



Colosseum 27
7521 PV Enschede
The Netherlands

www.tno.nl

T +31 53 483 52 00

F +31 53 483 52 22

Public service improvement using runtime service composition strategies

Place Enschede
Date 20th August 2010
Author(s) Joni Hoppen dos Santos

Institute University of Twente, Enschede the Netherlands
Faculty School of Management and governance (SMG)
Programme Business Information Technology (MBI)

Company TNO (Netherlands Organization for Applied Scientific Research)

Graduation
committee: Luís Ferreira Pires
 1st *university supervisor*
 Electrical Engineering, Mathematics and Computer science (EEMCS)

 Maria Eugenia Iacob
 2nd *university supervisor*
 School of Management and governance (SMG)

 Eduardo Gonçalves da Silva
 3rd *university supervisor*
 Electrical Engineering, Mathematics and Computer science (EEMCS)

 Michael van Bekkum
 1st *company supervisor*
 TNO Information and Communication Technology

 Wout Hofman
 2nd *company supervisor*
 TNO Information and Communication Technology

All rights reserved. No part of this report may be reproduced and/or published in any form by print, photoprint, microfilm or any other means without the previous written permission from TNO.

All information which is classified according to Dutch regulations shall be treated by the recipient in the same way as classified information of corresponding value in his own country. No part of this information will be disclosed to any third party.

In case this report was drafted on instructions, the rights and obligations of contracting parties are subject to either the Standard Conditions for Research Instructions given to TNO, or the relevant agreement concluded between the contracting parties. Submitting the report for inspection to parties who have a direct interest is permitted.

Management Summary

A runtime customer-driven service composition strategy can improve public services if it is able to operationalize the main principles of the life event strategy, enabling citizens to dynamically compose services according to their goals through a single point of access. This operationalization is achieved by defining a citizen-oriented interaction pattern that links front-end to the service composition engine. We assume that those principles of the life event strategy can improve the quality of public services.

The demand for service composition in the e-government context has two main reasons:

1. With the advent of the Internet, individuals are demanding more personalized services.
2. To have an overall acceptance of the e-government systems, they need to be integrated and easy to use.

To address these requirements for e-government, we have studied the life events strategy and built a solution framework with suitable technologies to support this strategy.

The life events strategy suggests that e-government web portals need to assume the perspective of the citizens by interacting with citizens to understand what they need rather than only publishing information. The aspects that are relevant to the citizens are the events that take place in their lives and require public services. Examples of life events are birth, school inscription, change of address and marriage. Furthermore, all these events should be supported through a single point of access that shields the citizens from the fragmentation and complexity of the governmental structure.

To support the life events strategy, we developed a solution framework that adopts the PSE-TNO solution as the front-end and DynamiCoS framework as the service composition engine in the back-end. Our main contribution consists of an interaction pattern between the front and back-end that enables users (citizens) to create personalized services at runtime. The suggested interaction pattern is implemented as an orchestrator of the web service orchestration engine, which defines the workflow of *interaction types* that the user follows while performing a service composition.

As proof of concept, we have built a prototype and demonstrated a dynamic service composition performed by a disabled citizen who requires a handicapped parking place next to his house.

Management Samenvatting

Een runtime, klantgestuurde strategie voor samengestelde diensten kan publieke dienstverlening verbeteren, als het de belangrijkste uitgangspunten van de "life events" strategie ondersteunt, waarbij gebruikers volgens hun eigen doelstellingen dynamisch diensten kunnen samenstellen via één enkel toegangspunt. De uitvoering daarvan kan worden bereikt door een klantgestuurd interactiepatroon te gebruiken, dat een front-end verbindt met een engine om diensten samen te stellen (service composition engine). We gaan er vanuit dat de principes van de "life event strategie" daarmee daadwerkelijk de publieke dienstverlening kunnen verbeteren.

The vraag naar samengestelde diensten in de context van de e-overheid kent twee oorzaken:

1. De opkomst van internet heeft ervoor gezorgd dat burgers een meer gepersonaliseerde dienstverlening verwachten.
2. Voor algemene acceptatie, moeten e-overheidssystemen in hoge mate geïntegreerd zijn en eenvoudig zijn in gebruik.

Om deze eisen in te vullen, hebben wij de "life events" strategie bestudeerd en een solution framework opgesteld met geschikte technologieën om deze aanpak te ondersteunen.

De "life events" strategie stelt dat elektronische overheidsportalen vanuit het perspectief van de burger moeten werken, door de dialoog aan te gaan met de burger om zo te achterhalen wat voor deze burger relevant is voor in plaats van alleen maar informatie te verstrekken. De aspecten die relevant zijn, zijn juist die gebeurtenissen in het leven van de burger/gebruiker die een publieke dienst vereisen. Voorbeelden hiervan zijn: geboortes, inschrijvingen op scholen, verhuizingen (verandering van adres) en trouwen. Daarnaast moet de ondersteuning voor elk van deze gebeurtenissen beschikbaar zijn vanuit een enkel toegangspunt, wat de fragmentatie en complexiteit van het achterliggende overheidssysteem afschermt voor de gebruiker.

Om de "life events" strategie te ondersteunen hebben wij een solution framework ontwikkeld dat de PSE-TNO oplossing als front-end gebruikt en het DynamiCoS framework als back-end als service composition engine. Onze belangrijkste bijdrage bestaat daarbij uit een interactie patroon tussen de front- en back-end dat de gebruikers in staat stelt om een gepersonaliseerde dienst te creëren na een dynamische interactie. Het voorgestelde interactiepatroon is in feite een instrumentatie van de web service orchestration engine, die de werkstroom van activiteiten in de interactie vaststelt, die de gebruiker volgt terwijl hij een dienst samenstelt.

Als testcase hebben we een prototype gebouwd en geëxperimenteerd met een dynamische samenstelling van diensten, uitgevoerd door een invalide burger die een invalidenparkeerplaats nodig heeft naast zijn huis.

Sumário Executivo

A composição de serviços orientada ao consumidor em tempo de execução pode ser de grande utilidade na melhoria dos serviços públicos, caso esta seja capaz de operacionalizar os princípios da estratégia conhecida como eventos de vida. Este tipo de composição de serviço permite aos cidadãos compor serviços dinamicamente, de acordo com seus requisitos, através de um único ponto de acesso. Esta operacionalização é obtida através de uma definição de um padrão de interação orientada ao cidadão, que permite conectar a interface do usuário com os mecanismos de composição de serviço. Neste trabalho é assumido que os princípios da estratégia (eventos de vida) podem melhorar a qualidade dos serviços públicos.

A demanda por composição de serviços no contexto de governo eletrônico tem duas razões principais:

1. Com o advento da Internet, usuários demandam serviços mais personalizados.
2. Para terem um bom nível de aceitação, os serviços necessitam ser integrados e fáceis de serem utilizados.

Para atingir tais requisitos no contexto de governo eletrônico, foram investigados os aspectos da estratégia de eventos de vida para desenvolver uma solução com tecnologias adequadas para suportarem esta estratégia.

A estratégia de eventos de vida sugere que os portais de governo eletrônico necessitam interagir com os cidadãos para atender suas necessidades, ao invés de simplesmente publicar informações. Os aspectos mais relevantes aos cidadãos são aqueles cujos eventos fazem parte de suas vidas e requerem serviços públicos. Exemplos de tais serviços incluem nascimento, matrícula escolar, alteração de endereço e casamento. Além disso, todos estes eventos devem ser disponibilizados através de um único ponto de acesso que protegem os usuários da fragmentação e complexidade da estrutura governamental.

Para suportar a estratégia de eventos de vida, foi desenvolvido neste trabalho uma solução que adota a os sistemas PSE-TNO como o interface de usuário e DynamiCoS como a máquina de composição de serviços. A principal contribuição deste trabalho consiste na determinação de um padrão de interação entre os dois sistemas, o que permite aos usuários (cidadãos) criarem serviços personalizados em tempo de execução. O padrão de interação proposto pode ser interpretado como um orquestrador da máquina de orquestração de composição de serviços, que define o workflow dos tipos de interação que os usuários devem seguir quando executam a composição de serviços.

Como prova de conceito e viabilidade da proposta, um protótipo foi implementado e a composição dinâmica de serviços foi executada tendo como estudo de caso um cidadão portador de necessidades especiais, que necessita de uma vaga de estacionamento adequada a suas necessidades nas proximidades de sua casa.

“Não ta morto quem peleia”

Acknowledgements

The writing of this master thesis at the University of Twente marks the end of two years of intensive studies in the Netherlands, where I learnt not only a great deal of technical information, but also and most importantly, the remarkable aspects of the Dutch culture, which I hope to carry with me for the rest of my life.

First, I offer my sincerest gratitude to my supervisors Luís Pires, Mari Jacob, Eduardo Silva, Michel van Bekkum and Wout Hofman for their guidance, which helped me spot and tackle the research problem effectively. I also want to thank them for revising texts and engaging in numerous conducive meetings.

I would also like to express my gratitude to Menno Holtkamp who has made available his support in various ways. He was the linking person between the university and the company. I wish him a lot of success being the linking person between Brazil and the Netherlands.

Philosopher and pioneer at heart, Marcos Henrique dos Santos deserves many thanks for sharing his sparkly ideas, which in recent years, have deeply shaped my worldview. I am delighted to see that some of his theories are now being used to leverage businesses and change the path of development of the Brazilian economy. I wonder if we could use them to change the massive Brazilian football-based economy into a solid knowledge-based one.

I am indebted to Frederik Bonte and John Donker for their unconditional help during the coding phase. The prototype would surely not exist in such a short timeframe without all their drawings, explanations and patience. Thanks to Diego Garcia, Giovane Moura and Joe Loufer for unconditional help some parts of the text.

Thanks to Mr. Paul HoppenBrouwers, for mentioning the word "scholarship". I even remember looking it up in the dictionary. Long ago, he also translated a letter from poor English to Dutch. I gave it to the princess of the Netherlands when she was visiting Florianopolis. As a result I received some directions to apply for a scholarship, which eventually worked out. I am glad the Dutch government makes this channel of interaction available to international students. I hope I can help other student pass through this long and competitive selection process.

All in all, this work is dedicated to Ivadir Alves dos Santos and Bernardete Hoppen for dedicating so much of themselves for the sake of the education of their children (Rafael, Lilian and I). I also would like to acknowledge all the families that helped me with shelter, books and even with some food during critical moments when the line of dignity was broken for the sake of some ideals.

Thanks to the Bosman family and Silvie Pothof for demonstrating the core values of the Dutch culture, which remains a mystery to many foreigners who live in the Netherlands.

Last but not least, I offer my regards and blessings to all of those who supported me in any respect during the completion of the master in Business Information Technology.

Contents

1	INTRODUCTION	1
1.1	CONTEXT.....	1
1.2	PROBLEM DESCRIPTION	2
1.3	RESEARCH OBJECTIVE	3
1.4	RESEARCH QUESTIONS	3
1.5	RESEARCH APPROACH	4
1.6	THESIS STRUCTURE	5
2	E-GOVERNMENT.....	7
2.1	DEFINITION E-GOVERNMENT	7
2.2	THE GOAL OF E-GOVERNMENT	8
2.3	CURRENT CHALLENGES	8
2.4	CLASSIFICATION OF E-GOVERNMENT INITIATIVES	9
2.5	DUTCH E-GOVERNMENT INITIATIVE.....	11
3	SERVICES AND SERVICE COMPOSITION TECHNOLOGIES	15
3.1	SERVICE-ORIENTED ARCHITECTURE	15
3.2	TECHNOLOGIES FOR SOA	17
3.3	BUSINESS PROCESSES	18
3.4	SERVICE COMPOSITION.....	18
3.5	DYNAMIC SERVICE COMPOSITION	21
3.6	DYNAMIC SERVICE COMPOSITION APPROACHES	22
3.6.1	<i>WSMF</i>	23
3.6.2	<i>METEOR-S</i>	23
3.6.3	<i>DynamiCoS</i>	24
4	TECHNOLOGIES FOR ENHANCING E-GOVERNMENT EXPERIENCE.....	27
4.1	SOLUTION FRAMEWORK OVERVIEW	27
4.2	PUBLIC SERVICE EXPERIENCE - TNO	28
4.2.1	<i>Overview</i>	29
4.2.2	<i>User Interface</i>	30
4.2.3	<i>Natural language recognition</i>	31
4.2.4	<i>Supporting technologies</i>	32
4.3	DYNAMICO S FRAMEWORK.....	33
4.3.1	<i>Overview</i>	33
4.3.2	<i>DynamiCoS architecture</i>	34
4.3.3	<i>Interaction types</i>	34
4.3.4	<i>Interaction pattern</i>	37
4.3.5	<i>Supporting technologies</i>	37
4.4	CHARACTERIZATION OF USERS	42
4.4.1	<i>Citizen interaction pattern</i>	43
4.4.2	<i>Civil servant interaction pattern</i>	47
5	HANDICAP PARKING PERMIT – PROTOTYPE	51
5.1	CASE CONTEXT	51

5.2	SERVICE DEFINITION.....	51
5.3	PUBLIC SERVICE ONTOLOGIES.....	53
5.3.1	<i>E-gov ontology</i>	53
5.3.2	<i>IO ontology</i>	55
5.4	IMPLEMENTATION.....	56
5.5	TESTING SCENARIO.....	59
5.5.1	<i>Situation 1</i>	59
5.5.2	<i>Situation 2</i>	62
6	CONCLUSIONS	67
6.1	ANSWERS TO THE RESEARCH QUESTIONS.....	67
6.2	CONTRIBUTIONS.....	69
6.3	LIMITATIONS.....	70
6.4	FUTURE WORK	70
	REFERENCES.....	73

List of figures

Figure 1: Public service fragmentation [4].	1
Figure 2: Provision of an interrelated public service.	2
Figure 3: Research model and thesis structure.	4
Figure 4: Research information resources. Adapted from [13]	5
Figure 5: The evolution of public service delivery [32].	8
Figure 6: Classification scheme for e-government [39]	10
Figure 7: Overview of the Dutch e-government architecture [36].	12
Figure 8: The evolution of the Dutch e-government.	13
Figure 9: Overview of a simple service request	16
Figure 10: Department interaction based on SOA.	17
Figure 11: web service protocol layer [12].	18
Figure 12: Sequential composition of web services.	19
Figure 13: Parallel composition of web services.	19
Figure 14: Web service execution after a decision point.	19
Figure 15: Ordering handicap parking permit (UML activity diagram).	20
Figure 16: BPEL process composition of the ordering handicap parking permit.	21
Figure 17: Backward vs. forward chaining of services	22
Figure 18: Solution framework overview.	27
Figure 19: Conceptual model for service discovery and mediation of PSE [8]	29
Figure 20: Current solution developed by TNO.	30
Figure 21: User-interface layout.	31
Figure 22: Natural language query demonstration.	32
Figure 23: DynamiCoS framework architecture.	34
Figure 24: DynamiCoS WSDL visualization.	35
Figure 25: DynamiCoS and current supporting technologies [45].	38
Figure 26: SPATEL description of the address validation web service.	40
Figure 27: Matching parameters between services adapted from [17].	41
Figure 28: Citizen <i>Interaction pattern</i> – Activity diagram [73].	45
Figure 29: Citizen <i>Interaction pattern</i> – Sequence diagram [73].	47
Figure 30: Civil servant interaction pattern.	49
Figure 31: Web services for the e-government scenario.	52
Figure 32: Extended ontology of public services (e-gov ontology).	54
Figure 33: E-government ontology taxonomy.	55
Figure 34: IO ontology.	56
Figure 35: Prototype structure.	58
Figure 36: S1Creator.php	58
Figure 37: S1.xml	58
Figure 38: SR1Creator.php	59
Figure 39: First citizen interaction.	60
Figure 40: Service discovery result.	60
Figure 41: Validation of the inputs.	61
Figure 42: Service validation results	61

Figure 43: service execution.....	62
Figure 44: Empty form to be validated.	63
Figure 45: Inputs to be resolved	63
Figure 46: Automatic suggestion of services to fulfil the missing inputs.....	64
Figure 47: Automatic suggestion of services to fulfil the missing inputs (Continuation)	64
Figure 48: Order parking permit selected and ready to be validated.....	65
Figure 49: Fee payment missing.....	65
Figure 50: Fee validation web service	66
Figure 51: Validation of the fee validation service.....	66
Figure 52: Fee validation service execution	66

List of tables

Table 1: Front-end supporting technologies	32
Table 2: Interaction types	35
Table 3: Semantic web service matching patterns	41
Table 4: End-users' classification for e-government.....	43

List of abbreviations

BPEL	Business process execution language
DynamiCoS	A framework for D ynamic C omposition of S ervices;
e-Government	Electronic Government
IT	Information Technology
OWL	web Ontology Language
PA	Public Administration
PSE	Public Service Experience
RDF	Resource Description Framework
SEW	Service Engineering Workbench
SOA	Service-oriented Architecture
SOAP	Simple Object Access Control
SOC	Service-Oriented Computing
TNO	Netherlands Organisation for Applied Scientific Research
UDDI	Universal Description Discovery and Integration
UI	User Interface
UML	Unified Modeling Language
URL	Universal Resource Locator
W3C	World Wide web Consortium
WSDL	Web service Description Language
WSMF	Web service Modeling Framework
WSML	Web service Modeling Language
WSMO	Web service Modeling Ontology
WSMX	Web service Execution Environment
XML	Extensible Mark-up Language
XSD	XML Schema Definition

1 Introduction

This chapter introduces our research by discussing its context, problem description, research objective, research questions, approach, and structure.

1.1 Context

The primary function of most organizations is to attend the needs of their customers by performing a series of internal activities (processes). These processes must take into account the internal capabilities of the organization in relation to its environment [1-2]. However, with the advent of the Internet and digital economy the business environment became more dynamic, which demands more flexible business processes. This fact poses managerial challenges to virtually all sectors of the economy.

In the public sector, the government is responsible for providing the most fundamental services that support the development of its society [3]. However, although governments world-wide are regarded as essential institutions, they are notorious for being bureaucratic and fragmented.

The underlying purpose of public services is to be an instrument that allows people to exercise their citizenship throughout their lifetime. Therefore, overtime, citizens have to interact directly or indirectly with a public institution to achieve their civil needs. However, because of its inherent organizational fragmentation, the government might pose several unnecessary obstacles that prevent citizens from accomplishing their goals. Figure 1 depicts a stereotyped view of a public organization when an ordinary person requests public service.

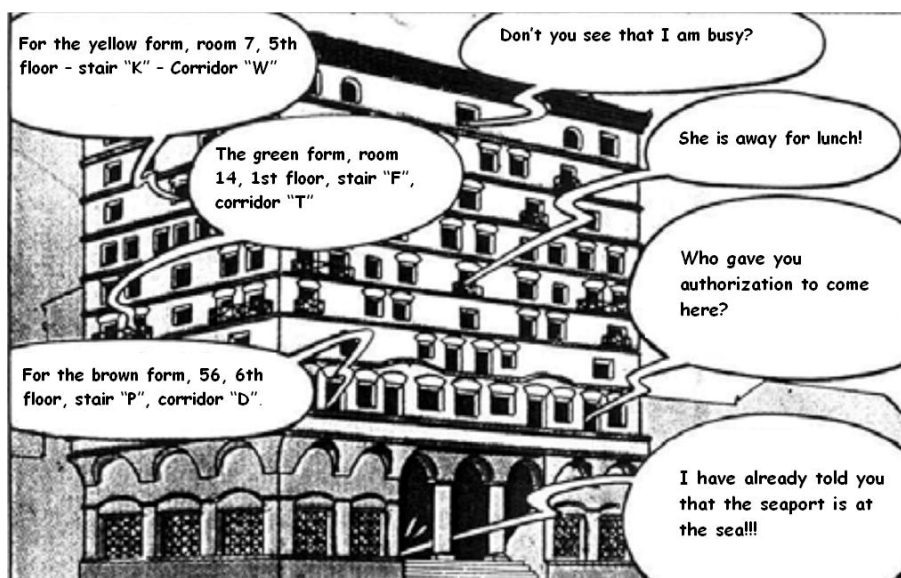


Figure 1: Public service fragmentation [4].

In recent decades, governments around the world have acknowledged their awareness about the quality of public services and the pivotal role that information technologies (IT) can play in

reducing the gap between the government and its citizens [5]. As a consequence, national policies were designed to foster the utilization of IT in the governmental setting, originating the term electronic government (e-government) [5].

1.2 Problem description

Despite the well-known bureaucracy and fragmentation of the public sector, governmental institutions from many countries are striving to improve their customer's service by applying information system technologies. Dutch e-government initiatives, in turn, already offer a wide variety of online public services (see examples in [6-7]). Nonetheless, most of them still reflect the inherent physical and logical fragmentation of the governmental structure. This is considered one of the main obstacles to improve the overall quality of the electronic public services [8-9].

For instance, some government departments might offer a service that requires functions that belong to another sectors, but the current systems are still not fully interoperable, obliging citizens to figure out by themselves the course of action to achieve their goals amongst a multitude of disparate services [4]. Figure 2 illustrates an imaginary scenario of the interdependency and fragmentation of public services when a parking permit is required. This fragmentation is replicated in the digital version [8].

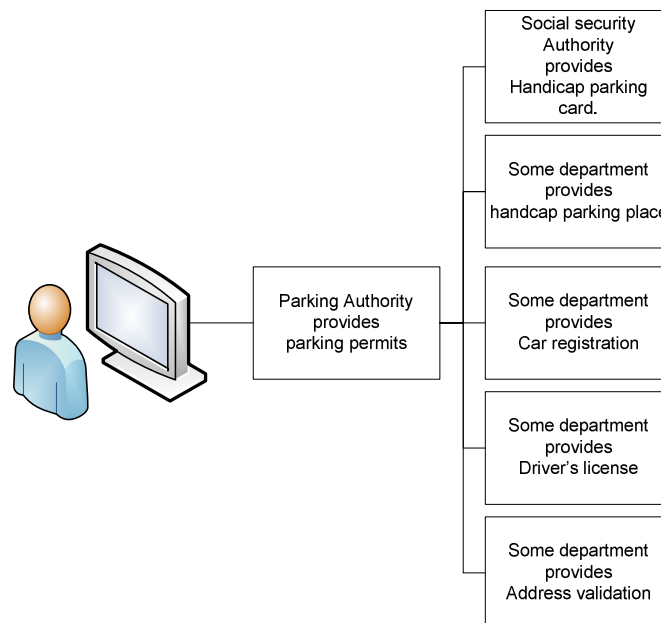


Figure 2: Provision of an interrelated public service.

Figure 2 shows a citizen who requires a handicap parking permit. To do this, he/she must contact other departments that achieve the prerequisites of this service. For example: the registration of a car has no direct connection with the department that validates citizens' addresses. Therefore, interoperability mechanisms should be in place to overcome this organizational fragmentation [8-9].

