Ph.D. Thesis

Knowledge Modelling for Service-oriented Applications in the e-Government Domain

Candidate: Alessio Gugliotta

Supervisor: Vito Roberto

April 24, 2006
To Stefania.
The current trends in e-Government applications call for joined-up services that are simple to use, shaped around and responding to the needs of the citizen, and not merely arranged for the provider’s convenience. In this way, the users need have no knowledge of - nor direct interaction with - the government entities involved.

One of the possible solutions, which has recently become a popular development practice of e-Government programmes, is the adoption of service-oriented systems. The application of the service-oriented model to Web resources to provide a loosely coupled model for distributed processes is manifested by Web Services. In this way, services are autonomous platform-independent computational elements that can be described, published, discovered, orchestrated, and programmed using XML artifacts for the purpose of developing massively distributed interoperable application. Despite the advance in the use of standards for Web Service description (e.g. WSDL) and publishing (e.g. UDDI), the syntactic definitions used in these specifications do not completely describe the capability of a service and cannot be understood by software programs. It requires a human to interpret the meaning of inputs, outputs and applicable constraints as well the context in which services can be used.

The Semantic Web aims to alleviate interoperability and integration problems. By allowing software agents to communicate and understand the information published, the Semantic Web enables new ways of consuming services. In particular, Semantic Web Services (SWS’s) technology provides an infrastructure in which new services can be added, discovered and composed continually, and the organization processes automatically updated to reflect new forms of cooperation. It combines the flexibility, reusability, and universal access that typically characterize a Web Service, with the power of semantic mark-up, and reasoning in order to make feasible the invocation, composition, mediation, and automatic execution of complex services with multiple conditional paths of execution, and nested services inside them.

Although still at a premature level of development, the SWS’s provide a promising infrastructure for next generation e-Government services. However, the integration between e-Government applications and SWS’s is not an easy task.

In this thesis, we present an approach of Knowledge Management (KM) based on SWS’s and specific e-Government requirements. We argue that a more complex semantic layer for managing government services needs to be modelled – and a middleware system designed on such a model – in order to meet the requirements of real-life applications.

The aim of our work is to provide an approach and define such a model. As a result, we provide a framework with which most Public Administrations can identify, from which they can work when designing and delivering e-Government services. This general framework can be adapted and applied as appropriate. The approach is grounded on a technological paradigm capable to fit a general distributed organization of knowledge, with focus on the supply of services.

We show two applications of the obtained framework. The first one involves the use and transfer of described knowledge. The second one takes advantage of the meta-ontologies that compose our framework for defining a knowledge elicitation methodology.
Acknowledgments

First of all, I would like to express my gratitude to my advisor prof. Vito Roberto that with his experience and wisdom introduced and guided me – as a big brother can do – in these years. My gratitude also goes to prof. Carlo Tasso and the people of the Artificial Intelligence Laboratory (Matteo, Christian, Alan, Marco, Massimiliano, Nello, Andrea) that gave me a lot of advices in the first years of my PhD experience.

Thanks to all of my university course mates Simona, Luca, Fulvio, Speed, Simone, Luca D.M., Augusto wherewith I shared a lot of the days at our department.

A special special thanks to Stefania. She provides me an enormous energy, which really helped me in the last months of this thesis work.

I am grateful to all the people that I met in my experience in England. Firsts of all Enrico Motta and John Domingue that gave me the opportunity to work with them, then Liliana and Dnyanesh that often addresses me in my work, and at least but not last all of the KMi people: Simona, Michele, Vanessa, Vlad, Roberto, Neil, Barry, Carlos, Tom, Mark, George, Dinar, Chiara, Nanda e Vicky.

