeljs.com/.

[62] James Mills. *Python Interface to Graphviz dot language.* URL: http://code.google.com/p/pydot/ (visited on 12/03/2013).

[63] Marcel Hellkamp. *Bottle: Python Web Framework.* URL: http://bottlepy.org.

[64] Mariano Raeingart. *pyFPDF: Simple PDF generation for Python.* URL: http://code.google.com/p/pyfpdf/ (visited on 04/16/2013).

[65] C. Batini, E. Nardelli, and R. Tamassia. "A layout algorithm for data flow diagrams". In: *Software Engineering, IEEE Transactions on* SE-12.4 (1986), pp. 538–546. ISSN: 0098-5589. DOI: 10.1109/TSE.1986.6312901.

Appendix A

Appendix

A.1 XML Code Generated by the Workflow Editor

This is an example of the XML code, produced by the workflow editor. It represents a configuration file for the Bungeni Parliament System, to draw the workflow represented on Figure A.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<workflow permissions=".View .Edit .Add .Delete" title="dissert">
   <allow permission=".Edit" role="bungeni.Speaker" />
   <allow permission=".Add" role="bungeni.Speaker" />
   <feature action="schedule" enabled="true" />
   <feature action="notification" enabled="true" />
   <feature action="signatory" enabled="true" />
   <feature action="attachment" enabled="true" />
   <feature action="workspace" enabled="true" />
   <feature action="download" enabled="true" />
   <feature action="event" enabled="true" />
   <feature action="group_assignment" enabled="true" />
   <state id="scheduled" title="scheduled" />
   <state id="scheduled" title="scheduled" />
   <state id="scheduled" title="scheduled" />
   <state id="working_draft" title="working_draft" />
   <state id="adjourned" title="adjourned" />
   <state id="adjourned" title="adjourned" />
   <state id="clarification_required" title="clarification_required" />
   <state id="completed" title="completed" />
   <state id="rejected" title="rejected" />
   <state id="elapsed" title="elapsed" />
   <state id="dropped" title="dropped" />
   <state id="withdrawn" title="withdrawn" />
```

```
<state id="withdrawn" title="withdrawn" />
<state id="adopted_amendments" title="adopted_amendments" />
<state id="received" title="received" />
<state id="withdrawn_internal" title="withdrawn_internal" />
<state id="inadmissible" title="inadmissible" />
<state id="submitted_signatories" title="submitted_signatories" />
<state id="submitted" title="submitted" />
<state id="adopted" title="adopted" />
<state id="deferred" title="deferred" />
<state id="deferred" title="deferred" />
<state id="admissible" title="admissible" />
<state id="admissible" title="admissible" />
<state id="admissible" title="admissible" />
<state id="draft" title="draft" />
<state id="withdrawn_signatory" title="withdrawn_signatory" />
<state id="redraft" title="redraft" />
<state id="incomplete" title="incomplete" />
<transition condition="" destination="submitted" roles="roles"
    source="submitted_signatories" title="Submit" />
<transition condition="" destination="inadmissible" roles="clerk_Assembly"
    source="completed" title="Disapprove" />
<transition condition="" destination="rejected" roles="clerk_Assembly"
    source="scheduled" title="Reject a scheduled motion" />
<transition condition="" destination="submitted" roles="clerk_Assembly"
    source="redraft" title="Submit" />
<transition condition="" destination="withdrawn_internal" roles="clerk_Assembly"
    source="received" title="Withdraw" />
<transition condition="" destination="adopted_amendments" roles="clerk_Assembly"
    source="scheduled" title="Adopt with amendments" />
<transition condition="" destination="withdrawn_signatory"
    roles="clerk_Assembly" source="submitted_signatories" title="Drop" />
<transition condition="" destination="submitted" roles="clerk_Assembly"
    source="draft" title="Submit" />
<transition condition="" destination="admissible" roles="clerk_Assembly"
    source="completed" title="Approve" />
<transition condition="" destination="adopted" roles="clerk_Assembly"
    source="scheduled" title="Adopt" />
<transition condition="" destination="redraft" roles="clerk_Assembly"
    source="clarification_required" title="Redraft (Automatic)" />
<transition condition="" destination="deferred" roles="clerk_Assembly"
    source="admissible" title="Defer" />
<transition condition="" destination="elapsed" roles="clerk_Assembly"
    source="deferred" title="Elapse" />
```