To tackle the fragmentation of digital services, the service-oriented architecture (SOA) was developed [10-13]. This architectural style envisions digital systems interoperating in a loosely coupled manner to achieve a business requirement [2, 11]. In other words, SOA facilitates the interactions between departments when they need to exchange information. In the previous case, with SOA, the parking authority would request and receive information from the other departments if necessary.

Furthermore, among the benefits of SOA is the possibility of creating value-added services by combining several existing ones, which is also known as a service composition [10-12, 14-16]. For example, the parking authority could offer richer services by adding extra information to the services that is already provided. In this way, this department could make use of an external service that identifies the geographical location of the citizen, suggesting other parking places in the region.

Even though many public and private organizations already have implemented SOA [10-12], two issues stand out:

1. The promises of SOA are only fully realized when services meet the end-users' requirements.
2. The activity of finding and composing disparate services is still challenging and time-consuming [15].

In the literature, there are several strategies to automate service composition [17-20], and although many of them show interesting results, not much attention has been given to the interaction between users and environments (tools) that support service composition [19, 21-22]. Improvements in this area could potentially improve the quality of these services through a better alignment between citizens' goals and public services capabilities.

1.3 Research Objective

Although the Dutch government has already made several public services available to its citizens over the Internet, those services still reflect the physical and logical fragmentation of the governmental structure. To tackle this problem, new information technology approaches such as SOA were adopted, but these still present some shortcomings. Therefore, the objective of this research is:

“To identify ways to improve public services using a runtime citizen-driven service composition approach, and demonstrate this approach in a prototype”

1.4 Research questions

To attain the research objectives we present the main research question.

“How can a runtime customer-driven service composition strategy improve public services?”

Below, we define the research sub-questions that help us answer the main research question. They are divided into two main groups: knowledge questions (KQ) and practical questions (PQ)

[23]. A distinction between these questions is important because it contributes to a better understanding of the problem itself, and the purpose and scope of the solution.

On one hand, knowledge questions aim to help acquire information about a certain subject. On the other hand, practical questions have to do with identifying the procedures to attain the intended outcomes. Answers for practical questions should be evaluated according to their usefulness for solving a specific problem. In this research, we want to define a strategy for enabling citizen-driven service composition in the e-government context. This strategy has to adapt and reuse already available solutions for the front-end and back-end.

Our research sub-questions are:

RQ1 - What are public services, e-government, and the current state of affairs of the Dutch e-government? (KQ)

RQ2 - What is service composition and why is it necessary? (KQ)

RQ3 - What are the current technologies to support service composition? (KQ)

RQ4 - How to use a service composition tool in the e-government context? (PQ)

1.5 Research approach

In order to answer the aforementioned questions according to established scientific standards, we adopted a working method. Figure 3 gives an overview of our research approach, in which the project was structured in four distinct phases according to the solution engineering life-cycle proposed in [23].

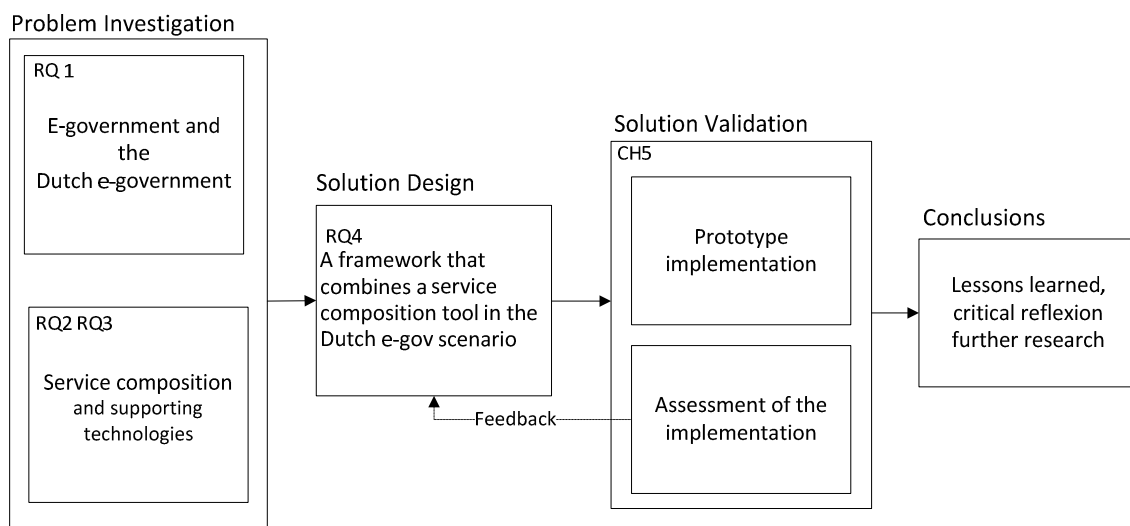


Figure 3: Research model and thesis structure

During the problem investigation phase, we identify the topics that serve as the basis for answering the main research question and perform a literature review for each topic [24-25]. The available resources for this task are depicted in Figure 4.

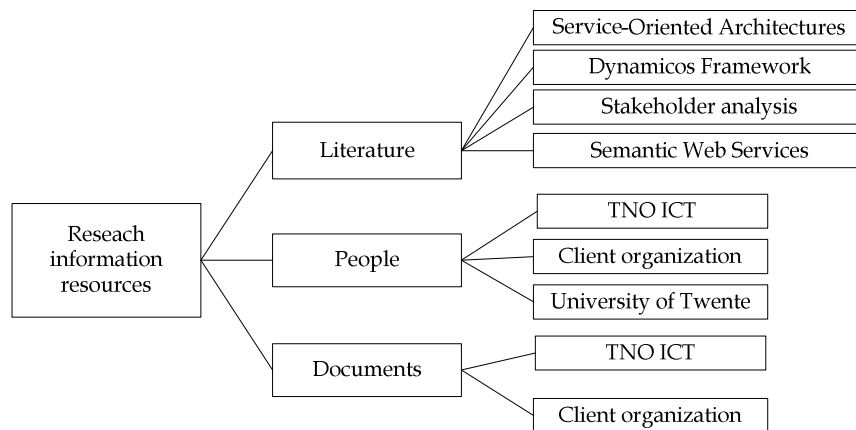


Figure 4: Research information resources. Adapted from [13]

1.6 Thesis structure

This thesis is further structured as follows:

- Chapter 2 investigates the relationship between public services and information technologies, showing the current state-of-affairs of the Dutch e-government and the characteristics of the solution proposed by TNO.
- Chapter 3 introduces the service-oriented architecture as well as the task of service composition and related supporting technologies. At the end, it gives an overview of the state-of-the-art in the field of user-driven service composition.
- Chapter 4 presents our solution framework to attend the requirements of e-government using a dynamic service composition approach.
- Chapter 5 discusses the validation of the solution framework by means of a prototype implementation [26-27]. The application scenario is provided by TNO and the context is the Dutch e-government. The prototype is based as much as possible on already existing software modules. However, some implementation effort was necessary to interconnect the components of the solution. During the validation phase, a feedback loop allows us to return to the solution design phase to fine-tune the suggested method according to the intended results.
- Chapter 6 reflects critically on our work by presenting our conclusions, limitations of our approach, lessons learned, recommendations and directions for future work.

2 E-government

This chapter introduces the concept of e-government, its objectives and challenges. It also investigates the relationship between public services and information technologies, showing the current state-of-affairs of the Dutch e-government.

2.1 Definition e-government

A government can be either a group of people who officially control a country, or a system that comprises mechanisms designed to manage a nation [28]. Our intention here is neither to discuss aspects related to political theory nor insights of a modern state, but rather the role that a government plays in providing services to its population.

Governments world-wide are regarded as bureaucratic institutions that are responsible for the public order and for attending the basic needs of their population. Although there are many different governmental regimes, all of them need to interact with their own population by means of public services. These services are fundamental activities to the development and maintenance of a civil society. Hence, the quality of public services directly affects the well-being of citizens.

As an organization, a government needs to improve its management capabilities not only to supply the demands of the population, but also to improve the country's international competitiveness. In this way, management expertise in the private sector can be useful to shape and improve governmental organizations [29]. However, the relationship between citizen and government goes beyond the conventional commercial trade. Besides having a set of rights that need to be respected, citizens as clients of public services have also several civil duties [29].

Furthermore, it is noticeable that over time governments have constantly adopted new technologies (in a broader sense) to improve their activities [30]. For instance, radio technology has been incorporated and has been used as a powerful instrument for mass communication already for some time by many governments.

With the recent emergence of information technologies and significant electronic business achievements in the private sector, governments have realized that these technologies might potentially increase the effectiveness, efficiency and transparency of the public sector. This marks the beginning of the electronic government, also called "e-government".

From now on we adopt the definition of e-government from [31], which states:

"E-government refers to the use of electronic information and communications technology so as to integrate the customer into the activities of government and the public service".

Because the assimilation of new technologies varies from country to country, e-government initiatives can be found in different development stages [32]. However, they tend to follow

similar evolution paths. Figure 5 depicts the common evolution path of public services, starting from traditional forms of government towards an effective e-government that provides added-value to its citizens.

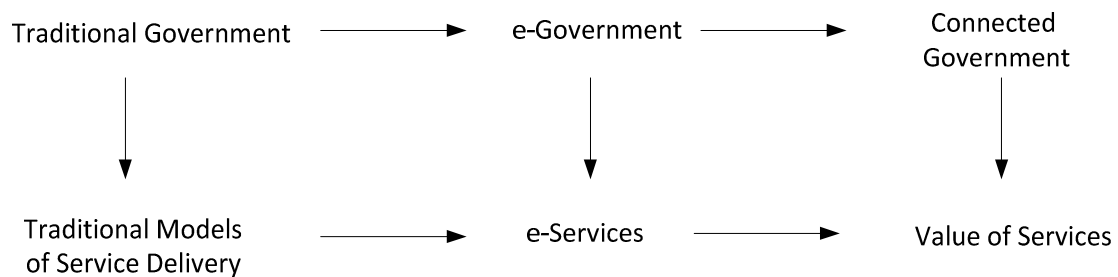


Figure 5: The evolution of public service delivery [32].

2.2 The goal of e-government

From many investigations on citizen values, one clear conclusion is that citizens are more often disillusioned than enthusiastic about their governing institutions [33]. This seems to be related to inevitable comparisons with services provided by the private sector and specially to rapid changes in citizens' values, which became more individualized with the presence of digital technologies [32-34]. Moreover, much of this individualization trend that we see in the contemporary society was already depicted by Debord in his book 'The society of the spectacle' [34]. A question that arises is: how can public agencies meet the more individualized and volatile demands of its citizens?

The answer to this question is the application of digital technologies in the public sector to promote the development of society and to address the elevated expectation of the population [32]. However, in comparison with private organizations, governments have to foster social inclusion, allowing people to exercise their citizenship [35]. For example, e-government facilities are more useful for disabled people than non-disabled people due to mobility constraints [31].

Below we present some other potential benefits of e-government initiatives:

- Improve efficiency, transparency, and quality of public services.
- Reduction of operational costs.
- Improve the image of the public services to foster citizens' trust in public services.
- Improve the interaction between public administration and citizens.
- Increase the participation of citizens in governmental decisions.
- Reduction of procedures that need to be performed by the staff and/or citizens.

2.3 Current challenges

The implementation of comprehensive e-government is a step-by-step process that encompasses several months or even years to be accomplished [36]. In practice, there are still

several gaps between the promises of e-government and the actual results [32]. Among all e-government initiatives, the creation of web portals is an important step to reduce the distance between the government and the citizens. Therefore, since the second half of the 90's, governments were challenged to implement this new channel of communication with citizens [32, 36]. However, many of the governmental portals still reflect the governmental organizational hierarchy, causing two important drawbacks:

1. Information and services are available in a fragmented way, obliging the web clients to figure out which services satisfy their needs and where to find them. This happens because web sites simply remount in an electronic way the structure and logic of regular public services.
2. E-government websites offer public services without taking the perspective of the citizen into account. Although there are some layouts that are well organized in suggestive themes, usually they are more attached to the governmental logic than events that take place within the society itself.

One widely recognized approach to tackle both issues is called "Life events", which suggests a better way to structure electronic public services [37-38]. According to this strategy, a given web portal can better attend its users if it assumes the perspective of the citizens, while shielding them from the complexity of the public service structure. This organization model is based on the main happening in a citizen's life. Some examples of life events are birth registration, school registration, change of address, marriage and death. For each of these events, one or more services might be required. Furthermore, many public services require data interactions amongst different departments and that is where the technological challenge starts.

Multiple services that belong to a given event must be realized seamlessly to the end user. Therefore the access of several public services should be simple, intuitive and fast in a way that the citizen does not need to know how the government deals with the information within its disparate departments. This strategy could potentially also enable for interactions with external organizations that do not comprehend the public administration, but offer services that are necessary for a given event.

Portals that are life-event-oriented can be implemented by simple models based on a well-defined hierarchy of life-event related topics, where each citizen browses the content of these portals in an intuitive manner. More complex models can be based on intelligent systems [38], in which the structure of the information is flexible enough to establish a dialog between the citizen and the government.

2.4 Classification of e-government initiatives

Here we present the classification of electronic public services suggested by [39]. Based on this model we can position the current state-of-affairs of the Dutch e-government and its

envisioned position. Besides that, the utilization of the classification scheme can provide the following outcomes:

- Improve of the understanding of the e-government activities.
- Influence the assignment of priorities in the implementation strategy.
- Allow different e-government to be compared.

Figure 6 shows a bi-dimensional scheme for e-government classification. The horizontal axis refers to the perspective of the citizen, whereas the vertical axis covers the perspective of information technologies capabilities.

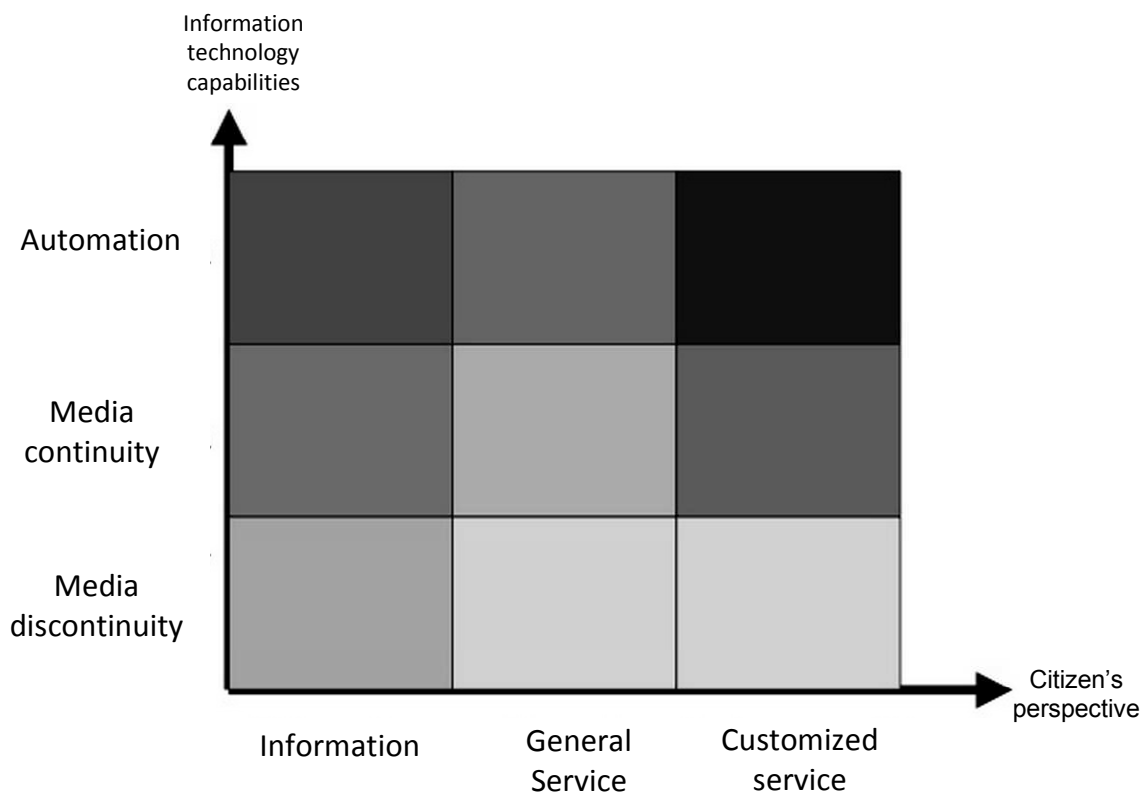


Figure 6: Classification scheme for e-government [39]

From a citizen's perspective, the scheme defines three levels of interaction between an individual and e-government:

- Information: is a broadcasting communication type, where the e-government makes the same information available to every one. However, people can only consume the published information. No identification is required to have access to this information.
- General Service: is a bilateral communication, where services are available to everyone. The difference from the previous type is that it allows for a dialog between

the system and the citizens so that they can solve specific problems. No prior identification and authentication is required.

- Customized Service: is a bilateral communication, where the services are specifically tailored to attend the needs of each specific citizen. This level of personalization requires user authentication.

From a technological perspective, the scheme defines three levels of IT support:

- Media discontinuity: Services are not fully supported by IT resources and different types of media are required.
- Media continuity: Services are fully supported by IT, but human interaction is required.
- Automation: Services are fully supported by IT without human interaction. This applies to administrative procedures which do not require the consideration of individual circumstances.

2.5 Dutch e-government initiative

The application of information technologies in the governmental setting is a common concern in many countries that are attempting to revitalize their public administration to make it more proactive, efficient, transparent and more customer-oriented. The Dutch government acknowledges these goals and has been constantly working to achieve them at municipal, provincial and national level. As a result, the Netherlands is currently ranked as the 5th in world according to the e-government readiness index [32]. Moreover, as well as other nations, the Netherlands initially fostered the exposition of public services on the Internet, but this resulted in an uncontrolled proliferation of public websites from different localities and purposes. This might be attributed to the decentralized character of the Dutch government and the excessive autonomy given to the municipalities [40-41].

To tackle this issue, in 1996, the Dutch government launched a project called Public Counter Project (OL2000), which aims the creation of a national-wide network of public services that can be accessible at one single place, therefore removing the barriers to apply for multiple services and at same time, increasing the likelihood of users to become aware of the existence of various relevant services [40].

Since 2003, the Dutch government has been focused on projects that tackle the data integration and system interoperability towards a more effective and efficient government [13]. Furthermore, with a crescent number of e-government projects taking place in the Netherlands, a high level reference architecture of the Dutch government was designed to help the administration steering all the e-government actions (Figure 7). An important characteristic of this architecture is the application of the service-oriented principles [36] (see chapter 3).

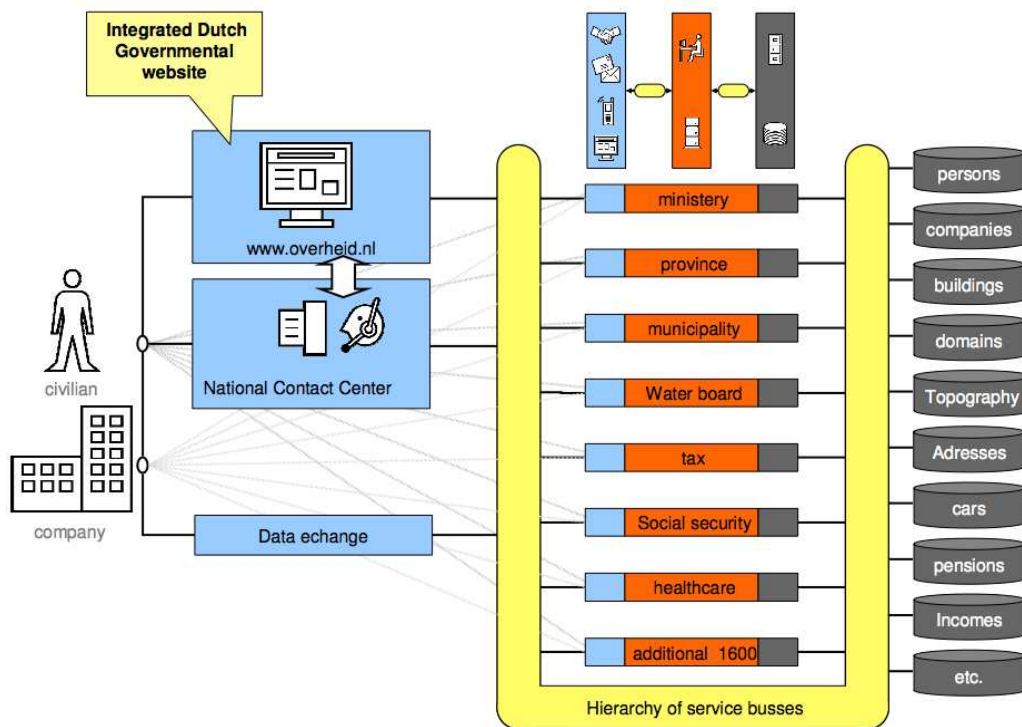


Figure 7: Overview of the Dutch e-government architecture [36].

The utilization of a service-oriented approach in the Dutch government brings several benefits, especially the possibility to tackle the fragmentation of current services. However, in practice there are still important technological challenges ahead to the realization of this architecture.

Figure 8 depicts the evolution of the Dutch e-government strategies, starting in 1990, when the government began with its electronic presence by publishing information about itself and its activities. It is followed by an increasing number of projects to integrate disparate systems and at the same time providing more interactions with the general public.

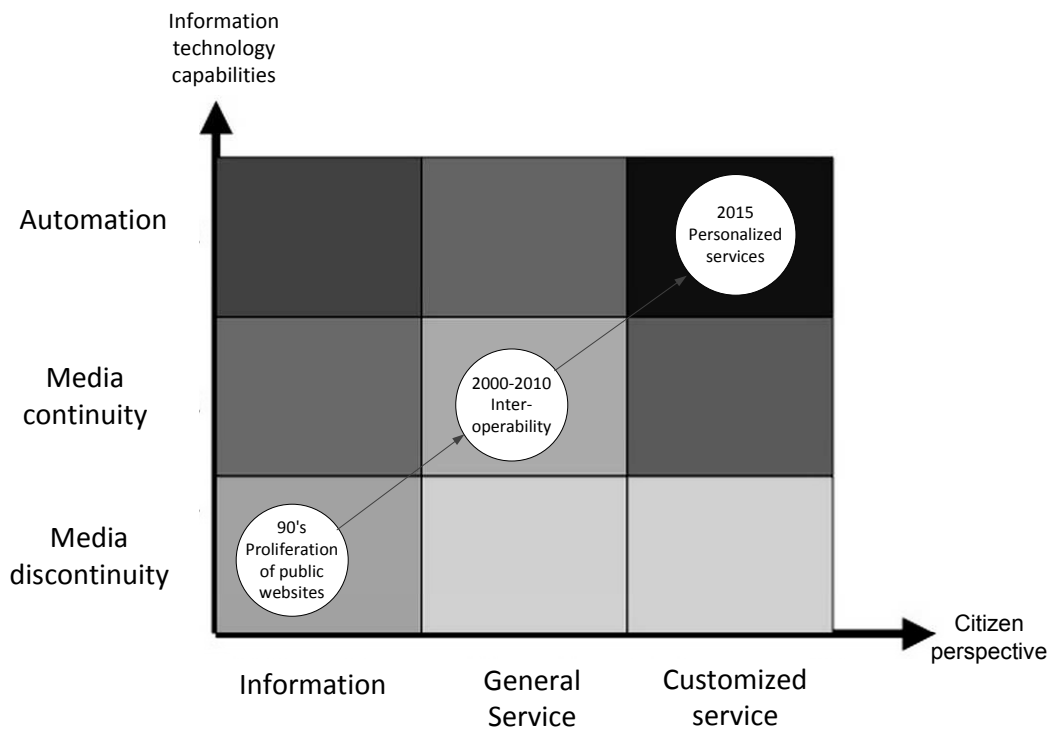


Figure 8: The evolution of the Dutch e-government.