Finally, I must thank my family and all of my friends for the untiring support and unconditional trust. I owe most of what I have learned to all these people and I will never be able to thank enough.
Contents

Introduction 1
  I.1 Quick Overview of Knowledge Management ................. 1
  I.2 Knowledge Management and e-Government ................. 6
    I.2.1 Knowledge Modelling within e-Government ........... 6
    I.2.2 Innovative Systems supporting the KM within e-Government:
      Service-Oriented Systems ............................ 6
  I.3 Limitations of existing approaches
      and research problems .................................... 7
  I.4 Research Approach ...................................... 8
  I.5 Thesis Contributions ................................... 9
  I.6 Thesis Organization .................................... 10
  I.7 Relation to previous published work of the author ........ 11

I First Part 13

  1 Ontologies 15
    1.1 What is an ontology .................................... 15
      1.1.1 Informal definitions ................................ 15
      1.1.2 A formal definition ................................. 16
    1.2 Main components of an ontology ......................... 17
    1.3 Criteria for designing ontologies ....................... 18
    1.4 The life-cycle for developing ontologies ............... 19
    1.5 Methodologies for building ontologies ................. 20
    1.6 Classification ........................................ 22
    1.7 Formalisms for representing ontologies ................. 23
      1.7.1 Logic ............................................. 23
      1.7.2 Semantic Nets ..................................... 23
      1.7.3 The description logic ............................... 24
      1.7.4 Frames .......................................... 24
      1.7.5 Conceptual Graphs ................................. 25
    1.8 Application fields of ontologies ......................... 25
    1.9 Languages for Building Ontologies ....................... 27
      1.9.1 OCML .......................................... 28
      1.9.2 OWL ........................................... 28
    1.10 Comparison of languages ................................ 31
    1.11 Using ontology languages in applications .............. 33

  2 Knowledge Modelling of Services 35
    2.1 Web Services ........................................... 35
    2.2 Semantic Web .......................................... 37
    2.3 Semantic Web Services .................................. 39
      2.3.1 The usage activities of a SWS ..................... 40
      2.3.2 The architecture for SWS ............................ 40
      2.3.3 The service ontology of SWS ....................... 41
    2.4 Languages for describing Web Services .................... 41
      2.4.1 A language for describe Web Services: WSDL ....... 41
<table>
<thead>
<tr>
<th>2.4.2</th>
<th>Ontology-based Web Service Description Languages</th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>IRS-III Framework</td>
<td>47</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Solving mismatch problems: data, process, and goal mediation</td>
<td>48</td>
</tr>
</tbody>
</table>

3 Approaches to e-Government

3.1 Web Information Systems in E-Government
   3.1.1 eGov Project | 51 |
   3.1.2 EU-PUBLI.com Project | 51 |

3.2 Ontologies in E-Government
   3.2.1 SmartGov Project | 52 |
   3.2.2 ICTE-PAN Project | 54 |

3.3 Semantic Web Service technology in E-Government
   3.3.1 OntoGov Project | 54 |
   3.3.2 Terregov | 56 |

3.4 What needs to be done? | 57 |

II Second Part

4 A Semantic-Based Framework

4.1 Requirements | 61 |

4.2 e-Government Conceptual Models | 64 |

4.3 Meta-Modelling the Conceptual Model
   4.3.1 Ontologies for Meta-Modelling | 70 |
   4.3.2 OCML as a knowledge modelling tool | 73 |

4.4 The Ontological Framework
   4.4.1 Core Life Event Ontology (CLEO) | 74 |
   4.4.2 Domain Ontology | 76 |
   4.4.3 Service Ontology | 77 |
   4.4.4 Ontology Stack | 77 |

5 Formalizing the Ontological Framework

5.1 The Core Life Event Ontology (CLEO)
   5.1.1 Life Event Description Module | 79 |
   5.1.2 Static Descriptions | 81 |
   5.1.3 Dynamic Descriptions | 83 |

5.2 The Domain Ontology
   5.2.1 Building the Domain Ontology | 95 |
   5.2.2 The Generic Level of the Domain Ontology | 97 |