```
<transition condition="" destination="incomplete" roles="clerk_Assembly"
    source="completed" title="Require Recomplete to Approve" />
<transition condition="" destination="withdrawn_signatory"
    roles="clerk_Assembly" source="redraft" title="Drop" />
<transition condition="" destination="withdrawn" roles="clerk_Assembly"
    source="admissible" title="Withdraw" />
<transition condition="" destination="withdrawn_internal" roles="clerk_Assembly"
    source="submitted" title="Withdraw" />
<transition condition="" destination="submitted_signatories"
    roles="clerk_Assembly" source="draft" title="Submit to Signatories" />
<transition condition="" destination="adjourned" roles="clerk_Assembly"
    source="scheduled" title="Adjourn debate" />
<transition condition="" destination="completed" roles="clerk_Assembly"
    source="received" title="Complete" />
<transition condition="" destination="submitted_signatories"
    roles="clerk_Assembly" source="redraft" title="Submit to Signatories" />
<transition condition="" destination="clarification_required"
    roles="clerk_Assembly" source="incomplete" title="Require Clarification to
    Receive" />
<transition condition="" destination="dropped" roles="clerk_Assembly"
    source="scheduled" title="Drop" />
<transition condition="" destination="withdrawn_internal" roles="clerk_Assembly"
    source="clarification_required" title="Withdraw" />
<transition condition="" destination="withdrawn" roles="clerk_Assembly"
    source="deferred" title="Withdraw" />
<transition condition="" destination="withdrawn_internal" roles="clerk_Assembly"
    source="completed" title="Withdraw" />
<transition condition="" destination="redraft" roles="clerk_Assembly"
    source="submitted_signatories" title="Redraft" />
<transition condition="" destination="received" roles="clerk_Assembly"
    source="submitted" title="Receive" />
<transition condition="" destination="completed" roles="clerk_Assembly"
    source="incomplete" title="Recomplete" />
<transition condition="" destination="clarification_required"
    roles="clerk_Assembly" source="received" title="Require Clarification to
    Receive" />
<transition condition="" destination="submitted" roles="clerk_Assembly"
    source="working_draft" title="Submit" />
<transition condition="" destination="submitted" roles="clerk_Assembly"
    source="clarification_required" title="Resubmit" />
<transition condition="" destination="withdrawn" roles="clerk_Assembly"
    source="scheduled" title="Withdraw" />
<transition condition="" destination="withdrawn" roles="clerk_Assembly"
```

```
        source="adjourned" title="Withdraw" />
</workflow>
```

A.2 Workflow Representation of a Worflow Created on Editor

Figure A.1, represents a workflow created using the previous XML code.

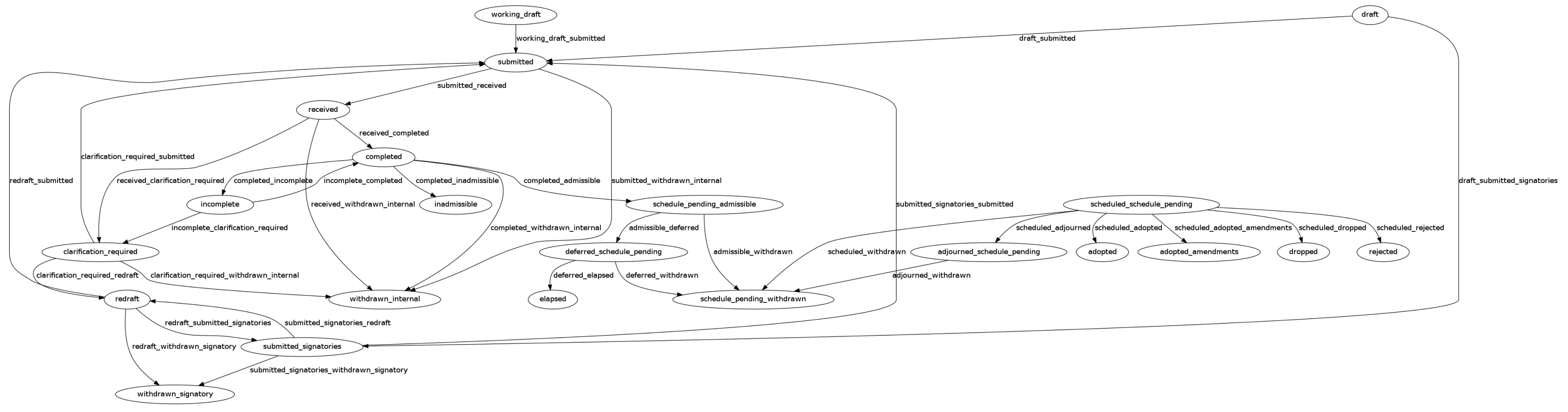

working_draft
working_draft_submitted
draft
draft_submitted
submitted
submitted_received
received
received_completed
completed
redraft_submitted
clarification_required_submitted
received_clarification_required
completed_incomplete
incomplete_completed
completed_inadmissible
completed_admissible
submitted_withdrawn_internal
submitted_signatories_submitted
draft_submitted_signatories
incomplete
received_withdrawn_internal
inadmissible
schedule_pending_admissible
scheduled_schedule_pending
incomplete_clarification_required
completed_withdrawn_internal
admissible_deferred
scheduled_adjourned
scheduled_adopted
scheduled_adopted_amendments
scheduled_dropped
scheduled_rejected
clarification_required
deferred_schedule_pending
admissible_withdrawn
scheduled_withdrawn
adjourned_schedule_pending
adopted
adopted_amendments
dropped
rejected
clarification_required_redraft
clarification_required_withdrawn_internal
deferred_elapsed
deferred_withdrawn
adjourned_withdrawn
redraft
withdrawn_internal
elapsed
schedule_pending_withdrawn
redraft_submitted_signatories
submitted_signatories_redraft
redraft_withdrawn_signatory
submitted_signatories
submitted_signatories_withdrawn_signatory
withdrawn_signatory

Figure A.1: Example of a workflow produced by the User Friendly Editor