In line with this development path, this research is a step towards the automation and personalization of the public services. Assuming that there are already certain acceptable levels of interoperability between the governmental systems, and the authentication of users is already possible at all levels of the government services. At this stage the system should enable a citizen to perform transactions over the Internet such as applying for social programs or services, signing up for classes and so on.

3 Services and service composition technologies

This chapter introduces the service-oriented architecture as well as the task of service composition and its related supporting technologies. At the end, it gives an overview of the state-of-the-art in customer-driven service compositions.

3.1 Service-oriented architecture

Nowadays, to implement information systems that add value to an organization, it is important to consider their ability to interoperate with other systems. Striving to a high level of *interoperability*, new systems address important requirements such as: flexibility, scalability and compatibility with legacy systems, not to mention that their implementation has to happen in a seamlessly way without compromising current systems. To address these problems, two concepts emerged in the area of distributed systems: Service-Oriented Computing (SOC) and Service-oriented Architecture (SOA)[2].

The authors of [14] claim that “A generation ago, we learned to abstract from hardware, now we are progressing by learning to abstract from software”. This means that we can think about the functionality and services we need, regardless of which program language or operating system is used to perform them.

In fact, after a tremendous growth of interconnectivity caused by the Internet, service-oriented architectures became a popular subject in academic and business worlds. The application of SOA already yields good results in the area of e-commerce and integration of enterprise applications [14]. Nowadays, SOA is becoming a suitable solution to public organizations, as we depicted Figure 7.

The fundamental principle of SOA is the atomization of services and externalization of system functionality, which consequently, open a broad range of possibilities towards integration as well as reusability [12]. In order to expose internal software functionality, companies make use of web services [11]. Which, in turn, have to be built up using standardized protocols that operate above technology, such as: operating systems, programming languages and so forth. However, SOA has to do with the design choices in the specification of a solution and not a set of technologies, whereas web services themselves are responsible to the actual realization/implementation of SOA. Below we present the fundamental characteristics of web services:

- Services: expose system functionality for external use in terms of services.
- Self-describing interfaces: The utilization of services happens through web services interfaces. A web service interface defines a set of public operations that can be performed by the web service regardless of the underlying system platform.
- Message exchange: Operations are defined as a set of messages that describe the data being transferred based on an XML Schema.

- Synchronous and asynchronous communication: The clients of services can exchange message synchronously or asynchronously in two ways on top.
- Low coupling: Services are not dependent on each other; this is possible due to the self-describing interfaces and the support of synchronous and asynchronous communication.
- Service registry: Because many services are available, it is necessary to have a central repository to keep meta-formation about the services to help service clients find a given service.
- Quality of service: Is usually associated with attributes such as reliability, manageability, security. Services have different quality levels.
- Service composition supporting business process: A set of services has to be compound according to business objectives.

To understand service-oriented architecture and later the composition of services we need to have a clear understanding of the term 'service'. Figure 9 shows a service in its simplest form.

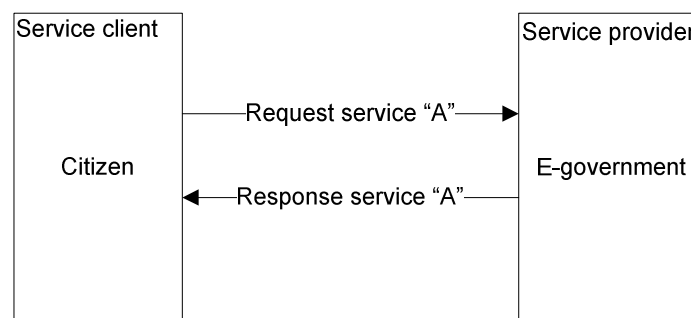


Figure 9: Overview of a simple service request

A service-oriented architecture has basically three roles in order to operate. Service provider, service client, and the service repository [12]. In principle, the citizen finds the provider by the service repository, which is the place where the service providers publish their services. Afterwards, they are able to establish an electronic transaction.

Here, instead of looking at the citizen as the client, in Figure 10 we take the perspective where one department of the government is the client of a service that is provided by another department. For example, a parking authority that provides parking permits needs to identify which authority is responsible for providing information about the driver's license of the citizen. It has to find which department is responsible for what, what are the functions provided and how to realize the service.

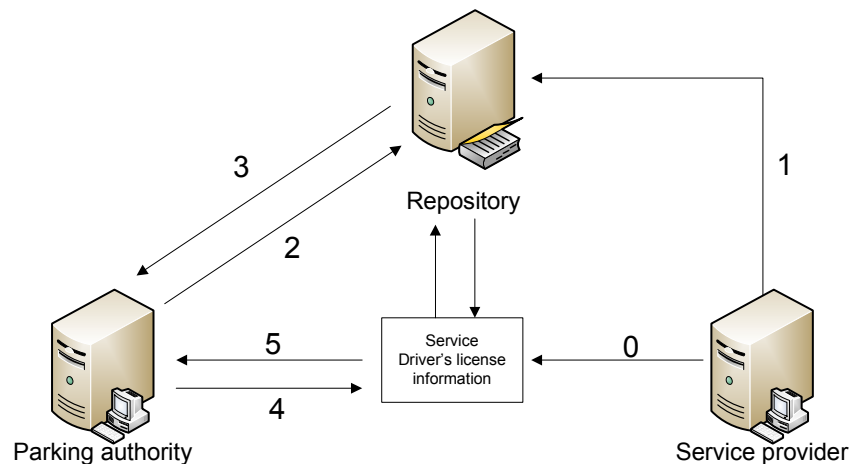


Figure 10: Department interaction based on SOA

In summary, in SOA each service can encapsulate application functionality and exposes it to external access based on stabilized standards. Services are autonomous and heterogeneous in the sense of that they can be executed on different platforms and in different organizations. This might be one of the reasons why SOA is increasingly being used to support many organizations.

3.2 Technologies for SOA

In traditional distributed system architectures, the components of the application are often strongly linked to each other (tightly coupled), making the application itself fragile to the constantly changing business environment. After a wide acceptance of XML as a standard language for flexible definition of communication protocols, web services emerged as a solution with characteristics that better cope with new business dynamics. The core of this architecture proposal is that web services operate in a higher level, and standardized protocols that shields service clients from peculiarities of programming languages and operating systems [11]. Yet, differently from web sites, which are developed to interact with humans, web services are built to allow interactions between computer systems [11].

Through web services, clients can access services that are hosted in heterogeneous systems in a standardized way. To do so, the only information that a client needs to have is the description of the service (what it does, where it is located, how to access it). This is provided by a WSDL (Web Service Description Language).

In order to facilitate the integration of components of systems in this architecture, they need to communicate by means of other standard protocols. The most important ones are the Hypertext Transfer Protocol (HTTP), the Simple Object Access Protocol (SOAP). With exception of HTTP, these protocols are XML-based.

UDDI	Searching
WSDL	Description
SOAP	Communication
HTTP	Transport

Figure 11: web service protocol layer [12]

Messages are encapsulated in a SOAP envelope and then delivered to their destination on the network, mostly using the HTTP protocol. This possibility of using HTTP to exchange information with external systems is an important factor to the acceptance of web services in because it avoids time-consuming firewall configurations [12].

3.3 Business processes

A business process can be defined as a set of activities that are combined to achieve a certain business goal [2]. Since any organization needs to interact with other entities to operate in the business environment, they need to carry out a set of business process to deal with other entities such as suppliers, clients, government agencies, and employees occurs by means of business processes.

Since the industrial revolution, business processes have been designed to carry out mass production of goods and remained static for several years. However, in the contemporary world, globalization forces and the availability of new technologies have forced organizations to deliver more personalized services, which results in more dynamic and flexible business processes. Understanding, designing and constantly optimizing business processes is becoming a vital activity to organizations.

By using SOA architectural principles, business processes can be improved in several ways because it enables high levels of systems interoperability, hence businesses processes can have a greater myriad of business partners no matter their actual geographical position.

3.4 Service composition

The creation and execution of business processes are strongly related to composition of services because a service composition is done to support business process. The service composition is the action of concatenating separated services with complementary functionality to provide end-users with higher added-value services [2, 11-12].

Apart from the creation of more added-value services, the composition is also a topic of growing concern because it can be used to integrate applications not only internally, but also externally with business partners. This integration reduces the fragmentation of information, which is one of the main obstacles faced by e-government initiatives.

In order to carry out service composition, some aspects need to be taken in to consideration. Firstly, because a single web service might not fulfill the client's requirements, a business process has to be as personalized as possible to meet individual requirement. Secondly, user's

goals and preferences tend to be informal and subjective, which are sometimes difficult to represent. Thirdly, legal aspects of the relationship between organizations and departments might be an important constraint.

Below we present some basic types of service composition, which is the snippet of a composed process. In Figure 12, web service B starts only after web service A is finished, in Figure 13 both web services are executed at the same time (in parallel), whereas in Figure 14, there is a decision point that defines the execution of only one service.

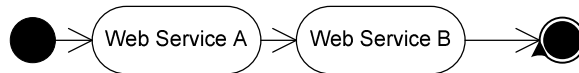


Figure 12: Sequential composition of web services

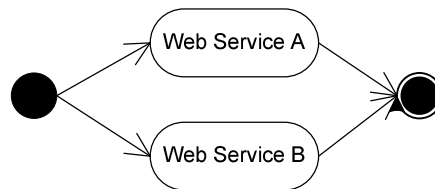


Figure 13: Parallel composition of web services

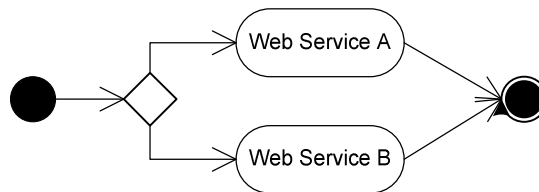


Figure 14: Web service execution after a decision point

The complexity of the service composition is strongly related to the intricacy of the business process itself and the inputs and outputs of each service. Therefore, defining business process can be a complex undertaking.

Figure 15 portrays the business process that was introduced in Chapter 1. In this case there are six departments and each of them is responsible for one service. The process only finishes after all the activities that are executing in parallel finish.

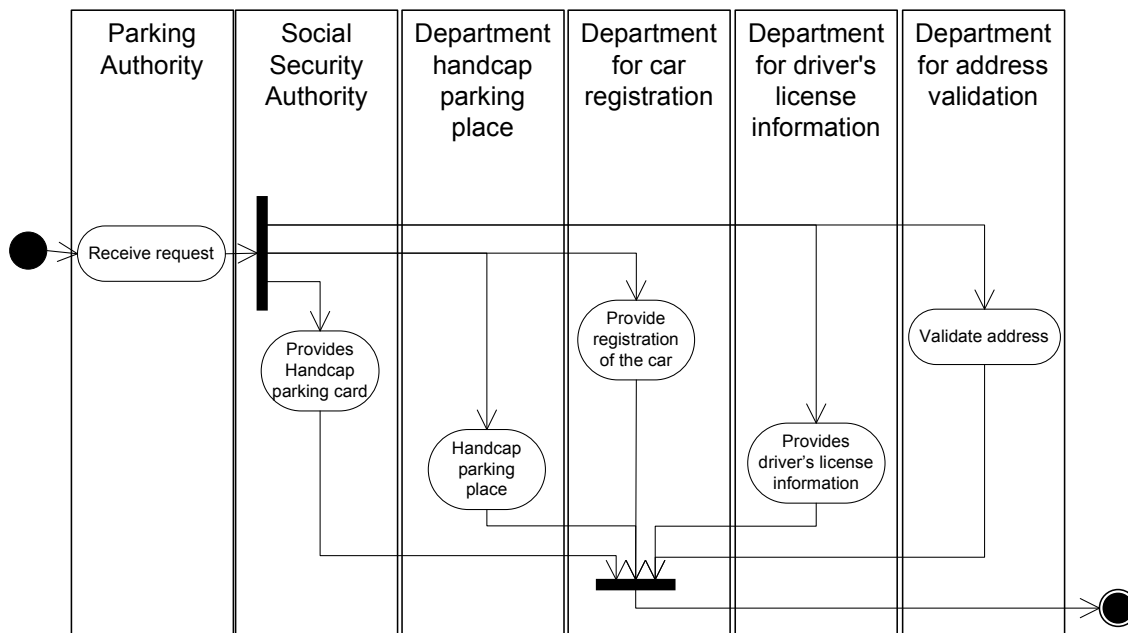


Figure 15: Ordering handicap parking permit (UML activity diagram).

The process flow in Figure 15 is human-readable. For example, we can understand where the process starts, its main activities and finishes. BPEL [42] was developed to make this specification computer-readable.

BPEL is a standard language for business process orchestration based on web services and it can be used for the definition and execution business process [11]. BPEL permits to concatenate web services allowing operations that can be synchronous or asynchronous. Furthermore, activities can be executed sequentially or in parallel.

Below we list some other important capabilities of the BPEL language:

- Describe the business logic through composition of services.
- Tackle invocations of operations either synchronously or asynchronously.
- Invoke activities in sequence or in parallel.
- Provide error handling mechanism at runtime.
- Restart a composition that was interrupted or presented errors.
- Orchestrate the order in which the activities are executed.

Figure 16 shows the business process of the handicap parking permit from the perspective of BPEL. This perspective is more concrete than the UML activity diagram in the sense that it treats the BPEL composition as web service that interacts not only with the citizen, but also with other departments. Moreover, the internal square comprises the activities that are executed in parallel.

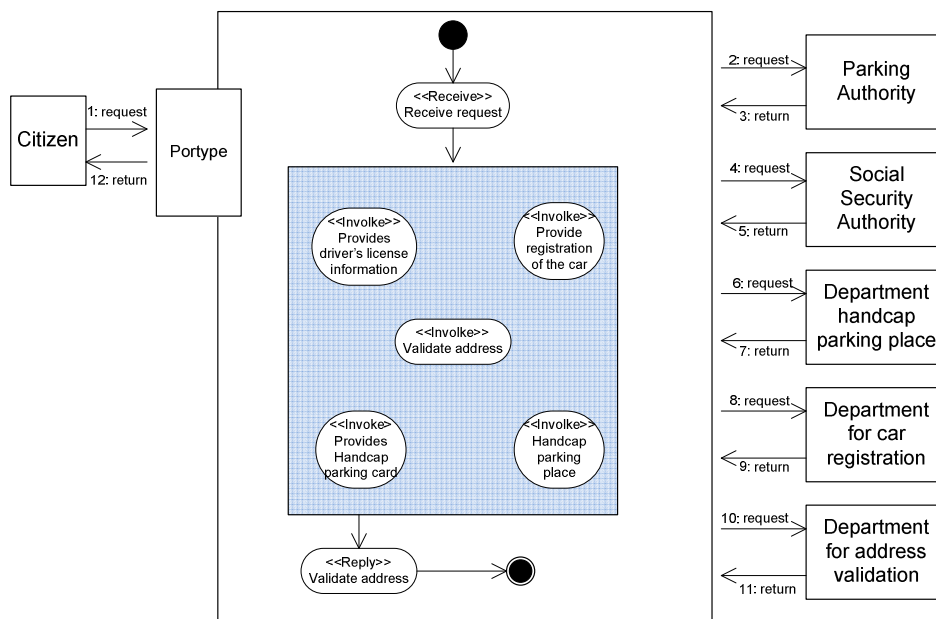


Figure 16: BPEL process composition of the ordering handicap parking permit.

There are already several tools to help users compose web services using BPEL. Some examples are ActiveVOS [42], EclipseBPEL [43] and Oracle BPEL [44]. However, manual service composition still be complex and time-consuming, which demands automatic or semi-automatic approaches.

3.5 Dynamic service composition

The SOA approach and its related technologies can leverage numerous benefits to organizations. However, there are some obstacles for its realization. For instance, the composition of web services is still a complex and time consuming activity for reasons such as:

- The subjectivity of the end-users' goals.
- Technical complexity while creating and executing compositions.
- Proliferation of web services gives several new possibilities.
- Constant updates of composite services.

The fact that the basic web service standards such as WSDL and SOAP process only syntactical functions, obliges service designers to interpret the meaning of the users' requirements (what they want) against to the descriptions of services (what the service is able to deliver) and this task is usually complex and time consuming, requiring a great deal of technical knowledge. Therefore end-users are not able to compose a service by their own; they rely on professionals that create the process for them.

To tackle these challenges, some authors suggest a user-centric composition approach, which allows end-users to play the central role in the service composition process [45]. However, to achieve this vision, two major points of concern need to be addressed.

1. The user interface of the service composition system needs to be simple and intuitive to guide the end-user to compose a service according to their needs.
2. The complexity of interconnecting web services should be hidden in the background by a service composition engine.

According to recent studies, ontologies can help the service composition engines establish conceptual links between user goals and service functionalities. The conceptual links can be performed by systems that reason on the semantic of the data. Because of this, ontologies became one of the key components of automated composition frameworks [46].

Before selecting a composition framework for a given context it is necessary to have a clear understanding of how services can be composed. In this way, when composing a service in an automatic way the engine can be oriented to go back or forward to find complementary services.

Figure 17 shows a distinction between a backward and forward chaining of services, where a backward strategy takes the “Service 1” as the starting point of the interaction and based on the requirement of this service. Then the composition engine searches operations outputs that correspond to it. In contrast, the forward method analyses the output of “Service 1” and searches for related services that could be suitable for the final composition.

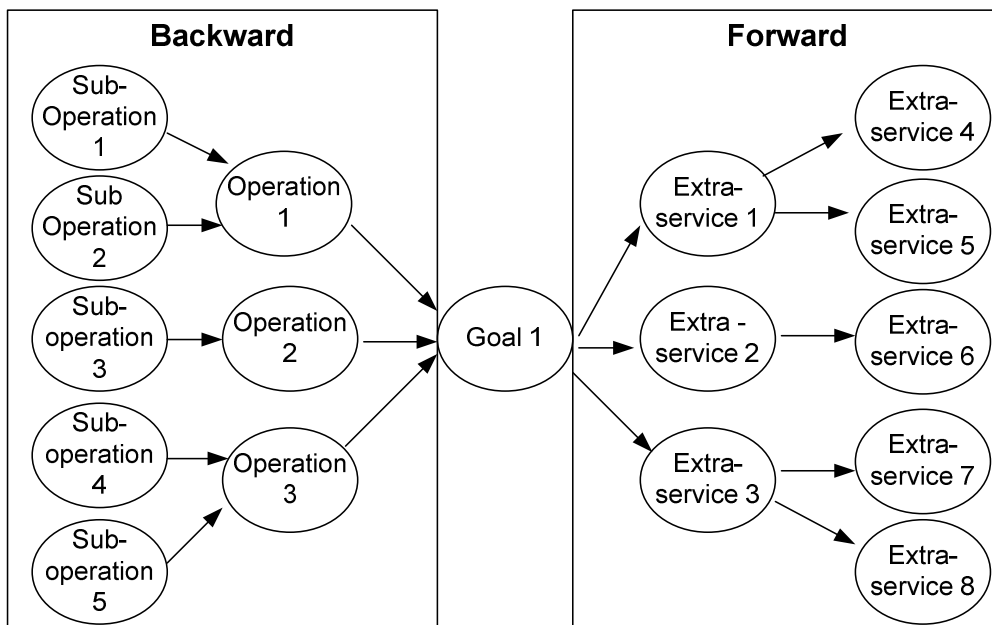


Figure 17: Backward vs. forward chaining of services

3.6 Dynamic service composition approaches

Currently, several research initiatives tackle user-centric dynamic service composition [18, 47-49]. In a survey on these approaches the authors of [49] suggest the following categorization:

- **Workflow-based** – It is mostly used when a process model is already defined and the composition engine works to find out web services and concatenate them according to the

end-user's requirement. In other words, it can be viewed as a dynamic workflow, where the workflow is set up and the web services that support it are located and executed on the fly. Yet, in this approach, there is a clear centralization of the composition process since the composition engine orchestrates the entire process.

- **AI planning** – This method is used when the user enters a set goals and constraints, but the process model is not defined. So, the process model is generated automatically by the system. In contrast to the workflow-based strategy, this approach is based on decentralized composition, where web services are regarded as autonomous entities communicating with each other to identify the best course of action to achieve the goals of the user. Instead of orchestration of services, AI planning moves the focus to the choreography of web services.

While a detailed discussion about various service composition frameworks is given in [20], here we briefly discuss only three approaches that support service composition at runtime. This characteristic would be potentially valuable to the next generation e-government solution therefore it was the determinant selection criterion. We also briefly discuss the DynamiCoS framework that can supports runtime service composition, although it was not listed in the survey.

3.6.1 WSMF

The Web Service Modeling Framework (WSMF) focuses on bringing e-commerce to its full potential through peer-2-peer interaction (choreographies), by allowing a strong decoupling of services and strong service mediation [50]. Based on the AI planning principles, the framework uses ontologies, goal repository, web services semantic descriptions and mediators to overcome the interoperability issues and define the process workflow on the fly [50]. The WSMF framework includes: web service Execution Environment (WSMX), Web Services Modeling Ontology (WSMO) and the Web Service Modeling Language (WSML)[20].

In 2009, TNO attempted to apply WSMF in the e-government context with the project so called Service Engineering Workbench (SEW). However, the project experienced several difficulties during its implementation because the WSMF documentation presented several gaps that were crucial to the continuation of the project. According to staff members who were actively involved in the SEW project, the WSMF stressed a great deal of possibilities, but it lacked supportive guidelines to achieve them. For instance, the backward chaining of services was not specified [8]. Furthermore, examples in the documentation did not include third party case studies to support different phases of the project. As a result, the SEW project did not achieve satisfactory results.