5.3 The Service Ontology
   5.3.1 The Goal Ontology module | 98 |
   5.3.2 The Web Service Ontology module | 100 |
   5.3.3 The Mediation Ontology module | 101 |

6 A case study in applying the approach.

6.1 Capturing Requirements | 103 |

6.2 The methodology | 104 |

6.3 A Change of Circumstance Case Study
   6.3.1 Overview of use case | 105 |

6.4 Live event and actor analysis | 107 |

6.5 Viewpoint analysis
   6.5.1 State of Affairs analysis | 108 |
   6.5.2 Interaction analysis | 109 |
   6.5.3 Conception analysis | 111 |
Contents

6.5.4 Plan analysis ......................................................... 117
6.6 Model Specific Scenario ............................................... 119
6.7 Create the SWS descriptions ........................................... 120
6.8 Summary ............................................................... 123

7 A Semantically-Enhanced Middleware 125
7.1 The proposed Service-Oriented System Architecture ............... 125
7.2 The semantically-enhanced Middleware ................................ 126
7.3 Inferences on the ontological framework ............................. 127
    7.3.1 Moving House Life Event ..................................... 127
    7.3.2 Use Case 1: Invoking a service ............................... 128
    7.3.3 Use Case 2: exploring the scenario description ............ 133

Conclusions 135

Bibliography 139
# List of Figures

1. The general Knowledge Model. .................................................. 4
2. Structure of the Thesis ............................................................... 10

1.1 Relationships between knowledge representation and reasoning components of an ontology ................................................................. 18
1.2 General life-cycle for building an ontology ........................................... 20
1.3 Example of Semantic Net: the collocation of the blue whale in the animal kingdom. ................................................................. 24
1.4 Display Form ............................................................... 25
1.5 Ontology Language Classification .................................................... 27

2.1 Web Services Architecture ........................................................... 36
2.2 Web Services enabling standards ..................................................... 36
2.3 Semantic Web’s Layer cake ............................................................. 38
2.4 Top level of the OWL-S service ontology ............................................. 44
2.5 Meta-ontology for SWSs: WSMO [32] ................................................. 45
2.6 WSMO Web service Interface ......................................................... 46
2.7 WSMO Mediator Structure ............................................................. 47
2.8 Data Mediation ............................................................. 48
2.9 Mediation between a Goal and a Web Service. Two inputs of Goal are transformed into one input of the Web Service. ........................................ 48
2.10 Mediation between a Goal and Web Service. One input of the Goal is transformed in 2 inputs of the Web Service. ........................................ 49
2.11 Mediation between two sub-goals. The input of goal 1 is transformed in one input of goal 2. ................................................................. 49

4.1 Meta modelling approach .............................................................. 63
4.2 The government service-supply scenario conceptual model. .................. 64
4.3 The e-Government system conceptual model. ........................................ 65
4.4 The way in which public services are grouped and integrated into life events. ................................................................. 66
4.5 Life event as point of contact among different actor’s viewpoint. ................ 67
4.6 Plan and Interaction models ............................................................ 68
4.7 Polovina’s accounting model. The concepts are represented by rectangles and relations by circles; the direction of the arrows determines the direction of the reading. A concept that appears with a specific referent (e.g. Economic Resource: *c) represents an instance. ................................................................. 69
4.8 Use of meta-models for cooperative building of application-specific knowledge level. ................................................................. 70
4.9 Taxonomy of DOLCE basic categories ................................................. 71
4.10 Description and Situation ............................................................. 72
4.11 The proposed ontological framework ................................................. 75
4.12 Ontology Stack ............................................................. 78

5.1 Multi-viewpoint structure of CLEO .................................................... 80
5.2 The Life Event Description Module. Top gray boxes represent elements of the D&S ontology. Bottom gray boxes represent elements of other CLEO modules. ................................................................. 82
5.3 The State of Affair Description Module. Gray boxes represent elements of the D&S ontology. ................................................................. 83
5.4 Top-level taxonomy for the descriptive entities and the main relations between them. ................................................................. 84
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>The Conception Description module. Gray boxes represent elements of other CLEO modules.</td>
</tr>
<tr>
<td>5.6</td>
<td>The Plan Description module and the goals-plan-description example. Top-right gray boxes represent concepts of the D&amp;S ontology. Mid-left gray boxes represent elements of the Conception Description Module for the goals-description-plan example. The three bottom classes aggregate control and description constructs of a plan.</td>
</tr>
<tr>
<td>5.7</td>
<td>The goals-plan-description example.</td>
</tr>
<tr>
<td>5.8</td>
<td>The goals-plan-description example: the highlighted arrows show the new flow of the plan.</td>
</tr>
<tr>
<td>5.9</td>
<td>The Interaction Description module.</td>
</tr>
<tr>
<td>5.10</td>
<td>The state of a transition (gray boxes) as a shared definition.</td>
</tr>
<tr>
<td>5.11</td>
<td>The transition resource description. Dashed arrows represent the backward inference path for obtaining elements of the viewpoint A, starting from the resource element.</td>
</tr>
<tr>
<td>5.12</td>
<td>Using the Interaction Descriptions Module (gray boxes) within a multi-viewpoint scenario.</td>
</tr>
<tr>
<td>5.13</td>
<td>Links between Need and Offer Descriptions through the Interaction Description (gray boxes).</td>
</tr>
<tr>
<td>5.14</td>
<td>Using the transition event descriptions (gray boxes) to link goals and services of two distinct viewpoints.</td>
</tr>
<tr>
<td>5.15</td>
<td>The two-levels structure of the Domain Ontology.</td>
</tr>
<tr>
<td>5.16</td>
<td>The three-stage process for creating the generic level of the Domain Ontology. We distinguish four ontologies: DOLCE (and its module D&amp;S) represented by the gray ovals; the DOLCE e-government mapping ontology represented by white ovals; CLEO and Domain Ontology, extending the previous ontology with bold ovals.</td>
</tr>
<tr>
<td>5.17</td>
<td>Intersection between the knowledge levels modeled by the Service Ontology and CLEO; the former represents the service delivery knowledge level (SWS), the latter the configuration knowledge level (e-Government application). The lines labeled with &quot;goal&quot; and &quot;service&quot; represent the belongings of these two concepts to both knowledge levels.</td>
</tr>
<tr>
<td>5.18</td>
<td>Aligning web service concepts of the Service Ontology: the class WSMO-Web-service is defined as sub-class of both CLEO and WSMO web service descriptions. Further SWS approaches simply adopt the same alignment mechanism.</td>
</tr>
<tr>
<td>5.19</td>
<td>Mapping the CLEO service decomposition to WSMO orchestration by means of GG and WW mediators.</td>
</tr>
<tr>
<td>6.1</td>
<td>Reference domain ontologies for the considered viewpoints.</td>
</tr>
<tr>
<td>6.2</td>
<td>The state of affair analysis.</td>
</tr>
<tr>
<td>6.3</td>
<td>The interaction analysis.</td>
</tr>
<tr>
<td>6.4</td>
<td>Identified interactions between the case worker, community care, and housing department viewpoints.</td>
</tr>
<tr>
<td>6.5</td>
<td>The conception analysis.</td>
</tr>
<tr>
<td>6.6</td>
<td>Links between Need and Offer Descriptions through the Interaction Description (gray boxes).</td>
</tr>
<tr>
<td>6.7</td>
<td>The plan analysis.</td>
</tr>
<tr>
<td>6.8</td>
<td>Creating the SWS descriptions.</td>
</tr>
<tr>
<td>6.9</td>
<td>Sample structure of WSMO descriptions for the list service example.</td>
</tr>
<tr>
<td>6.10</td>
<td>Change of circumstance case study: the context ontologies.</td>
</tr>
<tr>
<td>6.11</td>
<td>Change of circumstance case study: the Domain and Service ontologies.</td>
</tr>
<tr>
<td>7.1</td>
<td>The semantically-enhanced infrastructure of a service-oriented system.</td>
</tr>
<tr>
<td>7.2</td>
<td>A generic inference path in CLEO from the user viewpoint.</td>
</tr>
<tr>
<td>7.3</td>
<td>The WfMS loads the needs plan description.</td>
</tr>
<tr>
<td>7.4</td>
<td>The WfMS loads the goals plan description and interacts with IRS-III.</td>
</tr>
</tbody>
</table>
7.5 A generic inference path in CLEO from the provider viewpoint.
## List of Tables