3.6.2 METEOR-S

METEOR-S (Managing End-to-End Open Rations for Semantic web services) approach was originated from the idea of merging large-scale transactional workflows technologies with semantic web services [48, 51]. The focus of this framework is to improve the capabilities of the already functional web services technologies and process design by means of semantic

matching [51]. The framework is characterized as a workflow based strategy, thus it fosters a centralized coordination (orchestration) of services. According to [52] this framework allows:

- Rapid process compositions using automatic and semi-automatic service discovery;
- Process templates configuration and reutilization;
- Flexible process design, where changes in the partner interface do not change the semantics of the interface, therefore try neither to affect the process template nor the discovery of services.
- Ready-made templates can act as business/reference models and can be re-used by different organizations that want to implement the same process with different services.

This approach is suitable for the e-government setting since the public services workflows can be improved with semantic description to match semantic described services. Moreover, in the METEO-S project website [53] we found graphical tools for editing the semantic description of the services. This indicates a certain maturity of the solution and would facilitate the implementation process especially with regard to the access of business experts to the framework. Furthermore, during the period of our research we could not have access to relevant case studies see the actual implementation of the framework.

3.6.3 DynamiCoS

DynamiCoS is an experimental framework developed at the University of Twente in 2007 and aims at supporting user-centric service composition. DynamiCoS has similarities with METEOR-S, but it basically aims at providing on demand runtime service composition [17, 21]. It also supports back and forward service compositions.

The main reasons for selecting the DynamiCoS as the framework for e-government initiatives are:

- Government processes present a workflow-based structure, which DynamiCoS supports.
- The e-government aims at providing citizens with runtime service compositions that require minimal human interaction at the public facilities, which DynamiCoS supports in its latest version.
- The backward composition of web services is fundamental to support citizens while resolving services that have several preconditions. This is a typical situation for government processes and is supported by DynamiCoS;
- The unsuccessful experience of TNO with WSMF framework suggests that a new approach should be tried, one that suits the characteristics of the public services;
- Difficulties to reach WSMF and METEOR-S relevant documentation such as case studies or platform implementations;
- Direct contact with the developers of DynamiCoS could result in a faster exchange of information, reduced learning curve, easily access to the code, faster prototype development and feedback loops.

- Users of e-government services have different understanding about governmental processes. DynamiCoS offers the flexibility necessary to create different types of support with the same base systems. Also, adaptations to DynamiCoS can be implemented quite rapidly because of the direct contact with the DynamiCoS developers.

4 Technologies for enhancing e-government experience

This chapter presents the main components, functions and technologies of our solution framework for enhancing public service experience. Initially it gives an overview of the solution framework and then goes in more details of its most important components (PSE-TNO and DynamiCoS). This also characterizes an *interaction pattern* for end-users that is designed to interconnect the front and back-end systems and is suitable way for the e-government context.

4.1 Solution framework overview

To improve the e-government service experience, the new generation of e-government portals need to provide personalized and easily accessible services. For personalized services, it is necessary to set up a profile-based system that allows to retrieve users' data according to the context of the service. In our approach, we assume that the citizen is already authenticated and the system already possesses some profile information about the citizen.

With regard to the accessibility of e-government services, we suggest the utilization of natural language search mechanisms to help users finding services that fulfill their goals (Life-events). In this way, an e-government portal can provide a single point of access to all electronic public services without obliging the end-user to visit several URLs. We assume that the e-government systems are already fully interoperable based on the principles of SOA (see Chapter 3).

Figure 18 shows an overview of our solution framework, divided in a front and back-end. In essence, our solution combines several types of technologies, however the most important ones are the user-interface and natural language recognition of TNO PSE [8], and DynamiCoS [17] as the service composition engine.

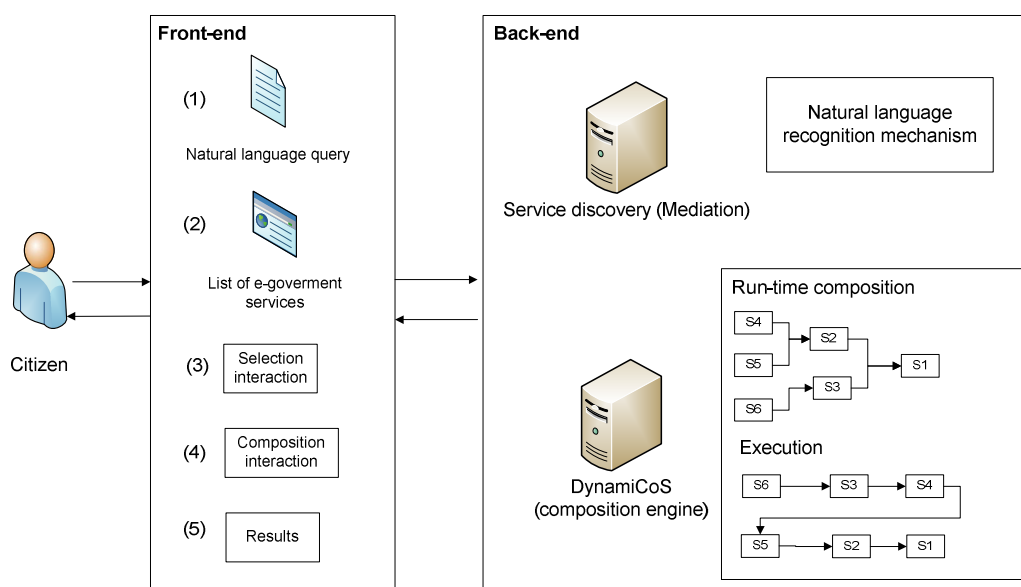


Figure 18: Solution framework overview.

We describe below the user's interaction with our solution framework:

1. **Natural language query** - According to the "life event" strategy [37-38], events that happen in real life need to be updated in the government databases. Therefore, the citizen has to find in the web site which service corresponds to his/her life event. Commonly, public organizations make available a hierarchical list with services they are able to perform. This tree structure of services can however become very complex due to the sheer number of services, and may be difficult to keep it up-to-date, especially in a nation-wide e-government scenario. In our framework we adapt the natural language search mechanism provided by TNO to index the information of e-government services. This allows us to match natural language queries to services.
2. **List of e-government services** – After a query is performed, the citizen receives a list of e-government services that relate to his/her goals (life events).
3. **Selection interaction** – After the services are displayed, the citizen selects a service from the list or repeats the search until an appropriate service is found.
4. **Composition interaction** – Depending on the complexity of the chosen service(s), it (they) might require one or more additional services to accomplish the goal of the citizen. Hence, these services need to be automatically discovered and orchestrated by a service composition engine. In our solution we apply a backward service composition strategy. In this strategy, the composition engine starts a dialog with the user and if additional information is needed to execute the selected service(s). In case relevant information is available in the citizen's profile, the engine uses this information automatically or makes an input suggestion to the user. The distinction between backward and forward chaining of services is discussed in Section 3.5. This composition interaction finishes whenever the services have all the required inputs for execution.
5. **Results** – The composition engine should return the executions of the services in the correct sequence that is necessary to achieve the first selected service. The execution it self has to be carried out by the front-end. The execution of the service is a transparent process to the end-user, which receives only the result of the execution.

4.2 Public Service Experience - TNO

TNO is a Dutch knowledge organization that is constantly involved with innovation and application of knowledge to the well-being of the Dutch population [54]. The department of information and communication technology (ICT) is responsible for developing IT solutions that are applicable to the Dutch society. Below, we briefly introduce the concept of the TNO Public Service Experience (PSE) project.

4.2.1 Overview

Inspired by the current challenges of the Dutch e-government, TNO has developed a dynamic public service discovery and mediation solution called “Public Service Experience” (PSE) that helps citizens identify which public services are suitable to fulfill their life events.

Figure 19 presents the conceptual model of this solution, which is a combination of a process model with an information model [8-9].

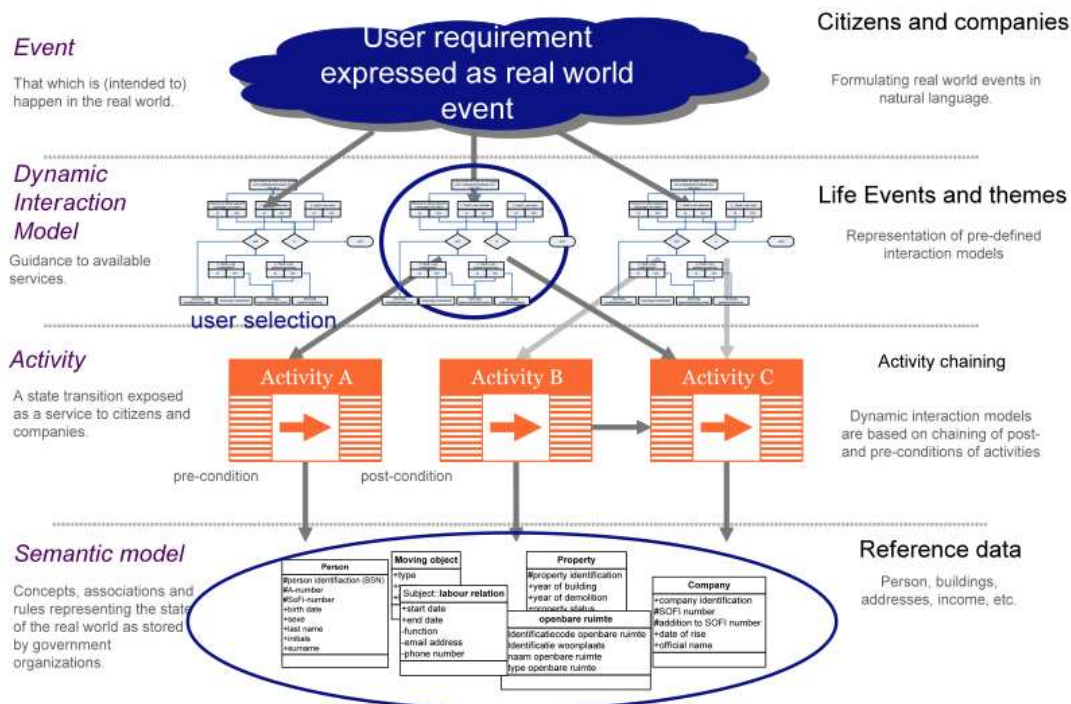


Figure 19: Conceptual model for service discovery and mediation of PSE [8]

Initially, the interaction with the citizen is performed through a central web site where he/she can describe his/her goals in natural language. After that the citizen follows a dynamic interaction with the application in order to identify the appropriate public services and execute them without the intervention of a public servant.

Some characteristics of the PSE solution are listed below:

- The decision tree used to identify the services is generated automatically.
- The ranking algorithm learns from users with similar goals to show the best matching services.
- The tool is built upon existing standards already in use by the Dutch government.
- All public services should be semantically described.

- The model makes no statement about the interdependence of services. As activities are related (chaining), this implies that a selected service also depends on the result of another service.

So far the PSE prototype (Figure 20) is able to receive a list of requirements, establish a dialog with the citizen to refine his/her actual goals (via an XML based interaction) and return a list of corresponding services (based on conditions and preconditions).

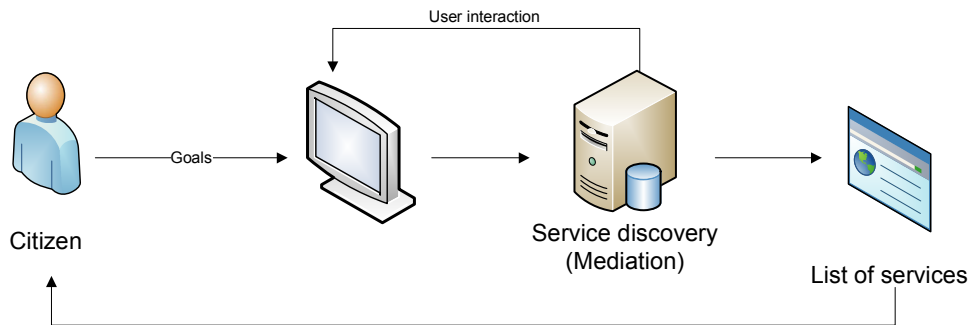


Figure 20: Current solution developed by TNO.

4.2.2 User Interface

In Chapter 2, we have mentioned the explosive growth of the Internet caused a proliferation of isolated e-government web sites, obliging citizens to visit several URLs to achieve their goals. Therefore it is necessary to establish a single point of access by means of, for example, a portal that supports all citizens' life events. However, one of the main obstacles to attain this vision is the lack of interoperability in the back-end systems, which is a recurring topic in the Dutch e-government agenda [36, 40].

Assuming that the integration of the Dutch e-government services is already well on its way, of a single portal that guides citizens through the e-government services is becoming feasible. Therefore, our user interface is partially based on [8], which takes into account the life events principles.

Figure 21 shows the layout of our user interface, which keeps the state of the session (progress of the user) and supports the interactions we suggested on Figure 18:

- Stage 1
 - Allow the citizen to perform the query.
 - Send the query to the composition engine.
- Stage 2
 - Connect with the composition engine to get the query results.
 - Format and display the result of the query.
 - Show the natural language description of each service.
 - Suggest the execution of another query if no services were found.
- Stage 3

- Perform a dynamic composition interaction based on the characteristics of the service that was selected in Stage 2.
- Request user intervention when necessary, allowing him/her to enter the inputs for the composition.
- Stage 4
 - Return the result of the composition.
 - Select the result and store it in user's context.

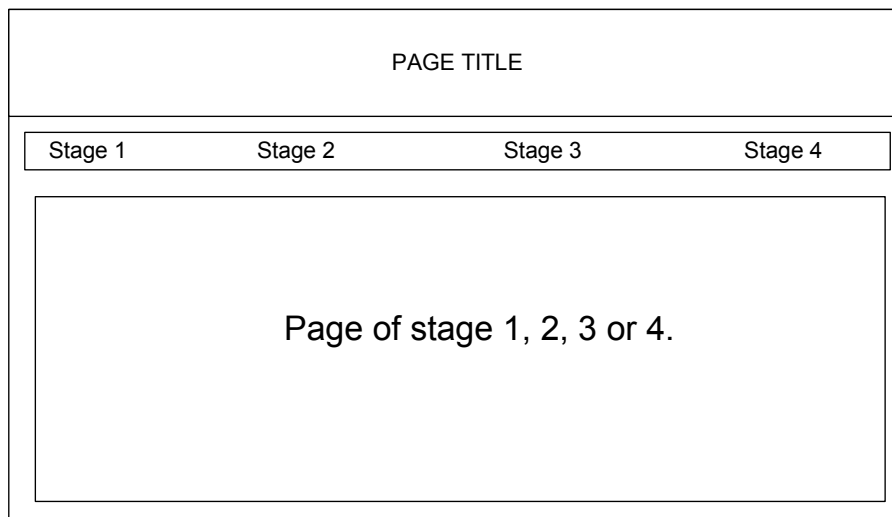


Figure 21: User-interface layout.

4.2.3 Natural language recognition

The PSE natural language recognition is based on the Apache Lucene Java library, which offers full text indexing and searching capability to applications [55]. The main feature of this mechanism is a ranking of terms that improves itself over time according to the usage of the system.

For our solution framework, we adapted the PSE function in two important ways as follows:

1. Adapted the search algorithm to support English queries. The database with public services descriptions that are used to generate the index was translated from Dutch to English. This corresponds to 440 XML files, which each file correspond to the description of a single public service.
2. Encapsulated the Java code into a web service, instead of Java Servlet.

To demonstrate the searching capability, we used the Java Luke [56] user interface to perform a textual query on the index of the XML files (see Figure 22). The natural language query we executed for this was "I want a Dutch working Permit". As result, the system returns a ranked list of 141 XML files with service descriptions. We expect that this mechanism can bring more meaningful results by the time the descriptions are written by civil servants with a great deal of domain knowledge.

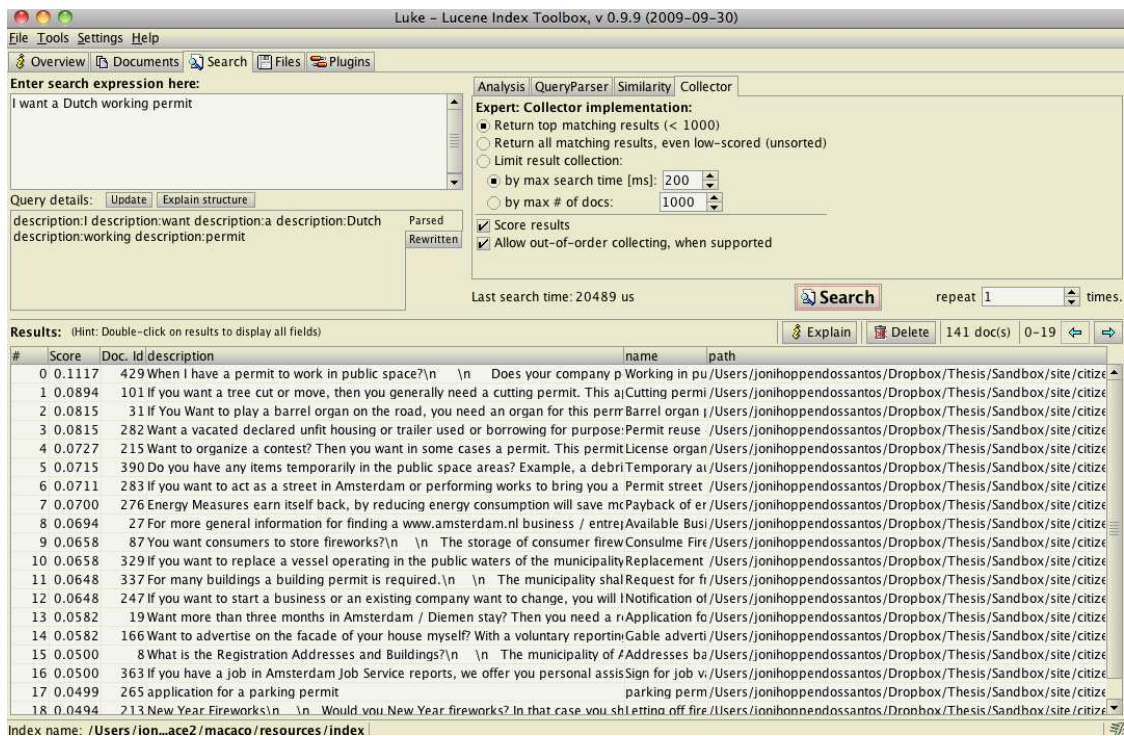


Figure 22: Natural language query demonstration

4.2.4 Supporting technologies

Table 1 summarizes the PSE technologies that we incorporated into the front-end of our solution framework. The front-end encompasses to the user interface (UI) and most importantly to the algorithms that manipulate the information that is exchanged with the back-end.

Table 1: Front-end supporting technologies

<p>HTML - HyperText Markup Language</p>	<p>HTML is the predominant mark-up language on the Internet, and provides the way to structure and present the content of the a web site [57]. The user interface consists of HTML pages.</p>
<p>XML (eXtensible Markup Language)</p>	<p>XML language is used to structure data and has been extensively used for data exchange [58]. The dynamic service composition front-end uses XML files to interact with service composition back-end.</p>

PHP - Hypertext Preprocessor	<p>PHP is widely used scripting language that was originally designed to produce dynamic web pages by embedding PHP code into web pages HTML source [59].</p> <p>We utilize PHP functions to manipulate the data and HTML code to establish a dynamic interaction between the citizen and the composition engine. The connection with the back-end is performed through PHP-SOAP functions.</p>
DOM - Document Object Model	<p>DOM is a cross-platform and language-independent convention for representing and interacting with objects in XML documents [60].</p> <p>DOM is used to manipulate the XML files that are used to communicate with the back-end. All interactions are XML-based.</p>
JAVA script	<p>JavaScript is a scripting based program language used for the adding functionality on dynamic web pages [61].</p>
CSS - Cascading Style Sheets	<p>CSS allows the separation of content a from document layout [62].</p>
Apache HTTP Server	<p>Apache HTTP Server developed by the Apache foundation [63]. This is the web server of choice to carry out the prototype experimentations; the PHP module is included in order to execute the PHP tags.</p>

4.3 DynamiCoS framework

The back-end of our solution is where the service composition takes place and the part of the solution that gives the necessary dynamicity to the e-government processes. This is supported by DynamiCoS, which plays the central role in the service composition process. This section presents the main characteristics of the DynamiCoS framework to better understand how we have used it in our project.

4.3.1 Overview

The DynamiCoS is a framework designed to support user-centric service compositions [17, 21, 64-66]. Thus, it supports users in the composition of services according to their specific goals. The framework is graph-based, where the graph represents the service composition execution model. The execution model is centralized and represents the orchestration of the composed web service.

DynamiCoS allows backward as well as forward service composition, linking atomic components (services) according to their semantic similarity of the services inputs and outputs. The services are interpreted and manipulated by the following attributes: Inputs (I), Outputs (O), Goals (G), Preconditions (P), Effects (E) and Non-functional requirements (NF).

4.3.2 DynamiCoS architecture

Figure 23 represents the DynamiCoS architecture, which is an extensible and adaptive framework to support user-centric service composition. The framework has three main components:

1. **User Interface (UI):** It allows the framework to collect requirements from the user, by defining a given *Interaction Pattern*, which has been constructed using combinations of *Interaction Types*.
2. **Coordinator:** It is responsible for interpreting calls (*Interaction Types*) from the UI and mapping them onto specific coordination supporting strategies, which are realized based on the components of the composition framework.
3. **Composition Framework:** It contains all the basic components associated with the aspects of the service composition life-cycle, such as: discovery, composition, management, execution and execution context.

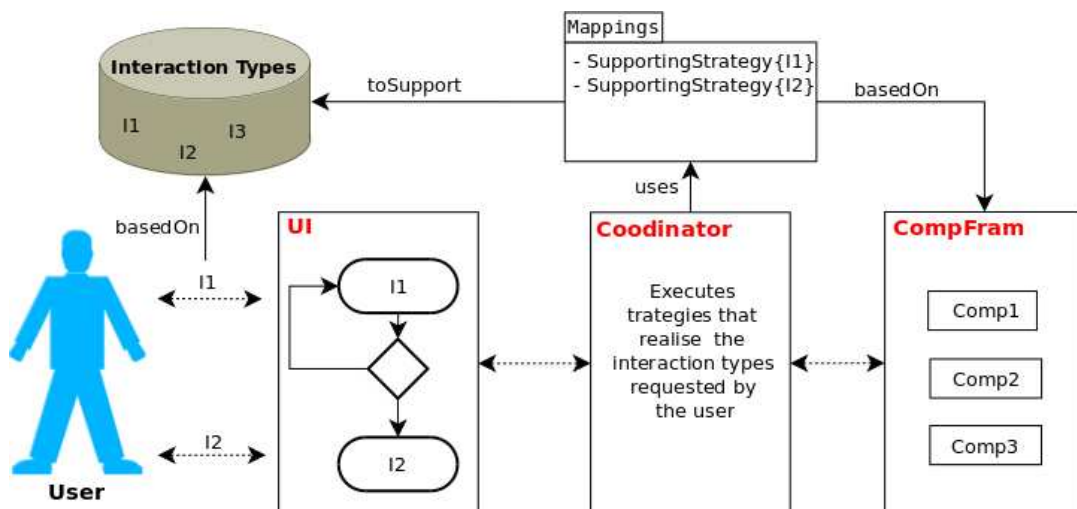


Figure 23: DynamiCoS framework architecture.

4.3.3 Interaction types

In DynamiCoS framework, *interaction types* play a pivotal role to make the composition process flexible. An *Interaction type* represents an action taken by a user in the UI that triggers a function in the composition engine.

In a broader perspective, an *Interaction type* gives guidelines to specify interaction points within the UI, indicating its supported capabilities, inputs, outputs and preconditions that have to be met before an *interaction type* can be activated.

Interactions are interpreted the DynamiCoS *Coordinator*, which is depicted in Figure 24. The left box indicates the URL of the Coordinator web service, whereas the other show the input and output parameters of the service and their corresponding data types.

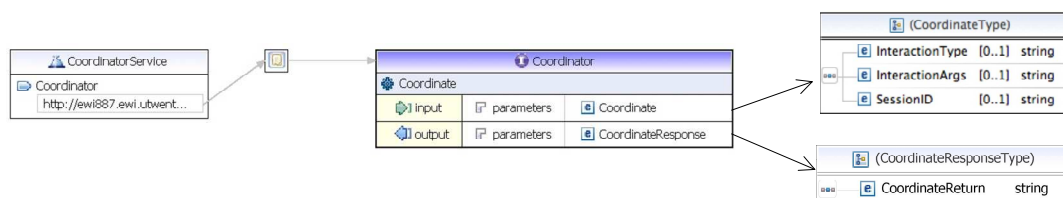


Figure 24: DynamiCoS WSDL visualization.

The *Coordinator* has an operation called “Coordinate”. The input message of this operation contains the *InteractionType*, *InteractionArgs*, and *SessionID*, whereas the output (message) has a *CoordinateReturn*. All requests and responses are string values that contain XML structures defined according to the chosen *interaction type XSD*. Table 2 presents the *interaction types* that are currently supported by DynamiCoS and used in our project.

Table 2: Interaction types

Name	ServTypeDiscovery
Description	Discovers services based on a service type (semantic concept on the goal ontology).
Interface	In: List {ServTyp Out: List{ServType, List{GenServInfo}}
Dependencies	ServType has to refer to a concept from the goal ontology of the framework; interaction types
Name	ServInfoDiscovery
Description	Discovers services based on high level description of the service, e.g.: in natural language.
Interface	In: ServRequest Out: List{ServType, List{Servs}}
Dependencies	-
Name	SeleServ
Description	Selects a service from a set of services.
Interface	In: List{Servs}

	Out: Lis{DetailedServInfo}
Dependencies	Services have to be discovered beforehand, by executing: <ul style="list-style-type: none"> • ServInforDiscovery or • ServTypeDiscovery.
Name	ResolveServ
Description	Finds services that can provide outputs to match inputs/preconditions of a service previously selected by the user.
Interface	In: List{Serv, InpPreToResolve}
	Out: List{GenServInfo }
Dependencies	A service has been selected before.
Name	ValidateInput
Description	Checks the validity of the user's inputs.
Interface	In: List{InputsToValidate}
	Out: List{InvalidInputs} or the validation of all inputs.
Dependencies	The service has been selected previously.
Name	ExecServ
Description	Executes a service, or service composition, previously selected by the user.
Interface	In: (Empty)
	Out: List{ServsToExecNext}
Dependencies	The service to be executed has to been discovered and selected before-hand.
Name	AddInfoToContext
Description	The result of the execution is saved in the execution context.
Interface	In: {ServOutPut, UserContextInfo}
	Out: -
Dependencies	-

The use of *interaction types* to define commands corresponds to a design pattern suggested by [67], where the request is encapsulated into an object, leaving the parameterization or the

request to the client. The main advantage of this pattern is the simplification of the prototype phase in which all functions can be invoked in the same way. The disadvantage is that developers need to inform the parameterization of object to all clients. Furthermore, this solution is not type-safe, since a client might issue commands that are not understood by the server.

4.3.4 Interaction pattern

By exposing its functions through the *Coordinator*, the composition framework becomes a black box that receives and returns messages on demand, while hiding its implementation details. Because of this loosely coupled style of interaction, a strategy has to be defined to align the context of the application and the functions of the framework. In other words, it is necessary to define a sequence of interactions, which we define as an *Interaction Pattern*.

The *interaction patterns* are defined by the UI designer. These *interaction patterns* should be designed to fulfil the requirements of the targeted set of users.

An *interaction pattern* can be created by setting up a set of *interaction types controlled* by some operators (control flow), for example, (if, while loop, and for loop) that shape the supporting workflow. In this way, an *interaction pattern* may contain several alternative execution paths, which offers different properties. Section 4.4 we characterizes two types of users for an e-government system, followed by their corresponding *interaction pattern*.

4.3.5 Supporting technologies

The DynamiCoS framework has been developed by combining various technologies that make it flexible and scalable to operate in different scenarios. For instance, although the language used in DynamiCoS for describing services is SPATEL [21], other languages could be used for this purpose. Moreover, the core of the framework is based on Java components that can run on different platforms. In Figure 25, the DynamiCoS life-cycle is extended with a set of technologies that were successfully used in a prototype experiment carried out in [45]. The most important elements of DynamiCoS is briefly discussed below.

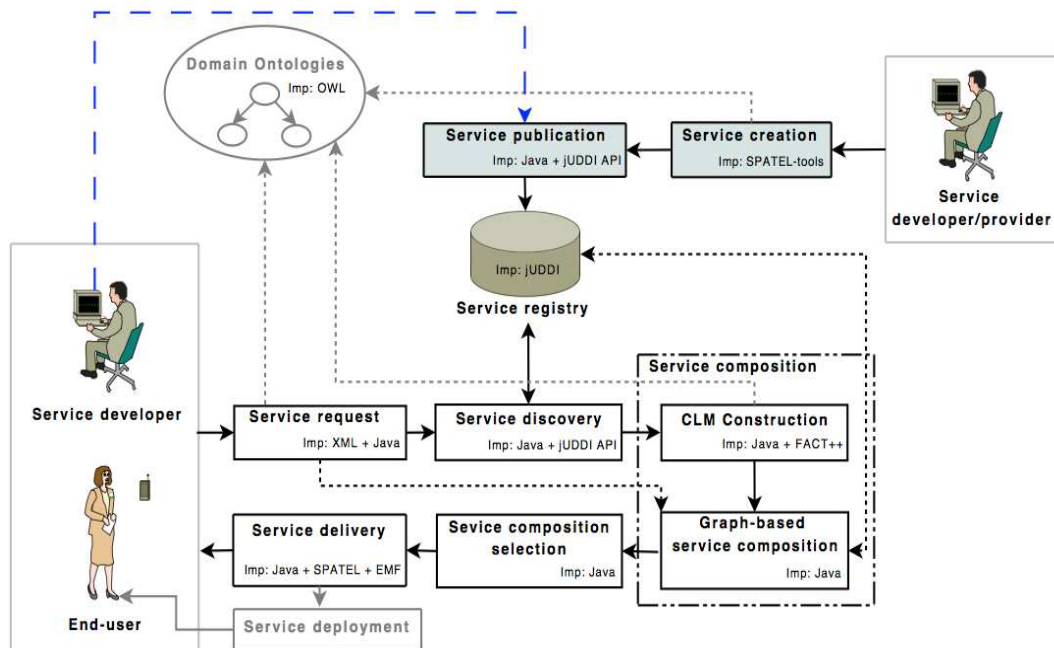


Figure 25: DynamiCoS and current supporting technologies [45].

Ontologies

An ontology is an “explicit specification of a shared conceptualization” [68]. These conceptual specifications are used in a wide variety of fields. For instance, they have been applied in medical diagnostic knowledge management systems to describe protein structures and genes sequences [69]. Yet, they serve to identify misleading assumption and ambiguities in modeling languages like UML [70].

An ontology can also be understood as a vocabulary that express a community’s consensus knowledge about a given domain [2, 55]. This representation of knowledge has to be formal and unambiguous to give a solid foundation for the development of semantic Web-based systems.

One of the main reasons why web services composition cannot be easily automated is the lack of formal semantics that give the meaning to attributes of web services. Because the widely accepted standards of web services only represent syntactical descriptions, the service designer needs to interpret user’s requirements (what users want) and service descriptions (what the service is offering). Therefore, within the field of automated service composition, ontologies are used to match the concepts of goals (from the users) to functionality offered by services.

The semantic description of the web services is achieved by means of ontology languages. Examples of semantic descriptions languages are OWL (Web Ontology Language) and Web Service Modeling Language (WSML). The DynamiCoS framework uses the OWL language to

define the domain ontology. Chapter 5 discusses OWL-based ontologies we use in DynamiCoS during the prototype experimentation.

SPATEL language

To allow DynamiCoS to automatically relate services to goals, it needs semantic support from ontologies to describe concepts of the application domains, and a semantic description language to describe the services of this domain.

DynamiCoS can import different types of semantic description languages, as long as an interpreter for the language has been implemented. Currently, DynamiCoS supports SPATEL [45] to semantically describe the services in a domain, and OWL to specify the Domain ontologies.

SPATEL is an XML-based language designed to semantically describe domain services. This capability is essential to allow the framework to carry out the automatic reasoning. Figure 26 presents the syntactical structure of a SPATEL file that describes a service ('addressValidation').

```

<?xml version="1.0" encoding="UTF-8"?>
<spatel:ServiceLibrary
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:spatel="http://istspice.org/spatel/1.0.0/Spatel"
  xmi:id="f+GPY8mGfUir09WcIsAkcgAA" name="Service Design">

  <nestedPackage xsi:type="spatel:ServicePackage" xmi:id="b5zEVcrIxxk6ZC07Yck0J5gAA"
    name="addressValidation">
    <service xmi:id="EYvIUr9HMUmJCrLAhddQaAAA" name="addressValidation" semPattern="GQIO">

      <!-- Definition of the ontologies -->
      <ontology xmi:id="id" name="e-gov.owl" uri="http://82.74.122.50/e-gov.owl"/>
      <ontology xmi:id="id" name="IOTypes.owl" uri="http://82.74.122.50/IOTypes.owl"/>

      <!-- Definition of the operation -->
      <ownedOperation xsi:type="spatel:ServiceOperation" xmi:id="vCl7yN6sgEinRpMBWJ7t+wAA"
        name="addressValidation">

        <!-- Attributes for the operation -->
        <ownedParameter xsi:type="spatel:ServiceParameter" xmi:id="Cf+s950fIEyLb9cX+x2rbQAA"
          name="address" semType="IOTypes.owl:Address" direction="in" instanceType="String"/>
        <ownedParameter xsi:type="spatel:ServiceParameter" xmi:id="jmF88w00AEC6phtzFpdJWAAA"
          name="citizenId" semType="IOTypes.owl:citizenId" direction="in" instanceType="Integer"/>
        <ownedParameter xsi:type="spatel:ServiceParameter" xmi:id="gEXysJA31Ui8XMKQWQoiPAAA"
          name="confirmation" semType="IOTypes.owl:addressValidationStatus" direction="return"
          instanceType="Boolean"/>
        <semTag xmi:id="id" name="OperationGoal" semType="e-gov.owl:ValidateAddress" kind="goal"/>
      </ownedOperation>

      <semTag xmi:id="id" name="ServiceGoal" semType="e-gov.owl:ValidateAddress" kind="goal"/>

    </service>
  </nestedPackage>
</spatel:ServiceLibrary>
    
```

Figure 26: SPATEL description of the address validation web service.

In this description all the inputs and outputs of the service are represented by inferring the domain ontologies. The tag `<ownedParameter>` refers to all the variables that are specified in the WSDL file of the 'addressValidation' service. The mapping defined in the attribute 'semType' relates variables to the concepts defined in the IO-Types ontology. The tag `<semTag>` with attribute "service goal" links the goal of the service to the concept 'validateAddress' in the 'e-gov' ontology.

The domain ontologies and SPATEL files need to be uploaded in the repository to allow DynamiCoS to operate. One SPATEL file is necessary for each service. Furthermore, although ontologies are intended to be a stable representation of knowledge, they can still be extended when new concepts of the domain or inputs/outputs ontologies are defined.

Semantic matching

Figure 27 shows the relation between two web services, A and B, where only one output parameter of service A can be used by as input parameter for service B. This link between two services is the minimum requirement to create a composite service. In a traditional composition scenario, a service developer needs to grasp all the inputs and outputs of each service to combine them. In dynamic service composition with the DynamiCoS framework, the composition algorithm supports users/developers in this task.

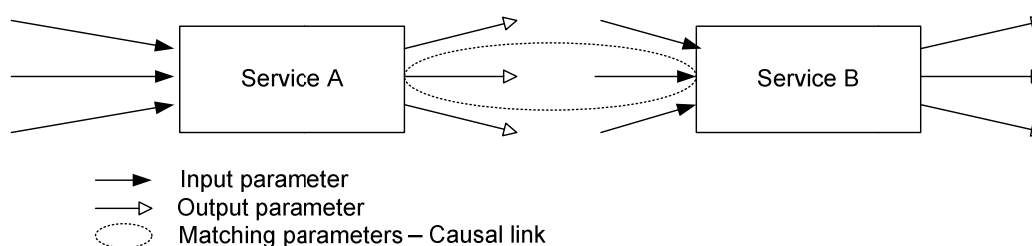
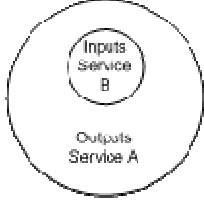
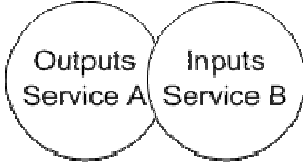


Figure 27: Matching parameters between services adapted from [17].

Table 3 presents the semantic matching patterns that are currently supported by the DynamiCoS.

Table 3: Semantic web service matching patterns

<p>No matching: This happens when none of the properties of the service request or service outputs are similar to the prerequisites of service B.</p>	
<p>Exact: The exact match is the perfect match between a service request (service output) and inputs of the service B. This is the best type of service matching.</p>	
<p>PlugIn: In this case, the inputs of service B are not entirely fulfilled by the outputs of Service A, in this case other PlugIn services would be necessary in the composition to perform service B.</p>	

<p>Subsume: This is the opposite situation of PlugIn matching, where the inputs of service B are entirely fulfilled by the outputs of service A.</p>	
<p>Intersection: In the intersection case, only some of the outputs of service A can be used for service B. Hence, service B needs other services that fulfill the remaining required inputs, just as it happens in the PlugIn matching case.</p>	

4.4 Characterization of users

In order to support users properly, we need to characterize them according to their background knowledge. Afterwards we it is possible to design an adequate *interaction pattern*. To give a more concrete example, consider an individual who goes to a car dealer to buy a car. In the car shop, the salesman asks what type of vehicle the person wishes, what would be its features such as: brand, colour, load capacity, year of manufacture, cost, speed and so on. In this case, the task of buying a car consists of a series of interactions to match user's requirements to the product that ultimately satisfy both parties.

In this example, the negotiation process would be significantly different depending on the type of background knowledge that both individuals possess. For instance, if the client is an engineer, he/she would focus on vehicle's functionality, whereas a car enthusiastic would rather pay more attention to its design or colour. Yet, someone else would be interested on the internal space and safety for the family. Thus, a knowledgeable and high skilled salesman would be able to conduct a personalized interaction from the beginning to the end of the process.

Similarly, in the e-government context we consider that a citizen accesses the government website to execute some electronic public services. The interaction between the user and system would be different according to the knowledge of the citizen about the service, and the service complexity. Therefore, a tailored approach to a certain target groups of the population based on assumption on their knowledge can potentially increase user's satisfaction of public services.

In [21], the authors characterize four basic types of users that interact with a service composition tool, namely: *layman*, *domain expert*, *technical expert*, and *advanced*. The *layman* does not have an in-depth knowledge about the technology itself, neither the service that he/she is willing to buy, whereas an *expert*, on the other hand, knows partially the context (the technology or the domain of the service). To finalize, an *advanced* user is fully aware of the

possibilities and constraints of the service. This type of user is more likely to derive a greater value from the system by creating more refined compositions.

For the e-government context, we define two categories of users to receive a specific type of support:

1. Ordinary citizens (*layman*), who wish to exercise their citizenship through e-government facilities, but do not possess in-depth knowledge about government public services and their underlying technologies. Therefore, elaborating a specific approach to attend this type of user will possibly increase their accessibility to e-government systems and foster the democratization of public services.
2. Civil servants, who are experienced with certain domains of public services (*domain experts*). The role of this user is to assist citizens at the counter and act on their behalf to help them achieve their goals. It is possible that the interaction strategy applied for ordinary citizens can also serve a civil servant, however, a tailor-made solution may increase satisfaction and productivity.

Table 4: End-users' classification for e-government.

End-user type	Domain knowledge	Technical knowledge
Citizen	No	No
Civil servant	Yes	No/Some

To allow the e-government system to determine which type of support should be selected to a given user, we suggest the utilization of a profile-ontology-based approach that defines the characteristic of the each user and allow for machine reasoning [71-72]. However, this lies beyond the scope of this project. We assume that the users are already logged in and have already been classified according to their knowledge or role.

We give special attention to the citizen end-user type because it requires a more general approach that can be extended to other end-user types if necessary. This method fulfils the e-government trend to provide services with minimum human intervention.

4.4.1 Citizen interaction pattern

In this interaction pattern we assume that citizens are already authenticated in the system, but are not fully aware about the characteristics of public services. For example, they neither know the official nomenclature of e-government services nor the procedures to access them. Therefore, to support this type of users we propose an *interaction pattern* that is focused on identification of suitable e-government services by means of a natural language search mechanism. After indicating his or her goals in natural language, the front-end component should guide the citizen in the process of backward composition until the initial goal service preconditions are all met.

The backward composition strategy is used to assure that the citizen fulfills all the prerequisites of his/her service. If the citizen has several goals, the process must be repeated for each goal. To perform this type of composition the engine, can make use of data from the citizen's profile. Afterwards the engine could act in two different ways:

1. Automatically retrieving the information and reducing the number of interactions with the user;
2. Suggesting values in the fields where the citizen verifies which information is being sent before proceeding with the composition.

In case the information is not available, the engine should ask the citizen to enter additional information. If the citizen cannot enter the missing information, the composition process is stopped, showing the contact information of the department where this missing information can possibly be found. If the service requires a personal meeting with the civil servant, then a service should be invoked to allow the citizen to set up an appointment with a civil servant.

The UML activity diagram in Figure 28 presents the *interaction pattern* that guides the citizen while composing a service. The *interaction pattern* corresponds to the coordination mechanism between the front and back-end execution context. Furthermore, the sequence is broken down in two distinctive phases:

1. **Composition:** During the composition phase, DynamiCoS plays the role of a composer assistant by receiving information and interacting with the citizen in order to locate the services that fulfil the goal of the citizen.
2. **Execution:** At this stage, the engine makes use of information gathered in the composition phase and executes the services. It is important to not that the actual execution of the services happen on the front-end, DynamiCoS only has all the necessary information to find and call the services in the correct composition order with the information that was provided by the user. At the end the results of the execution should be displayed to the user.

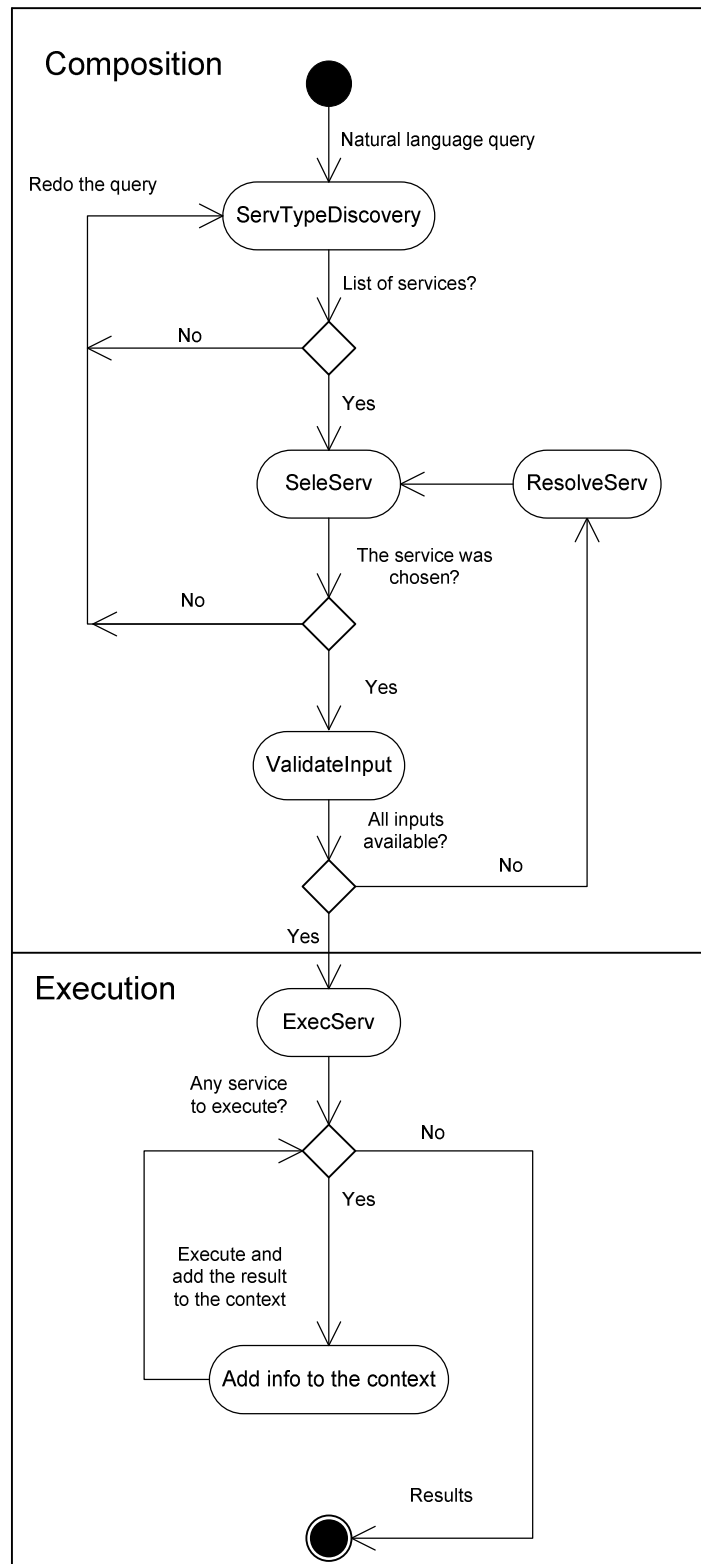


Figure 28: Citizen Interaction pattern – Activity diagram [73].