1.1 Definition of the main components of domain knowledge (left side) and reasoning mechanisms of OCML and OWL (right side) ............................................................................. 32

4.1 Life Events proposed by Italian Government ........................................................................ 66

6.1 Descriptive entities of the state of affair description case-worker-PMH-assessment-initial-SOA ......................................................................................................................... 111
Introduction

In this thesis, we investigate the aspects connected to the application of Knowledge Management (KM) techniques within service-oriented systems, in order to supply add-value services. In particular, we focus on the e-Government domain.

The importance of KM is being increasingly recognized in the public sector, in relation to e-Government realizations. Every Public Administration (PA) makes use of knowledge to increase the productivity of its activities. Therefore, efficient, scalable and flexible KM Systems (KMS) and coherent strategies are needed to support PA’s. Nonetheless, due to a debatable success of current KM implementations, it is still unclear how such topics should be addressed in highly distributed and heterogeneous environments [97].

From the other side, the Web and other Information and Communication Technologies (ICT) provide an environment to supply services in a virtual world. Organizations can provide their services globally and users from all over the world obtain access to these products. However, the heterogeneous and continually changing environment leads to several problems related to finding suitable services that can satisfy the user’s needs.

Moreover, the semantic technologies are providing a powerful and agile means for KM approaches [90], and aim to solve integration and interoperability issues.

In the following, we briefly introduce the foundation concepts driving our work.

I.1 Quick Overview of Knowledge Management

Knowledge

As it is difficult to define KM, it is also difficult to define Knowledge, that is the “under-layer” concept KM has to manage [62]. In fact, knowing about a domain or a concept does not simply mean holding pertinent information; we should also consider the capabilities to use, compare, and apply such a information [67]. For instance, the gathered and held data of an organization increase their value when they can be associated to the goals and needs of the organization itself, and used at the proper time. Thus, an information becomes knowledge if and only if it is understood and its value is transmitted by means of a process of learning. We introduce the following classification of knowledge [95]:

- **Tacit Knowledge**: it is the set of competences, opinions, and capabilities that beget the action and the thought of a person or a group of persons. It is unvoiced knowledge, complex characteristics of an individual that can not be easily structured. It refers to context and personal mental models. For these reasons it is difficult to formalize and communicate.

- **Explicit Knowledge**: it is documented and recorded information that can be structured in data. For this reason they can be easily diffused.

One of the main aspects connected to the KM is the transformation from tacit to explicit knowledge. Summarizing, the Knowledge can be seen such as:

- the effective use of data and information, and the set of the possible capabilities and competences, ideas, intuitions, experiences, comments, and motivations of the people;

- the processes of comprehension of data, information, and transformation from tacit to explicit and reusable knowledge;

- the acquired awareness and conscience originated from a continuous process of learning and based on acquiring concepts about logic and experience domains.
Knowledge Representation

A resource is manageable, reproducible, and communicable if it is standardized. Standardization is the process that purifies a resource from its contextual elements and generalizes it, and following formal criterion usable in every context.

The standardization concept can be also applied to the knowledge, bringing to the standardization of the knowing. This view is actualized with the idea of Knowledge Base (KB), that is the coded collection of the knowledge of an organization, and the concept of Knowledge Representation (KR), that is an abstract and simplified view of the knowledge.

The aim of the KR is to develop usable formalisms that represent the necessary knowledges within a system. Generally, a KR system satisfying the above request should consist of:

- a representation language, that is a set of syntactic structures adequate to code information that should be represented, and useful to be implemented in the memory of a computer. It should unambiguously represent any interpretation of a statement (logical adequacy), hold a translation method from natural language to its internal representation, and allow the reasoning [98];

- a set of rules, or operations, that allow to manage such syntactic structures, following the meaning that has been associated to them. The rules should be formulated as effective procedures and their application should allow to obtain the intended reasoning.

In the following, we report a formal definition of KR [26], where the latter is described in terms of roles that it plays:

- **KR is a surrogate**: it is a surrogate of the knowledge, that allows an entity reasoning on it to obtain results rather than perform actions.

- **KR is a set of ontological commitments**: it describes a domain, highlighting only the main concepts and neglecting the less important ones.

- **KR is a fragmentary theory of intelligent reasoning**: it concerns aspects that answer to the following questions: What might we infer from what we already know? What should we infer from what we already know?

- **KR is a medium for efficient computation**: since the machine reasoning is a computational process based on the KR, the efficiency is an important aspect of KR.

- **KR is a medium of human expression**: it is a machine language that expresses concepts connected to a human domain.

Some classes of KR languages are:

- **Frame Based.** A frame or class is a specific concept/object of a domain.

- **Description Logic.** It is a family of languages that represent the knowledge of a domain in a structured and well formalized way. These languages are different from other classes of languages because they have a logic based and formal semantic [8].

- **Production Systems.** They are based on the “if-then rules” expressions.

- **Database Systems.** They only offer representations of simple assertions, without inference.

The knowledge representation is the main element of this thesis. The following two subsections specify which particular aspects we consider.
I.1. Quick Overview of Knowledge Management

Ontologies

A formalization of the knowledge is based on a conceptualization, that is a representation of the set of objects, concepts, and other entities existing in a particular domain of interest (universe of discourse) and the relations among them. Thus, a conceptualization is an abstract and simplified description of the reality that has to be represented (conceptual model). Every knowledge base, knowledge base system, or agent working at the knowledge level is linked to an implicit or explicit conceptualization. An explicit specification of a conceptualization is an ontology.

In the literature, the term “ontology” presents multiple definition that also refer to some philosophical concepts. In our context, we refer to ontology as: the description of some concepts and relations between them, within a specific domain [49]. In an ontology, the universe of discourse is associated to human-understandable scripts describing the meaning, and formal axioms restricting the interpretation and describing the right utilization [50].

An ontology is a set of statements describing “ontological commitments”, highlighting only the main things of a domain. Thus, an ontology is a particular and explicit specification allowing knowledge representation and enabling different systems (agents) to communicate exchanging inquiries and assertions about a specific domain, and using the same vocabulary. It does not means that all the agents sharing the same vocabulary own the same knowledge (Knowledge Base): an agent can share the vocabulary but do not know a term of it. An ontology introduce consistency but not completeness.

How an ontology describes a domain depends on the chosen language of representation. Many ontology implementation languages have been created and other general Knowledge Representation (KR) languages and systems have been used for implementing ontologies, even though these were not specifically created with this purpose.

In this thesis, we present the ontologies as useful tools for knowledge modelling. We introduce the main concepts, the design methodologies, and the main steps of an ontology life-cycle. Finally, we provide a briefly view of the main developing methodology and formalisms for representing ontologies.

Service Representation

An emerging field in the knowledge representation is the representation of services. The quick development of service-oriented applications is leading to several attempts for conceptualizing the world of services. The following aspects are considered in a conceptualization of service:

- **Clarifying the meaning of “service”**. This term is being used in quite different ways in various domains. For instance, in the technical domain it is usually considered as a particular software system that can be published, located, and invoked. In the business domain it has a much broader and abstract meaning.