Below we present the four phases that the citizen has to pass through during the service composition (User interface), which executes one of more *interaction types* in the background:

1. **Your question:** The citizen enters his/her requirements in natural language and receives a list of electronic public services that correspond to his/her goal. Because the citizen might not precisely know which service corresponds to his/her goals before every interaction there is a possibility of repeating this process. This phase executes the Service type/info discovery.
2. **Related services:** From the list of services the citizen selects a service that best fits his/her life event (goal) based on the natural language descriptions. In the background a selection code is sent forward to the next phase when the service needs to have its inputs resolved.
3. **Your situation:** The system starts a personalized composition. Because citizens have different profiles and goals, the interaction will vary from citizen to citizen. The number of interactions and the success of the operation depend on factors such as:
 - The characteristic of the goal;
 - The amount of citizen's profile information that is held by the government;
 - The availability of e-government web services.

The more information gathered the less interaction steps should be necessary. The composition finishes when all the prerequisites are fulfilled. However, if the prerequisites neither be retrieved from an internal e-government system nor by the user, the operation is terminated indicating the contact information of the service that is missing.

The *interaction types* executed are: *Validate Inputs*, *Resolve service* and *Add info to the context*.

4. **Results:** In the last phase, if the all prerequisites are fulfilled, the service composition engine generates the final orchestration specification, which is then used to execute all the composition services. Once the services are executed one or more results are presented and added to the composition context.

The UML activity sequence diagram in Figure 29 shows the interaction of the citizens between and the user-interface, and the role of the other components of our solution framework for this *interaction pattern*.

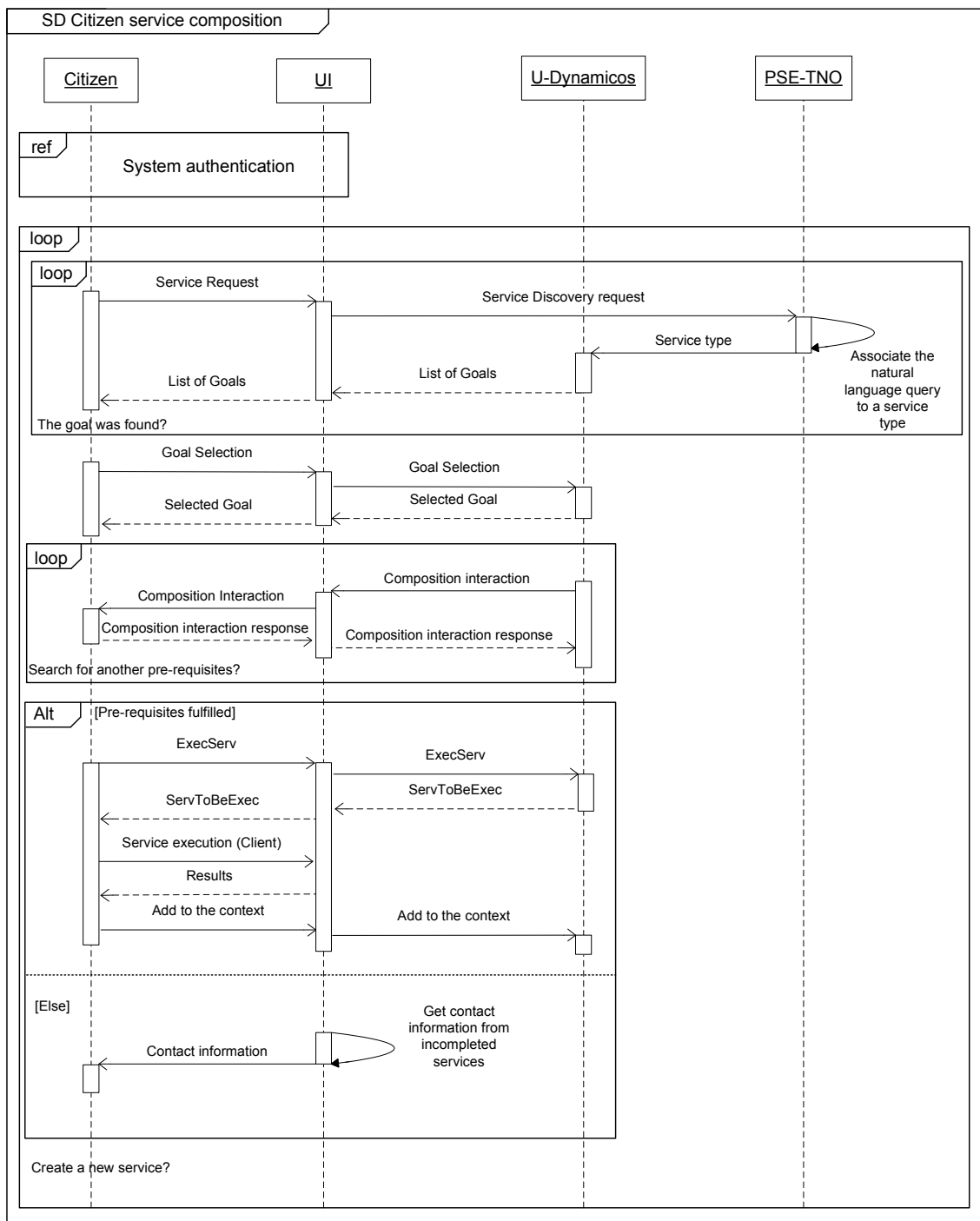


Figure 29: Citizen *Interaction pattern* – Sequence diagram [73].

4.4.2 Civil servant interaction pattern

A civil servant helps citizens with their requests at the counter. His/her role is to act on behalf of the citizen to compose a service using his/her knowledge about nomenclature and processes of public services. The interaction patten is depicted on Figure 30.

Below we discuss the main steps of the civil servant *interaction pattern* for attending a citizen:

1 – **Citizen's goals:** After the civil servant identifies the citizen in the system, he/she is able to enter the request of the citizen using his/her knowledge of public service to query the system for services that support the goals of the citizen.

2 - **Related services:** From the list of services the citizen has to select a service that best fit to the citizen's life event.

3 – **Citizen's situation:** The system starts the composition interaction with the civil servant according to the prerequisites of the goal. The number of interactions and the successfulness of the operation will depend on factors such as:

- The characteristic of the goal and its context.
- Knowledge of the civil servant in terms of government processes.
- The amount of citizen's profile information that is held by the government.
- The availability of e-government web services.

The more information gathered by the system the less interaction steps are necessary. The composition finishes when all the prerequisites are fulfilled. However, if the prerequisites cannot be retrieved from the e-government system, the operation is terminated indicating the contact information of the service that is missing. This information should be interpreted by civil servant before forwarding to the citizen.

Furthermore, to improve the effectiveness of the civil servant while attending citizens at the counter, in this stage he/she is able to redo the composition to add more services which, he/she considers being relevant to citizen based on experience. In this case the additional services are considered in a forward chain of services.

4 – **Results:** In the last phase, if the all prerequisites of all goals are fulfilled the service composition engine generates the final orchestration of services and presents the results. Otherwise, the system returns the contact information for the missing prerequisites, which need to be interpreted by the civil servant before forwarding the information to the citizen.

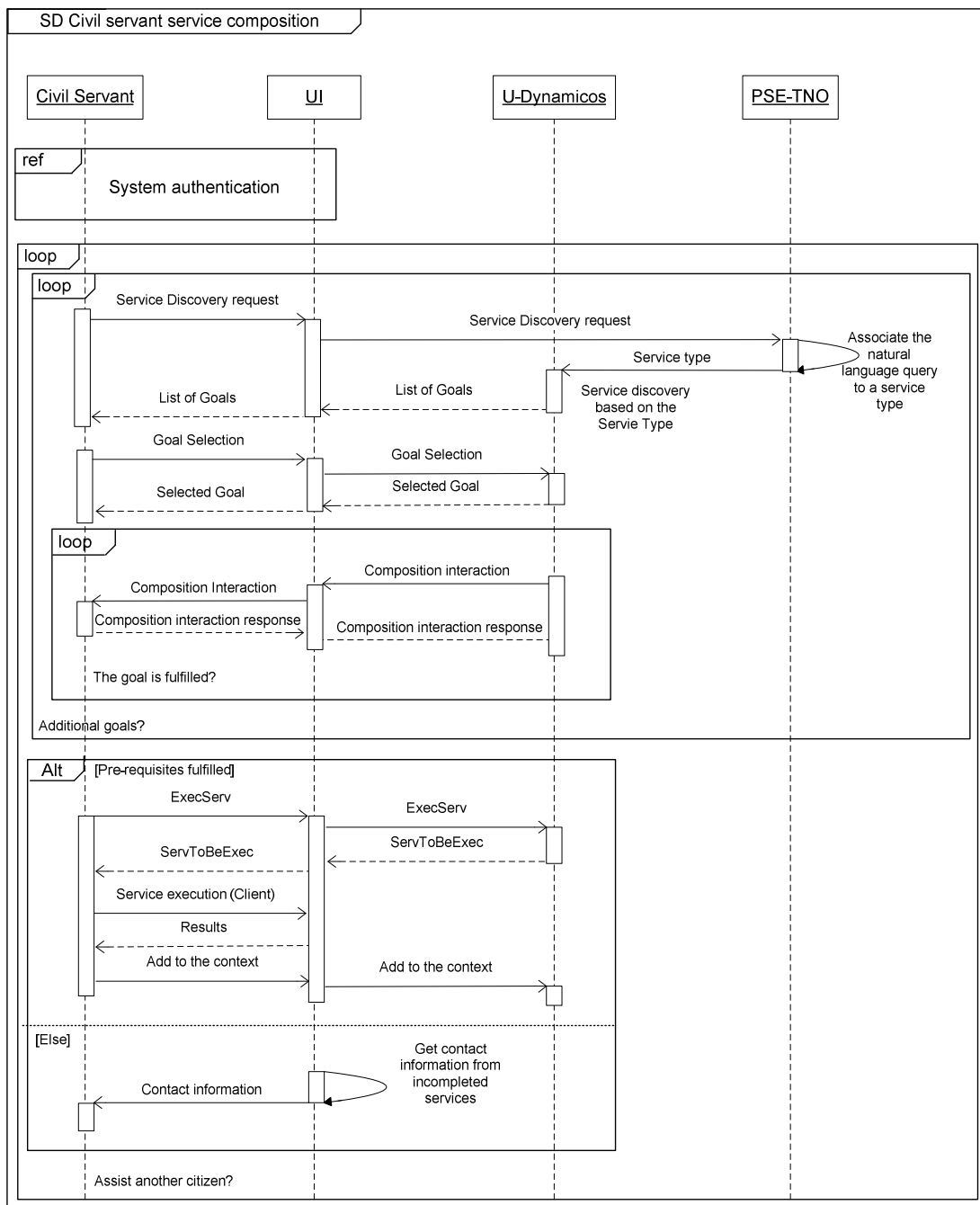


Figure 30: Civil servant interaction pattern.

5 Handicap Parking permit – Prototype

The case study that has been used as a proof of concept for this work is derived from the TNO report “Public Service Experience: Concepten en Voorbeelden” [9]. This chapter presents the “handicapped parking permit request” case in which we have applied our solution framework. Although this case has not been checked by any governmental organization, it already gives some indication of the suitability of our automated service composition.

5.1 Case context

Mr. De Vries has trouble walking and throughout most of his life he commuted by means of public transportation such as bus, train and metro. In recent months, he has purchased a car to improve his daily routine by reducing the distance which he has to walk. However, because the neighborhood is always crowded, most of the parking spaces situated in front of his door are usually occupied by other vehicles. This obliges him to park his car further away from his home. Furthermore, the cars in the nearest parking place belong to people that apparently are not handicapped.

To overcome this problem, Mr. De Vries logs into the Dutch government web site to request a parking permit for handicapped citizens. However, to request the public service he desires, several government agencies like the municipality (‘GBA’: Municipal Population Register and availability of parking lot), the ‘RDW’ (license plate), the ‘CBR’ (valid driver’s license) need to be consulted. As a result, Mr. De Vries spent a great deal of time visiting several websites to get data and in some cases he had to physically visit some government departments to validate documents, which costed him a great deal of time.

5.2 Service definition

In this section we describe the functionality of each web service used that compose the proof of concept scenario. We broke down the services in two categories:

1. Services that carry out validation checks with true or false outputs.
2. Services that perform orders of citizens, requiring several inputs and validations prior to their execution.

In addition, the functionality of the services is represented in terms of inputs/outputs as well as in terms of implementation. A real case scenario would certainly demand complex algorithms, in-depth knowledge about laws, regulations, and so forth. However, we believe the services that have been created sufficiently resemble a public service process and are suitable to test our interaction strategy.

Figure 31 shows the web services that have been developed for our proof of concept, depicting not only the inputs/outputs of each web service, but also how they are connected to support the service number 7 – Order Handicap Parking place.

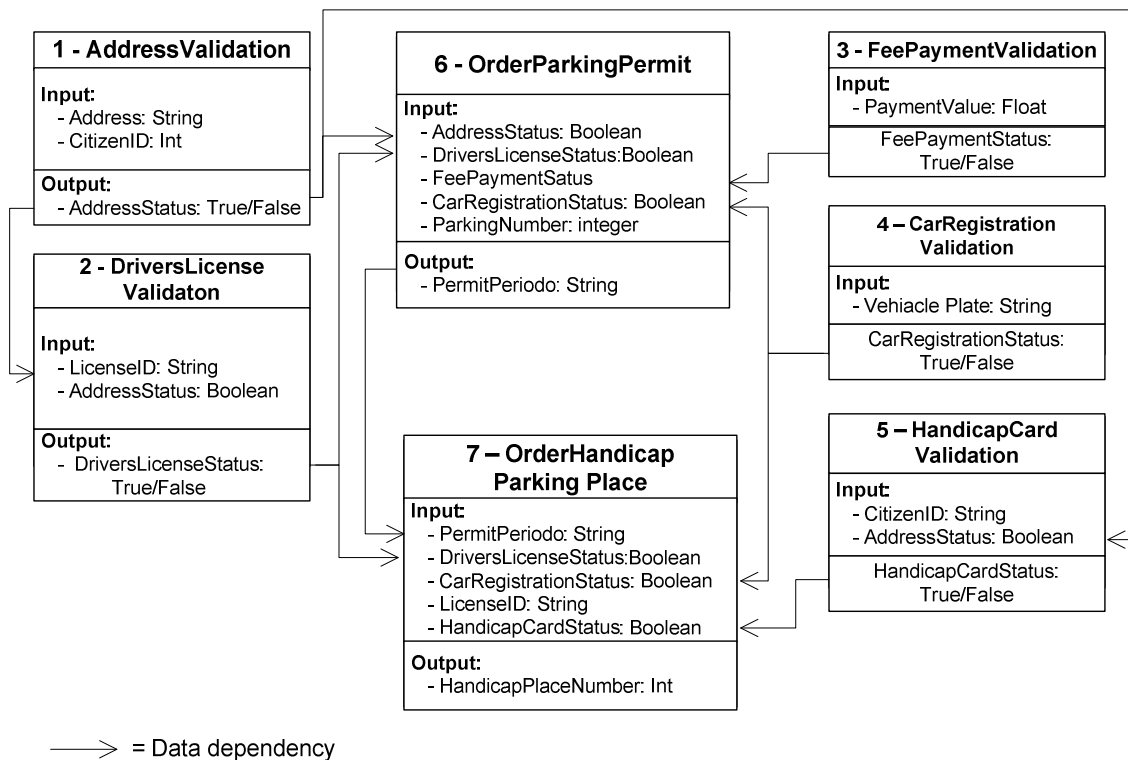


Figure 31: Web services for the e-government scenario.

Each web service is briefly described below:

1. **Address Validation:** Validating a home address enables the municipality to ensure that the records in their database are complete. This allows citizens to receive important licensing and documents in a timely and cost-effective manner. The address validation mechanisms work in several forms depending on the characteristics of municipality management, laws and regulations. For the Dutch case, every inhabitant of the city is obliged to have their citizen identity (Id) and living address registered at the municipality. Therefore, the address validation process checks if the citizen identification matches the informed address. As a result, this web service returns true or false.
2. **Drivers License Validation:** The Driver License validation web service intends to help third parties check if drivers meet the license requirements, thus providing a general means to improve good driving, road safety and identity security. In our case, we assume that the Dutch driving testing organization has a centralized database to control the registrations, and returns true or false values depending on the status of the drivers' licence ID. Furthermore, for this service the address of the driver has to be validated beforehand.
3. **Fee Payment Validation:** A fee is the price that one pays as remuneration for services. We assume that ordering a parking permit requires the payment of a fee. This web service returns a true or false value according to the status of the fee payment.

4. **Car Registration Validation:** The government needs to keep track of each car that is circulating in the country. This information is useful to define public policies regarding security, car traffic, tax payments and so forth. This web service checks the status of each vehicle, returning a true or false value for each case.
5. **Handicap Card Validation:** Citizens who wish to use handicap public facilities need to apply for a handicap card at the social security authority, which evaluates medical status and relevant documents to grant the rights to citizens in individual basis. With the number of the card and address validation, this web service returns true or false statements.
6. **Order Parking Permit:** Citizens who live in an area of the city that has restricted parking space may be eligible to order a resident's parking permit. In order to do so, the citizen has to indicate which parking space he/she wants, validate his/her address, driver's license and car registration. Because this service does not apply to all citizens, fee payment is also required. The result of this service is an allowance period for parking the car in that specific place.
7. **Order Handicap Parking Place:** This is the ultimate goal of the citizen in our testing scenario. At this point the citizen should have executed all the previous web services. The main out come of this service is an order number that will be added to the agenda of the municipality workers that will eventually go to the parking place and paint the handicap signals.

5.3 Public service ontologies

Currently, governments are moving from non-electronic to digital network-based services. Nonetheless, the digital form of government is still fragmented and not fully interoperable. To tackle this issue an ontological approach has been suggested in [74-77], which consists of developing and applying ontologies to public services. These ontologies should be capable of distinguishing in an unambiguous way the key concepts of the governmental structure, processes and their inter-relationships.

5.3.1 E-gov ontology

For our e-government solution, we adopted and adapted the e-government ontology suggested by [75], which is generic enough to cover different aspects of the public administration (PA). This ontology is presented by the white classes in Figure 32. For our study, the central relationships are "Societal entity – has – need" and "need – relatesTo – Goal" because the citizen is aware of his/her needs and not of the available services, whereas the opposite holds for the government [75]. Figure 32 also presents the two extensions we suggested for our scenario.

1. The classes in yellow represent the concepts of the Life events perspective derived from [76].
2. The classes in green represent our definition of Goals, which can be sub-divided in two types: Orders and Validations.

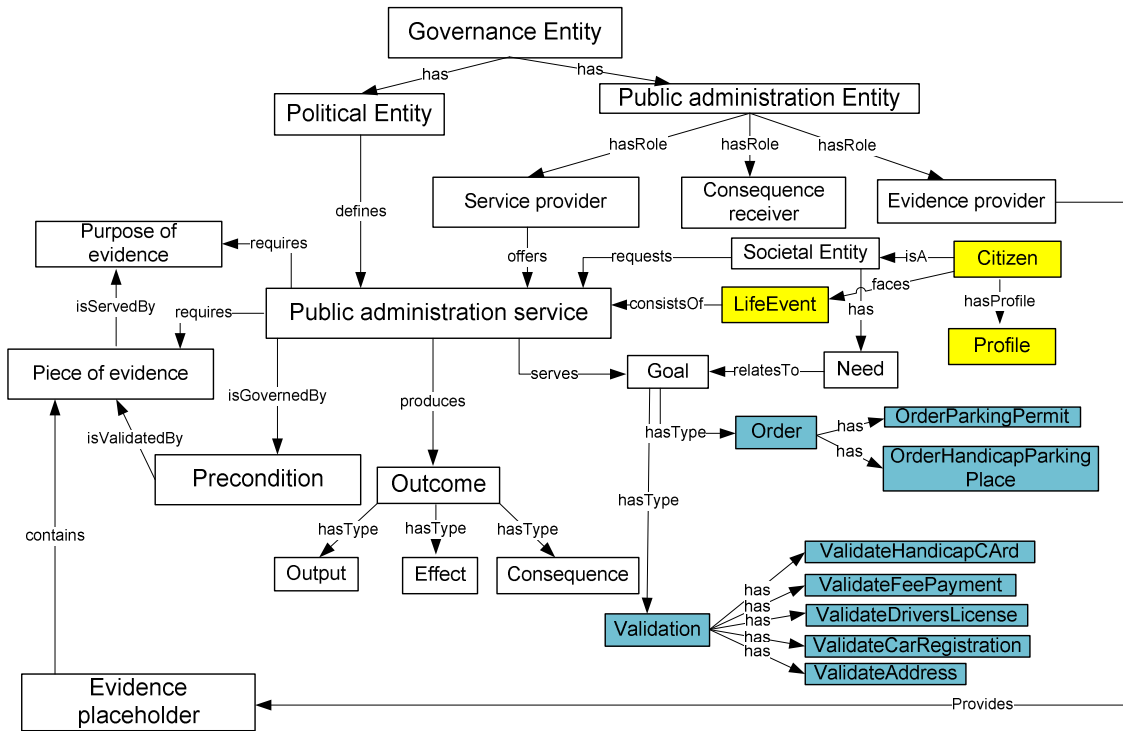


Figure 32: Extended ontology of public services (e-gov ontology).

Figure 33 shows the taxonomy of the e-gov ontology including the proposed extensions.



Figure 33: E-government ontology taxonomy.

5.3.2 IO ontology

The IO ontology refers to the concepts of the *input* and *output (I/O)* parameters of web services. In our case, we based the taxonomy on the I/O of the services that are used in the prototype experimentation phase. Furthermore, whenever a service introduces a new input or output concept, the taxonomy should be extended. Once the IO ontology becomes stable, the I/O of new services should be mapped onto this ontology without any changes in the ontology.