- **Clarifying the difference between a service and its public description**. A service is really available if information on how to access it is made public. In some cases services are confused with their public descriptions.

- **Clarifying the distinction between “simple” and “composite” services**. The former refers, respectively, to services that do not rely on the execution of others, and services that do so.

- **Identifying the various stakeholders that exploit, offer, and manage services**. Various actors are involved in a service-oriented system. Beside the usual roles of service provider and consumer, other important roles should be considered (e.g. mediators who compose or certify services, support service discovery, etc.).

- **Capturing the relevant aspects concerning service discovery, composition, mediation, publication, execution, and monitoring**. These are the main research areas of the knowledge modelling of services.
In this thesis, we deepen some approaches for knowledge modelling of services. In particular, we focus on the promising technology of Semantic Web Services (SWS’s) that adopts ontologies for representing Web Services.

Knowledge Management

In the literature, most of the available definitions of KM refer to the set of activities gathering and diffusing the knowledge to benefit an organization and their individuals [62]. In the following we report a general definition, that is useful for our goals:

**Definition I.1: Knowledge Management** Any activity or process creating, acquiring, capturing, sharing and using knowledge, wherever it resides, with the aim of improving the knowledges and the performances within an organization [81].

Figure 1 shows the General Knowledge Model [7][95], that explains the flow of knowledge within an organization, arranging four phases:

- **Creation**: synthesis and insertion of new knowledge into the system; it concerns activities such as acquisition and elicitation of the knowledge. The creation shifts the focus toward the ways of helpfully use the knowledge; actually, a common mistake is to store knowledge without knowing how can use it.

- **Retention**: preservation and sharing of the knowledge into the system, allowing the storage and the access of different individuals. It means formalizing the knowledge, extending the range from an individual dimension to an organizational and inter organizational dimension.

- **Transfer**: management of the flow of the knowledge within the system; it concerns activities such as communication, conversion, filtering, and keeping of the knowledge.

- **Utilization**: application of the knowledge to one of the processes of the organization.

![Figure 1: The general Knowledge Model.](image)

In this thesis, we cover all of the four phases of knowledge management. We develop an ontological framework for the retention of the knowledge, and adopt such a framework as the base for (i) a knowledge-based middleware that allows the use and transfer of the retained knowledge, and (ii) a methodology that elicits knowledge for the scenario descriptions and early captures requirements for the SWS definitions.
I.1. Quick Overview of Knowledge Management

Knowledge-Based Middleware

A knowledge-based middleware is an integration technology that not only allows to link various systems providing services, but also understands the feature and the context of the applications it is connecting.

Generically, it is able to link many heterogeneous systems together, allowing them to share knowledge without changing the participating applications or data, and do so in a such a way as to optimize and mediate the flow of knowledge through a particular domain. In other words, it maps the knowledge management activities in a distributed and heterogeneous environment.

The generic architecture of a knowledge-based middleware is composed by four separate layers:

- **the messaging level** provides a transportation mechanism for messages to and from all of the source and target systems.
- **the transformation and formatting layer** provides the reformattting of information when they are moved from one system to another.
- **the business process and workflow layer** provides a mechanism that defines how information is moved throughout and between organizations and how the movement of information relates to business event.
- **the knowledge-oriented layer** provides a memory of expert behavior to the middleware to retain “experiences”.

The middleware we outline in this thesis instantiates the above architecture adopting the semantic technologies introduced above: ontologies and service representation models. Such a middleware is the core of a Knowledge Management System.

Knowledge Management Systems

The Knowledge Management Systems (KMS) are the set of technologies that allow an effective and efficient knowledge management within an organization [66]. They extend the architecture of a middleware with further modules (e.g. visualization tools, business intelligence tools, net infrastructures, etc.) using the functionalities provided by the middleware.