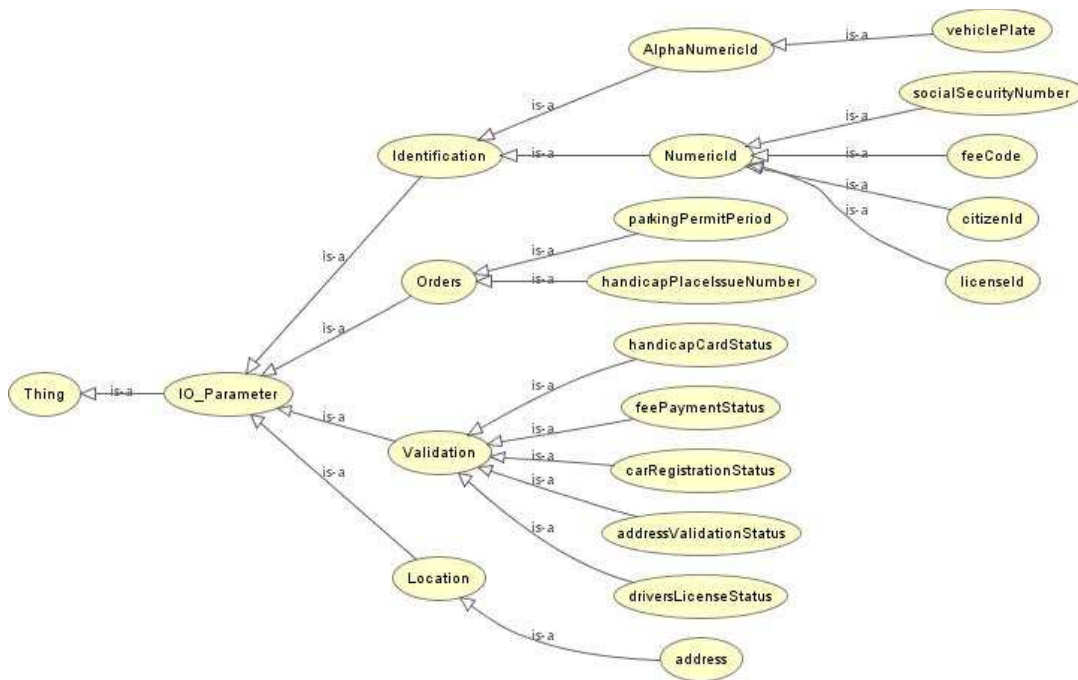


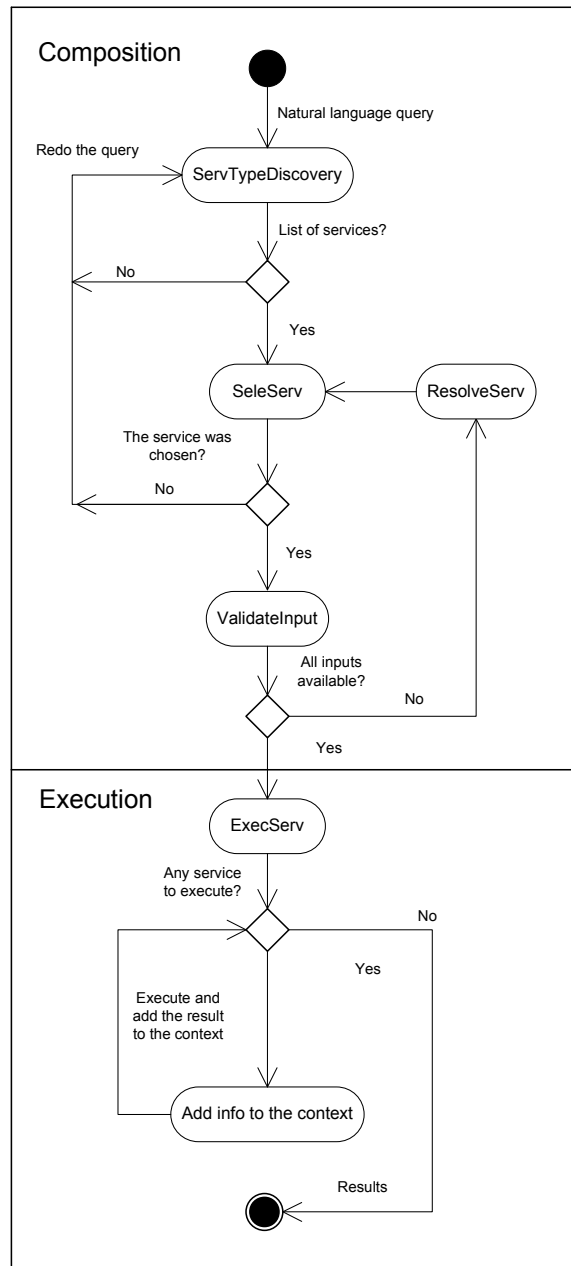
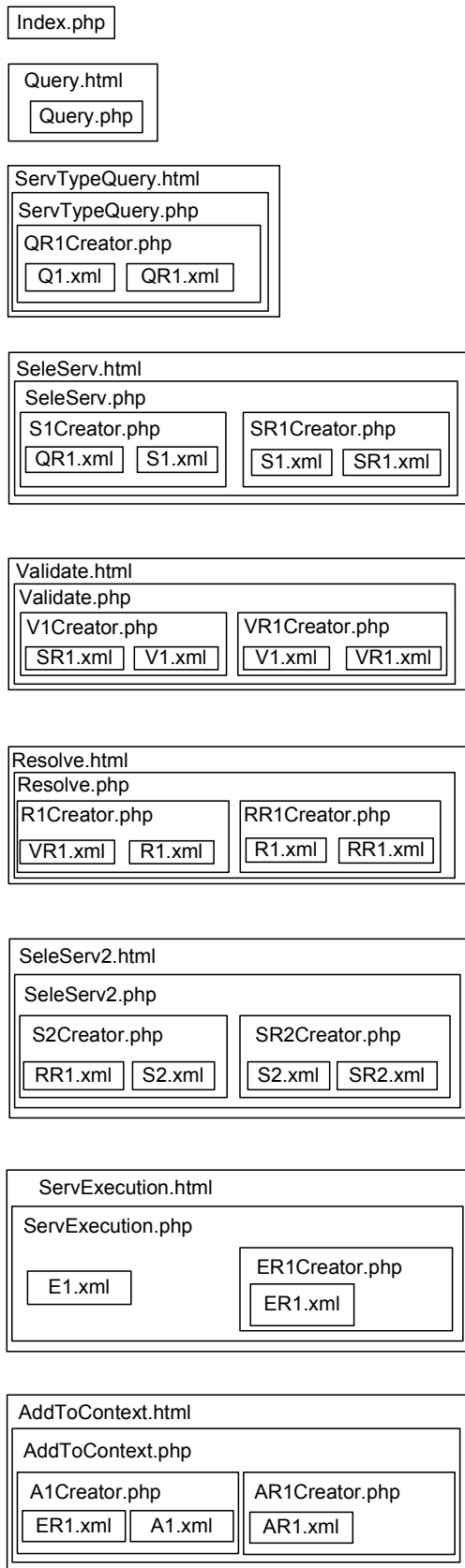
Figure 34: IO ontology

5.4 Implementation

The actual implementation of the prototype was carried out using user interface TNO PSE [8]. The *interaction pattern* has been implemented in between the user interface and DynamiCoS. Because this interaction has to be essentially dynamic we have chose to stick to the same set of technologies that were used by PSE-TNO, which is PHP based.

Figure 35 presents an overview of the prototype structure, the rectangles on the left represent the web pages, encompassing the scripts that manipulate data. On the right side, the activity diagram shows the sequence in which the pages are called (interaction pattern). We decided to indicate the HTML files incorporating the PHP ones, because it nicely depicts the current structure of the user-interface.

Because most of the interactions are dynamically generated, the folder XMLFactory contains the scripts that read and generate the XML files, which in turn are stored in the XMLOperations folder. At the bottom right of the figure, we present the list of acronyms of each XML file that is sent or received from DynamiCoS. The classes are stored in the directory "Classes" which also contains the XML schemas (XSDs).



- Q1.xml = Query 1
- QR1.xml = Query Result 1
- S1.xml = SeleServ 1
- SR1.xml = SeleServ Result 1
- V1.xml = Validation 1
- VR1.xml = Validation Result 1
- V2.xml = Validation 2
- VR2.xml = Validation Result 2
- R1.xml = Resolve 1
- RR1.xml = Resolve Result 1
- S2.xml = Selection 2
- SR2.xml = Selection Result 2
- E1.xml = Execution 1
- ER1.xml = Execution Result 1
- A1.xml = AddToContext 1
- AR1.xml = AddToContext Result 1

Figure 35: Prototype structure

Figure 36 shows the PHP code responsible to generate the S1.XML file (Figure 37). This file is later sent to DynamiCoS by the SR1Creator.php (Figure 38). This procedure of data manipulation is similar to the rest of the XML creators.

```

<?php
//Create the query S1 - SeleServResquestType
$dom = new DOMDocument ();
$dom->formatOutput = true;

//Create the root element does not accept name with spaces
$root = $dom->createElement( "tns:SeleServRequest" );
$dom->appendChild( $root );

//Create the Attribute NS according to the xml received by dynamicos
$attr_ns = $dom->createAttributeNS( 'http://dynamics/interactions/SeleServ', 'tns:attr' );
$attr_ns = $dom->createAttributeNS( 'http://dynamics/interactions/BasicTypes', 'tns1:attr' );
$attr_ns = $dom->createAttributeNS( 'http://www.w3.org/2001/XMLSchema-instance', 'xsi:attr' );
//Create the xsi:schemaLocation element and append it to the root element - as attribute
$root_attr1 = $dom->createAttribute('xsi:schemaLocation');
$root->appendChild($root_attr1);
$root_text = $dom->createTextNode('http://dynamics/interactions/SeleServ SeleServ.xsd');
$root_attr1->appendChild($root_text);

//Creating the element and add it as nested in the root element
$$ServiceSelectionInfo = $dom->createElement( "tns:ServiceSelectionInfo" );
$root->appendChild ( $$ServiceSelectionInfo );

//Creating the element and adding it as nested in the ServiceSelectionInfo element
$$SelectedServiceID = $dom->createElement( "tns1:SelectedServiceID" );
$$ServiceSelectionInfo->appendChild ( $$SelectedServiceID );

//Add the content to the element
$$SelectedServiceID->appendChild($dom->createTextNode($_SESSION['ChosenService']->id ));

//Creating the element and adding it as nested in the ServiceSelectionInfo element
$$ServiceType = $dom->createElement( "tns1:ServiceType" );
$$ServiceSelectionInfo->appendChild ( $$ServiceType );

////Add the content to the element
$$ServiceType->appendChild($dom->createTextNode($_SESSION['ChosenService']->type ));

//print $dom->saveXML() . "\n"; //print the created xml
$dom->save("./XMLOperations/S1.xml");

?>
    
```

Figure 36: S1Creator.php

```

<?xml version="1.0"?>
<tns:SeleServRequest xmlns:tns="http://dynamics/interactions/SeleServ"
    xmlns:tns1="http://dynamics/interactions/BasicTypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dynamics/interactions/SeleServ SeleServ.xsd">
  <tns:ServiceSelectionInfo>
    <tns1:SelectedServiceID>23156730-AA13-11DF-A730-DBD5C848C280-3195</tns1:SelectedServiceID>
    <tns1:ServiceType>http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#OrderHandicapParkingPlace</tns1:ServiceType>
  </tns:ServiceSelectionInfo>
</tns:SeleServRequest>
    
```

Figure 37: S1.xml

```

<?php
//This file connects to the webservice to create the SR1File
//Loads the XML S1 file
    $dom->load("./XMLOperations/S1.xml");

//Transfers the query to a string variable
    $str = $dom->saveXML();

//The connection is based on the WSDL file is defined in "script/vars.php"
//Send 3 arguments to the webservices "coordinator"
// 1 - InteractionType = "SELE_SERV";
// 2 - InteractionArgs = xml as a string;
// 3 - Session ID control.
    $resultServSelect = $_SESSION['soapConnection']->Coordinate(array
        ("InteractionType"=>"SELE_SERV",
         "InteractionArgs"=>$str,
         "SessionID"=>"joni"));

//The XML result is returned
    $str = $resultServSelect->return;

// Loads the XML that contains the result
    $dom->loadXML($str);

// The string xml is formatted into a tree structure
    $dom->formatOutput = true;

// Xml webservice response is stored at XMLOperations/QR1.xml
    $dom->save("XMLOperations/SR1.xml");

?>

```

Figure 38: SR1Creator.php

5.5 Testing scenario

In this section we present the test results of the prototype in which Mr. De Vries accesses the e-government portal to request a handicap parking place. The details of the services, which are not user-friendly, are presented to enable the reader to keep track of the interactions with DynamiCoS. For instance, the ID of services and inputs are generated randomly for each execution. We break down the scenario in two different situations. First the citizen has all the necessary information for ordering a handicap parking place. Second the citizen does not know the inputs for the service, therefore the interactions enable the citizen to access services that obtain all the missing inputs. The service dependency relations amongst the web services are depicted on Figure 31.

5.5.1 Situation 1

Figure 39 shows the first interaction of Mr. De Vries with the e-government web portal. According to the "life events" strategy, instead of showing a tree structure of the e-government services, the web portal has to enable Mr. De Vries to express his life goals that need governmental support. In this case, the question he performs is exact term that is found of the IO ontology (service type). The natural language mechanism, which would enable a full piece of text to be associated to a service type, has not been integrated to the prototype yet.



Figure 39: First citizen interaction.

Figure 40 shows the result of the query in DynamiCoS in which only one service was found for that life event. For Mr. De Vries, the only important information of this screen is the service description and the list of prerequisites. At this point, Mr. De Vries selects the service. In case this is not the service he is looking for he has the possibility of restating the query.

Service Selection

▶ **Name service 1 = orderHandicapPlace**
Service ID = 23156730-AA13-11DF-A730-DBD5C848C280-410
Service type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#OrderHandicapParkingPlace
Service Description = The order handicap parking place service gives the order to the municipality workers to paint and put the handicap signs the parking place defined the parking permit. The result of this service is a order number.
Service Inputs1:
 The name of the input = licenseld
 The ID of the input = 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-1
Service Inputs2:
 The name of the input = handicapCardValidation
 The ID of the input = 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-2
Service Inputs3:
 The name of the input = driversLicenseStatus
 The ID of the input = 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-3
Service Inputs4:
 The name of the input = carRegistration
 The ID of the input = 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-4
Service Inputs5:
 The name of the input = parkingPermitPeriod
 The ID of the input = 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-5

Your question: (Retrieve services according to their semantical type)

Figure 40: Service discovery result.

Figure 41 shows the selection of the service and a form is generated according to the inputs for that service. The information that has to be filled in the form can be entered by the end-user, taken from his profile or from the output of the mother subservice. In this case, we assume that Mr. De Vries has all the necessary information for ordering a handicap parking place. After, the system verifies the validity of the all inputs of the service, it enables Mr. De Vries to execute the service and get his permit (see Figure 42).

Inputs to validate

Selected service Name = orderHandicapPlace
Service ID = 23156730-AA13-11DF-A730-DBD5C848C280-410
Service Description = The order handicap parking place service gives the order to the municipality workers to paint and put the handicap signs the parking place defined the parking permit. The result of this service is a order number.

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#licenseId
SyntacticalType Integer

Insert the licenseId
Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#handicapCardStatus
SyntacticalType Boolean

Insert the handicapCardValidation
Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#driversLicenseStatus
SyntacticalType Boolean

Insert the driversLicenseStatus
Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#carRegistrationStatus
SyntacticalType Boolean

Insert the carRegistration
Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#parkingPermitPeriod
SyntacticalType Boolean

Insert the parkingPermitPeriod
Suggested values :

Figure 41: Validation of the inputs

Validation Results

Inputs were validated

Figure 42: Service validation results

During the execution phase, the system sends a request to DynamiCoS asking what is the next service that needs to be executed and what were the input values that were entered in the composition phase. DynamiCoS returns these information plus the information for invoking the service. The information of Figure 43 is not intended to be displayed to Mr. De Vries, but it is used by the front-end to perform the actual service execution, sending the results of the execution back to DynamiCoS that adds the return values to the user’s composition context. This information can used to fulfil the inputs of another service in the composition chain.

Execution of the service

Selected service Name = orderHandicapPlace
Service ID = 23156730-AA13-11DF-A730-DBD5C848C280-410

Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#OrderHandicapParkingPlace
Service Description = The order handicap parking place service gives the order to the municipality workers to paint and put the handicap signs the parking place defined the parking permit. The result of this service is a order number.

Execution Value 123456

Input Name licenseld
Input Id 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-1

Execution Value Validated

Input Name handicapCardValidation
Input Id 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-2

Execution Value Valid

Input Name driversLicenseStatus
Input Id 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-3

Execution Value Valid

Input Name carRegistration
Input Id 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-4

Execution Value 12/2010 - 05/2010

Input Name parkingPermitPeriod
Input Id 23156730-AA13-11DF-A730-DBD5C848C280-410-InputID-5

Figure 43: service execution

5.5.2 Situation 2

In this situation we repeat the steps of the previous situation with the difference that now Mr. De Vries does not possess the inputs for the service. Thus the service validation indicates which services are missing, and the composition engine gives suggestions of services that can provide that input. The interaction continues until all the requirements of the service are validated. Figure 44 shows the form that is sent to validation. Figure 45 indicates the result of the input validation.

Inputs to validate

Selected service Name = orderHandicapPlace
Service ID = 23156730-AA13-11DF-A730-DBD5C848C280-4986
Service Description = The order handicap parking place service gives the order to the municipality workers to paint and put the handicap signs the parking place defined the parking permit. The result of this service is a order number.

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#licenseId
SyntacticalType Integer

Insert the licenseId

Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#handicapCardStatus
SyntacticalType Boolean

Insert the handicapCardValidation

Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#driversLicenseStatus
SyntacticalType Boolean

Insert the driversLicenseStatus

Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#carRegistrationStatus
SyntacticalType Boolean

Insert the carRegistration

Suggested values :

SemanticalType http://ewi887.ewi.utwente.nl/ontologies/IOTypes.owl#parkingPermitPeriod
SyntacticalType Boolean

Insert the parkingPermitPeriod

Suggested values :

Figure 44: Empty form to be validated.

Validation Results

The input : 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-1 is not valid
 The input : 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-2 is not valid
 The input : 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-3 is not valid
 The input : 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-4 is not valid
 The input : 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-5 is not valid

Figure 45: Inputs to be resolved

Figure 46 and Figure 47 show the suggestions of services for each input of the initial service. The first unsolved input does not have a web service that supports it, and therefore this information has to be entered by the Mr. De Vries. The second unsolved input can be obtained from the output of the handicapCardValidation service, whereas input number 4 can be obtained from one of the two instances of carRegistrationValidation service.

Suggested Service(s)

<p>Not Solved service 23156730-AA13-11DF-A730-DBD5C848C280-4986</p>
<p><input type="checkbox"/> Not Solved Input = 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-1</p>
<p><input type="checkbox"/> Not Solved Input = 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-2</p>
<p><input type="checkbox"/> Matching Value ID = 01D17910-AA13-11DF-B910-DD31D4EA908B-5873-OutputID-1</p>
<p><input type="radio"/> Service Name = handicapCardValidation Service ID = 01D17910-AA13-11DF-B910-DD31D4EA908B-5873 Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateHandicapCard Service Description = The handicap Card Validation service checks if the citizen is recognized by the government as a handicap person to be able to apply for benefits. The result of is a true or false response. Input Name = socialSecurityNumber Input ID = 01D17910-AA13-11DF-B910-DD31D4EA908B-5873-InputID-1 Input Name = addressStatus Input ID = 01D17910-AA13-11DF-B910-DD31D4EA908B-5873-InputID-2</p>
<p><input type="checkbox"/> Not Solved Input = 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-3</p>
<p><input type="checkbox"/> Matching Value ID = CB1CCDC0-AA12-11DF-8DC0-F9D92FBF2FA5-517-OutputID-1</p>
<p><input type="radio"/> Service Name = driversLicenseValidation Service ID = CB1CCDC0-AA12-11DF-8DC0-F9D92FBF2FA5-517 Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateDriversLicense Service Description = The Driver Licence validation service checks if the drivers meet the license requirements. The result of is a true or false response. Input Name = licenseID Input ID = CB1CCDC0-AA12-11DF-8DC0-F9D92FBF2FA5-517-InputID-1 Input Name = addressStatus Input ID = CB1CCDC0-AA12-11DF-8DC0-F9D92FBF2FA5-517-InputID-2</p>

Figure 46: Automatic suggestion of services to fulfil the missing inputs

<p><input type="checkbox"/> Not Solved Input = 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-4</p>
<p><input type="checkbox"/> Matching Value ID = 91527E00-AA12-11DF-BE00-932E4B25BADB-599-OutputID-1</p>
<p><input type="radio"/> Service Name = carRegistrationValidation Service ID = 91527E00-AA12-11DF-BE00-932E4B25BADB-599 Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateCarRegistration Service Description = The car registration validation service checks if the car is in conformity with the car registration authority. The result of is a true or false response. Input Name = vehiclePlate Input ID = 91527E00-AA12-11DF-BE00-932E4B25BADB-599-InputID-1</p>
<p><input type="checkbox"/> Matching Value ID = 34DAC510-AA12-11DF-8510-D709E9443088-7070-OutputID-2</p>
<p><input type="radio"/> Service Name = carRegistrationValidation Service ID = 34DAC510-AA12-11DF-8510-D709E9443088-7070 Service Type = http://ewi887.ewi.utwente.nl/ontologies/e-gov.owl#ValidateCarRegistration Service Description = The car registration validation service checks if the car is in conformity with the car registration authority. The result of is a true or false response. Input Name = vehiclePlate Input ID = 34DAC510-AA12-11DF-8510-D709E9443088-7070-InputID-2</p>
<p><input type="checkbox"/> Not Solved Input = 23156730-AA13-11DF-A730-DBD5C848C280-4986-InputID-5</p>
<p><input type="checkbox"/> Matching Value ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-OutputID-1</p>
<p><input type="radio"/> Service Name = orderParkingPermit Service ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576 Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#OrderParkingPermit Service Description = The order parking permit service evaluates the current context of the citizen and if all prerequisites are ok it generates to the citizen an allowance period for parking his/her car in a specific place. Input Name = addressValidationStatus Input ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-1 Input Name = driversLicenseStatus Input ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-2 Input Name = feePaymentStatus Input ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-3 Input Name = carRegistrationStatus Input ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-4</p>

Select a service

Figure 47: Automatic suggestion of services to fulfil the missing inputs (Continuation)

At this point Mr. De Vries selects that the “orderParkingPermit” service. Because we want to demonstrate the backward composition, we assume that the inputs for this service were already validated except for the validation of the fee payment, which is supported by another service. Figure 49 indicates the missing input.

Service to Validate

Selected service Name = orderParkingPermit
 Service ID = 41D71740-AA13-11DF-9740-BE9EF99BC219-6576

Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#OrderParkingPermit
 Service Description = The order parking permit service evaluates the current context of the citizen and if all prerequisites are ok it generates to the citizen an allowance period for parking his/her car in a specific place.

Input Name addressValidationStatus
 Input Id 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-1

Input Name driversLicenseStatus
 Input Id 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-2

Input Name feePaymentStatus
 Input Id 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-3

Input Name carRegistrationStatus
 Input Id 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-4

Figure 48: Order parking permit selected and ready to be validated

Validation Results

The input : 41D71740-AA13-11DF-9740-BE9EF99BC219-6576-InputID-3 is not valid

Figure 49: Fee payment missing

Figure 50 shows once again the system giving a suggestion to obtain a missing input, but at this time the input belongs to the second service in the composition chain. After selection, the service “feeValidation” is validated and ready to execute. The execution (see Figure 52) has to be carried out by the front-end and the result of the fee validation has to be added to the execution context, which is responsible for adding this information to the unsolved inputs of the previous service.

In contrast to the composition phase in which the service tree is created, the execution phase executes the services in the opposite sequent until the target service has all the inputs correctly fulfilled.

Unfortunately, although we have developed and deployed all the web services for this scenario, due to time restrictions we did not test the actual execution of the services. Nonetheless, we believe that the current state of the prototype can already give a clear picture of our solution framework.

Matching Value ID = E4D790B0-AA12-11DF-90B0-92E182626779-74-OutputID-1

Service Name = feeValidation
Service ID = E4D790B0-AA12-11DF-90B0-92E182626779-74
Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateFeePayment
Service Description = A fee payment validation service checks if the citizen has paid the fee for the public service that he/she is applying for. The result of is a true or false response.
Input Name = feeCode
Input ID = E4D790B0-AA12-11DF-90B0-92E182626779-74-InputID-1

Figure 50: Fee validation web service

Service to Validate

Selected service Name = feeValidation
Service ID = E4D790B0-AA12-11DF-90B0-92E182626779-74

Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateFeePayment
Service Description = A fee payment validation service checks if the citizen has paid the fee for the public service that he/she is applying for. The result of is a true or false response.

Input Name feeCode
Input Id E4D790B0-AA12-11DF-90B0-92E182626779-74-InputID-1

Figure 51: Validation of the fee validation service.

Execution of the service

Selected service Name = feeValidation
Service ID = E4D790B0-AA12-11DF-90B0-92E182626779-74

Service Type = http://ewi887.ewi.utwente.nl/ontologies/Goals.owl#ValidateFeePayment
Service Description = A fee payment validation service checks if the citizen has paid the fee for the public service that he/she is applying for. The result of is a true or false response.

Execution Value 12345

Input Name feeCode
Input Id E4D790B0-AA12-11DF-90B0-92E182626779-74-InputID-1

Figure 52: Fee validation service execution

6 Conclusions

Our research was based on research questions well formulated in the beginning of the project. In this section we systematically answer these questions.

6.1 Answers to the research questions

“What are public services, e-government, and the current state of affairs of the Dutch e-government?”

Public services are services provided by the government to its citizens, whereas e-government (e-gov) is the utilization of electronic technologies to improve the quality and agility of public services, which will eventually be beneficial to the development of the entire society.

E-government initiatives are also a response from the government to support the demands of citizens, who are more often disappointed than enthusiastic about their governing institutions [33]. These higher expectations from the public can be explained by inevitable comparisons with services provided by the private sector and especially by rapid changes in citizens' values, which became more individualized with the presence of digital technologies [32-34].

In a historical analysis, we have seen that since the second half of the 1990's, governments were challenged to use the Internet as a communication channel with their citizens [32, 36]. As a result, there was a proliferation of governmental web portals that simply emulated, in an electronic way, the structure and logic of regular public services. Hence, the inherent fragmentation of the government was still present with low levels of personalization.

The Netherlands was considered by the United Nations [32] to be one of the leading nations in terms of e-government. However, while the Dutch e-government has achieved good results in integrating its disparate systems, it is still striving to provide highly personalized services to the citizens.

What is service composition and why is it necessary?

In order to understand service composition and how service composition technologies work, we first clarified the core concepts of the Service-Oriented Architecture (SOA). SOA is not a technology, but rather a set of principles that can help organizations integrate disparate systems and provide a more flexible structure that supports business processes.

In technological terms, the key concept of SOA is the atomization of services, which means the externalization of system functionality. This externalization can be achieved by means of web services technologies, which have been standardized to assure higher levels of interoperability.

Organizations willing to benefit from SOA, should carry out service composition. Web services (automated services) bring add-value to the business when they are coherently combined to support business processes that address the end-user's requirements, which in recent years have demanded higher levels of personalization. However, composing and orchestrating a set

of web services is still a rather technical, complex and time-consuming task. For this reason, various researchers and IT practitioners are investigating ways to perform automated service composition.

What are the current technologies to support service composition?

Technologies for service composition need to comply with the current standards for web services, such as WSDL, UDDI and SOAP, which are XML-based. A prominent approach for composing services is WS-BPEL, which is an XML-based language that allows the orchestration of web services to support business processes.

In this research we concluded that there are already several feasible technologies to support service composition using BPEL. However, they only tackle manual compositions at design-time. Although these technologies are capable of reducing the steep learning curve for composing business processes with web services, orchestration is still a complex and time-consuming task, demanding automatic or semi-automatic mechanisms that enable end-users to compose services according to their requirements (runtime).

From the literature, we have found out that although some standards that address interoperability have already been defined, an overall agreement on standards for automated service composition is still an open issue due to fundamental differences in the context and the purpose of each solution.

The service composition frameworks that we have studied (WSMO, METEOR-S and DynamiCoS) bring new perspectives to the business and service composition area alike, but they still need to mature before being applied in a production environment. In addition, from the experiences of TNO and the available documentation of the frameworks, we realized that much has been said about the capabilities of each framework, but the actual implementation of these capabilities is still an ongoing work. The outcome of our research intends to fulfil this gap.

How to use a service composition tool in the e-government context?

The demand for service composition in the e-government context has two main reasons:

1. With the advent of the Internet, individuals are demanding more personalized services.
2. E-government systems need to be as inclusive as possible to attend to the needs of the population. Hence, to have an overall acceptance of the e-government systems, e-government applications need to be integrated and easy to use.

To address these requirements for e-government, we have studied the life events strategy and composed a solution framework with suitable technologies to support it.

The life events strategy suggests that e-government web portals need to assume the perspective of the citizens, which implies that they need to interact with citizens to understand what they need, rather than only publishing information. The aspects that are relevant to the citizens are the events that take place in their lives and require public services. Examples of life events are: birth, school inscription, change of address and marriage. Furthermore, these events should be supported through a single point of access that shields the citizens from the fragmentation and complexity of the governmental structure.

To support the life events strategy, we developed a solution framework that adopts the PSE-TNO solution as the front-end and DynamiCoS framework as the service composition engine in the back-end. Our main contribution consists of an *interaction pattern* between the front and back-end that enable users (citizens) to create personalized services following dynamic interactions. The suggested *interaction pattern* is implemented as an orchestrator of the web service orchestration engine, which defines the workflow of *interaction types* that the user follows while performing a service composition.

As proof of concept, we carried out a prototype experimentation to demonstrate the dynamic service composition performed by a disabled citizen who requires a handicapped parking place for an automobile near his house.

Main question

“How can a runtime customer-driven service composition strategy improve public services?”

A runtime customer-driven service composition strategy can improve public services if it is able to operationalize the main principles of the life event strategy, enabling citizens to dynamically compose services according to their goals through a single point of access. This operationalization is achieved by defining a citizen-oriented interaction pattern that links front-end to the service composition engine. We assume that those principles of the life event strategy can improve the quality of public services.

6.2 Contributions

Our main contribution to the state-of-the-art in the area e-government and service composition has been:

- An environment that allows the proof of the hypothesis that the life event strategy can improve public services.
- Fully dynamic interactions between the citizen and the composition engine that responds according to citizen’s goals and functionality of the service.
- A backward service composition, which is a pivotal requirement to public services.
- The support the main principles of the life events strategy such as: personalization of services, interpretation of citizen’s perspective and centralization of access of public services.

- Two different approaches (*interaction patterns*) that addresses the characteristics of two types of users (citizen and civil servant)

6.3 Limitations

Our results have the following limitations:

- **Natural Language recognition:** We wanted to integrate the natural language mechanism provided by TNO into our prototype, but we did not achieve it due to time constraints. The main issue was the overlapping functionality of the natural language mechanism and the DynamiCoS service/discovery *interaction type*. Although, we were able to extract and adapt the Java code to support English queries using translated (Dutch to English) service descriptions, we had to keep the focus on the execution of the *interaction patterns*. Therefore, the two solutions are not fully integrated as we envisioned.
- **Service Execution:** The citizen *interaction pattern* was divided into the “composition” and “execution” phases. Due to time and technical limitations we could demonstrate only the composition part, where DynamiCoS gathers information from the user until all the inputs of services are fulfilled. Although we implemented all the web services for testing scenario, execution mechanism in the front-end still requires a function that receives the execution date from DynamiCoS and returns the result of the service execution to the DynamiCoS context.
- **Web Service Semantic Description:** DynamiCoS can import different types of semantic description languages, as long as an interpreter for the language has been implemented. For practical reasons, in our project we only used the SPATEL language with manual creation of the descriptions based on templates. We are aware that SPATEL is not suitable for business experts (without technological background) to describe services. However, it was outside of the scope of this project to survey current languages for semantically description of web services and possible implementation for the chosen language.

6.4 Future work

We identified the following topics for future work:

- **Natural Language recognition:** Although we were not able to integrate the natural language recognition mechanism with our prototype, our recommendation is to create another *interaction type* for this specific case. This solution should automatically associates the response of a natural language query to a service type. The mechanism and description samples were translated from Dutch to English and they can be used for new experimentation.
- **Service Execution and prototype structure:** Because the purpose of the prototype is to serve as a proof of concept, some work is necessary to finalize the execution phase. We recommend a restructuration of the software to make it more coherent and

modular, this implies in significant changes in the current front-end part of the solution.

- **Web Service Semantic Description:** we suggest that some research is performed to identify which web service semantic description would better fit the DynamiCoS in the e-government context. An interpreter for the chosen language needs to be developed. After that, it would be necessary to define methods and tools to allow users with domain knowledge (civil servants) to add services to the repository using the chosen description language.
- **Domain ontology:** The domain ontology of public services we used for our solution was useful for testing purposes. However, a proper ontology for the e-government domain should be defined. We also suggest exploring further the reasoning capabilities of ontologies. In our experiment we only used taxonomic structures.
- **Session management:** For the personalization aspect of the e-government portal, the user has to be authenticated. In our experiments, we took the authentication for granted. DynamiCoS already offers a session management [65] capability that eventually could be used to retrieve information from the user's profile and automatically suggest values for service inputs.
- **Forward chaining of service** - Although the interaction pattern for citizen and civil servant are quite similar, the main distinction between them is the possibility of the civil servant to use his/her domain knowledge to add services that may be relevant to the citizen's goals (forward chaining). Therefore, we suggest adding and testing the forward composition, which is also supported by DynamiCoS, in an improved citizen *interaction pattern*.
- **Life event hypothesis** – We suggest a research to prove the hypothesis that life events are suitable for improving public services, since our solution framework can be used as the testing environment.

References

- [1] J. Ward and J. Peppard, *Strategic Planning for Information Systems*, 3rd ed. Bedfordshire: John Wiley & Sons, LTD, 2002.
- [2] M. Papazoglou and P. Ribbers, *E-business : organizational and technical foundations*. Chichester, England ; Hoboken, NJ: John Wiley, 2006.
- [3] J. E. Kingdom, *The Civil Service in Liberal Democracies*. London: Routledge, 1990.
- [4] D. G.-J. v. Lochem, "Overheid en Xml:InformaTiekeTens van en naar de burger (In Dutch)," *Element Jaargang*, vol. 15, pp. 25-26, 2009.
- [5] S. C. J. Palvia and S. S. Sharma, "E-Government and E-Governance: Definitions/Domain Framework and Status around the World. ," presented at the ICEG 5th International Conference on E-Governance, In Agarwal, Ramana, V. V. (eds.) *Foundations of E-Government.*, 2007.
- [6] Gemeente-Enschede. (2010, 23th September). *Welkom in Enschede*. Available: <http://www.enschede.nl/>
- [7] Overheid.nl. (2010, 23 September). *De wegwijzer naar informatie en diensten van alle overheden*. Available: <http://www.overheid.nl/>
- [8] W. Hofman and M. v. Staalduinen, "Dynamic Public Service Mediation," ed. Delft: TNO, 2009.
- [9] W. Hofman and M. v. Staalduinen, "Public Service Experience: Concepten en Voorbeelden. (In Dutch)," TNO, Delft2009.
- [10] D. Fensel, *Enabling semantic web services : the web service modeling ontology*. Berlin: Springer, 2007.
- [11] M. P. Papazoglou, *Web services: principles and technology*: Pearson Education Limited 2008.
- [12] G. Alonso, *Web services : concepts, architectures, and applications*. Berlin ; New York: Springer, 2004.
- [13] M. Holtkamp, "Public Services Integration and Interoperability " Msc, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, 2006.
- [14] D. Fensel, *Implementing semantic web services*. New York: Springer, 2008.
- [15] E. Gonçalves da Silva, *et al.*, "Dynamic Composition of Services: Why, Where and How.," presented at the International Workshop on Enterprise Systems and Technology (IWEST 2008), Enschede, The Netherlands, 2008.
- [16] B. Srivastava and J. Koehler, "Web Service Composition - Current Solutions and Open Problems," *ICAPS Workshop on Planning for Web Services*, 2003.
- [17] F. Lécué, *et al.*, "A Framework for Dynamic Composition of Services. 2nd Workshop on Emerging Web Services Technology (WEWST)," in *The 5th IEEE European Conference on Web Services (ECOWS)*, ed Halle, Germany Springer-Birkhauser, 2007.
- [18] W. Shena, *et al.*, "An agent-based service-oriented integration architecture for collaborative intelligent manufacturing," *Robotics and Computer-Integrated Manufacturing* vol. 23, pp. 315-325, 2007.
- [19] E. Gonçalves da Silva, *et al.*, "On the Design of User-centric Supporting Service Composition Environments," *IEEE ITNG*, 2010.
- [20] R. M. Pessoa, *et al.*, "Enterprise Interoperability with SOA: a Survey of Service Composition Approaches " presented at the International Workshop on Enterprise Interoperability (IWEI), part of the IEEE EDOC Conference, Munchen, Germany, 2008.

- [21] E. Gonçalves da Silva, *et al.*, "Supporting Dynamic Service Composition at Runtime based on End-user Requirements.," in *User Generated Services Workshop, International Conference on Service Oriented Computing (ICSOC)*, ed. Stockholm, Sweden, 2009, pp. 23-27.
- [22] H. Berndt, *et al.*, *The TINA Book*. London: Prentice Hall Europe, 1999.
- [23] R. J. Wieringa, "Title," unpublished].
- [24] H. V. d. Linde, *et al.*, "A systematic literature review of the effect of different prosthetic components on human functioning with a lower-limb prosthesis," *Journal of Rehabilitation Research and Development* vol. 41, pp. 555-570, 2004.
- [25] H. T. Davies and I. K. Crombie. (2009) What is a systematic review? *Evidence-Based medicine*. 1-8.
- [26] M. Shiaa, *et al.*, "Towards the Automation of the Service Composition Process: Case Study and Prototype Implementations.," presented at the ICT Mobile and Wireless Communications Summit (ICT-MobileSummit 2008) Stockholm, Sweden, 2008.
- [27] F. Kordon and A. J. Luqi, "An introduction to Rapid System Prototyping," *IEEE Transaction on Software Engineering*, vol. 28(9), pp. 817–821, 2002.
- [28] K. Woodford and G. Jackson, "Cambridge Advanced Learner's dictionary ", ed. Cambridge: Cambridge University Press., 2003.
- [29] H. T. T. Nguyen and T. Obi, "Government Transformation: The first Step to Integrate E-business into E-government," in *Integrating E-business Models for Government Solutions*, S. Chhabra and M. Kumar, Eds., ed Hershey: IGI Global, 2009.
- [30] S. Coleman, "Foundation of Digital Government " in *Digital Government: E-government Research, Case Studies, and Implementation*, ed: Springer, 2008.
- [31] T. Hauschild, "Top Priority E-Government: Guidelines for heads of public agencies," B. f. S. i. d. Informationstechnik, Ed., ed. Bonn, Germany, 2004.
- [32] UN, "United Nations e-Government Survey," D. o. E. a. S. A. D. f. P. A. a. D. Management, Ed., ed. New York: United Nations, 2008.
- [33] T. N. Clark and K. Hoggart, "City governments and their citizens: an overview of the book " *Citizen Responsive Government*, vol. 8, pp. 1-24, 2000.
- [34] G. Debord, *The Society of the Spectacle*. Paris, 1968.
- [35] B. Scheer, "Accessible e-government: Guidelines for decision-makers, designers and programmers," B. f. r. S. i. d. Informationstechnik, Ed., ed. Bonn, Germany, 2003.
- [36] G. Bayens, "E-government in The Netherlands," *via-nova-architecture.org*, pp. 1-12, 2006.
- [37] M. K. a. M. B. A. Leben, "Evaluation of life-event portals: Trends in developing e-Services based on life-events," in *Proceedings of the 4th European Conference on e-Government 'Towards Innovative Transformation in the Public Sector'* Dublin, 17-18, Jun. 2004, pp. 469-480.
- [38] A. Leben and M. Vintar, "Life-Event Approach: Comparison between Countries " in *Electronic Government*. vol. 2739/2003, ed Berlin / Heidelberg: Springer 2004, p. 1057.
- [39] H. Isselhorst, "Classification scheme for e-government procedures," B. f. r. S. i. d. Informationstechnik, Ed., ed. Bonn, Germany, 2001.
- [40] R. Leenes, "Local e-Government in the Netherlands: From Ambitious Policy Goals to Harsh Reality," *ITA Manuscript, Austrian Academy of Sciences*, 2004.
- [41] M. Janssen, Kuk, G. and R. W. Wagenaar, "A survey of e-government business models in the Netherlands.," presented at the ICEC'05, Xi'an, China, 2005.
- [42] ActiveBPEL. (2010, 27th March). *BPM - Business Process Management Software | Business Process Management Suite*. Available: <http://www.activevos.com/>

- [43] EclipseBPEL. (2010, 27th March). *BPEL Project home*. Available: <http://www.eclipse.org/bpel/>
- [44] Oracle. (2010, 27th March). *BPEL Process Manager*. Available: <http://www.oracle.com/technology/products/ias/bpel/index.html>
- [45] E. Gonçalves da Silva, *et al.*, "Defining and Prototyping a Life-cycle for Dynamic Service Composition " presented at the 2nd Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC), part of the 3rd International Conference on Software and Data Technologies (ICSOFT), Porto, Portugal, 2008.
- [46] G. B. A. Urbietta, J. Parra, and A. Uribarren. , "A survey of dynamic service composition approaches for ambient systems. ," presented at the First International Conference on Ambient Media and Systems (Ambi-Sys 2008)- First Workshop on Software Organisation and Monitoring of Ambient Systems (ICST), Somitas, 2008.
- [47] E. Gonçalves da Silva, *et al.*, "A Framework for the Evaluation of Semantics-based Service Composition Approaches," presented at the IEEE European Conference on Web Services (IEEE ECOWS), Eindhoven, The Netherlands, 2009.
- [48] K. Verma, *et al.*, ""The METEOR-S Approach for Configuring and Executing Dynamic Web Processes"," 2005.
- [49] J. Rao and X. Su, "A survey of automated web service composition methods," in *International Workshop on Semantic Web Services and Web Process Composition 2005*, pp. 43–54.
- [50] D. Fensel and C. Bussler, "The Web Service Modeling Framework WSMF," *Electronic Commerce Research and Applications*, vol. 1, 2002.
- [51] A. A. Patil, *et al.*, "METEOR-S Web Service Annotation Framework," in *Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA 2004.
- [52] K. Sivashanmugam, *et al.*, "Framework for Semantic Web Process Composition," The University of Georgia, Athens2003.
- [53] LSDIS. (2010, 09/07). *Large Scale Distributed Information Systems*. Available: <http://swp.semanticweb.org/>
- [54] TNO. (2010, *Knowledge for Business*. Available: <http://www.tno.nl/>
- [55] Lucene. (2010, 17 April). *Apache Lucene - Overview* Available: <http://lucene.apache.org/java/docs/>
- [56] Luke. (2010, 22/07). *Luke - Lucene Index Toolbox*. Available: <http://www.getopt.org/luke/>
- [57] D. Raggett, *et al.* (1999, 01/05). *HTML 4.01 Specification*. Available: <http://www.w3.org/TR/html401/>
- [58] T. Bray, *et al.* (2008, 09/05). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available: <http://www.w3.org/TR/REC-xml/>
- [59] Php. (2010, 01/05/). *What is PHP?* Available: <http://www.php.net/>
- [60] php.net. (2010, *Document Object Model*. Available: <http://php.net/manual/en/book.dom.php>
- [61] A. White, *JavaScript Programmer's Reference*: Wrox Press 2009.
- [62] B. Bos, *et al.* (2008, 06/07). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Available: <http://www.w3.org/TR/CSS2/>
- [63] T. A. S. Foundation. (2010, 07/07). *The Apache HTTP Server project*. Available: <http://httpd.apache.org/>
- [64] E. Gonçalves da Silva, *et al.*, "An Algorithm for Automatic Service Composition," presented at the 1st Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC), part of the 2nd International Conference on Software and Data Technologies (ICSOFT), Barcelona, Spain, 2007.

- [65] E. Gonçalves da Silva, "PhD thesis. ," ed. Enschede: University of Twente, 2011.
- [66] R. Khadka and B. Sapkota, "An Evaluation of Dynamic Web Service Composition Approaches," in *Proceeding of the 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, ACT4SOC 2010*, Athens, Greece, 2010, pp. 67-79.
- [67] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley, 1995.
- [68] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications.," *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.
- [69] T. M. M. Swe and N. S. M. Kham, "Ontology-Based Medical Diagnostic Knowledge Structuring Using Case-Based Reasoning Methodology," in *ICIS*, ed. Seoul, Korea, 2009, pp. 24-26.
- [70] G. Guizzardi, "Ontological Foundations for Structural Conceptual Models," Phd, Center for Telematics and Information Technology, University of Twente, Enschede, 2005.
- [71] C. Felden and M. Linden, "Ontology-Based User Profiling " in *Business Information Systems*, ed Heidelberg: Springer Berlin, 2007, pp. 314-327.
- [72] T. Robal and A. Kalja, "Applying User Profile Ontology for Mining Web Site Adaptation Recommendations.," presented at the ADBIS 2007. 11th East-European Conference on Advances in Databases and Information Systems, Varna, Bulgaria, 2007.
- [73] G. Booch, *et al.*, *The unified modeling language user guide*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2005.
- [74] A. J. Klievink, *et al.*, "An Event- driven Service-Oriented Architecture for Coordinating Flexible Public Service Networks.," presented at the Electronic Government, 7th International Conference, Turin, Italy, 2008.
- [75] V. Peristeras and K. Tarabanis, "Reengineering the public administration modus operandi through the use of reference domain models and Semantic Web Service technologies," in *The Semantic Web meets eGovernment (SWEG)*, California, USA, 2006
- [76] I. Trochidis, *et al.*, "An Ontology for Modeling Life-Events," in *Proceedings of the IEEE 2007 International Conference on Services Computing*, 2007.
- [77] P. Salhofer, *et al.*, "Ontology Driven E-Government," presented at the Fourth International Conference on Software Engineering Advances, Porto, Portugal 2009.