To quote an example, some objectives satisfied by KMS may be: (i) begetting, sharing, and applying knowledge; (ii) detecting domain experts; (iii) developing and enabling the participation to nets and community networks; (iv) exchanging knowledge between community networks; (v) increasing the employee capabilities about learning and understanding relationships between knowledge, people, and processes of the organization.

A complete classification that is based on the aims of KMS is provided in [21]; pertaining examples of KMS are the following:

- **Information mediator systems**; they supply an interface (usually a Web interface) where the users can query a specific domain, referring to distributed and heterogeneous knowledge sources but giving to the user the idea of a centralized and homogeneous system. These systems have the following features: the knowledge capturing and integrating processes are already happened (a priori) and they do not involve any interaction with the end-user; using languages for describing and indexing the knowledge sources and their contents. An example of these systems is PICSEL [1].

- **Ontology-based systems**; they use ontologies for representing and achieving goals. Examples of applications are the following: systems supporting decision tasks by means of an inference engine working on an ontology; conceptual models for the distributed KM; systems allowing the automatic research and research engines. An example of these systems is Planet-Onto [30].
6. Introduction

I.2 Knowledge Management and e-Government

The concepts introduced so far have been applied within the set of processes named e-Government.

The latter points out the set of activities that lead to the continuous improvement of the public service supplying and to the citizen participation at the politic actions. These activities involve the gradual transformation both the internal and external relations of the PA's. They are primed by the use of technological innovations, and in particular by Internet and other Information and Communication Technologies (ICT).

Every PA makes use of knowledge to increase the productivity of its activities. Part of the work of the PA's includes the processing of large amount of data, information and knowledge about citizens, businesses, other administrations, society, laws, political environments, etc. Besides this, often the output of the work of PA's is the creation of new information and knowledge. For allowing all this knowledge could be useful to an organization it should be captured, collected and shared with all employees and, in case, other organizations, in a way that it would be easily accessible, understandable and usable.

In this thesis, we focus on the following two research fields:

- Knowledge modelling of concepts, services, and processes of the e-Government domain.
- Service-oriented systems for the cooperation among different organizations and the development and integration of innovative services for citizens and businesses.

I.2.1 Knowledge Modelling within e-Government

The e-Government scenario is an obvious and promising application field for ontologies, since legislative knowledge is by nature formal to a big extent and it is definition shared by many stakeholders.

The use of ontologies – and in general of knowledge representation methods – can play an important role in the development of e-Government. The knowledge to be represented within a PA ranges from common organizational concepts to processes for supplying services to citizens and business. Therefore, an ontology can be an important support to the following activities [9],[5]:

- Systematic and standard description of information resources: documents, processes and their relations;
- Support to the automation of services, systems and infrastructures involving PA's;
- Supply of added-value services, like selected information retrieval and personalization of contents.

There are not ontologies that completely describe all the e-Government concepts, and the same concepts could be differently used among distinct PA's. In fact, every single PA could have not the same point of view of the domain and different interoperability needs with other PA's. The e-Government domain standardization can help, but it does not necessarily join the aim and the language of the several involved actors.

Particular interest has been devoted to the description of services supplied by PA to citizens and companies, since – to quote an example – in a European framework the services supplied in a Country turn out to be useful to citizens, companies and administrations of another Country.

I.2.2 Innovative Systems supporting the KM within e-Government: Service-Oriented Systems

The current trends in e-Government applications call for joined-up services that are simple to use, shaped around and responding to the needs of the citizen, and not merely arranged for the provider’s convenience. In this way, the users need have no knowledge of - nor direct interaction with - the government entities involved.
One of the possible solutions, which has recently become a popular development practice of e-Government programmes, is the adoption of Service-Oriented Systems. They enable building agile networks of collaborating business applications distributed within and across organization boundaries. Combined with recent developments in the area of distributes systems, workflow managements systems, business protocols and languages, services can provide the automated support needed for e-Government applications.

Thus, services need to be interoperable in order to allow for data and information to be exchanged and processed seamlessly across government.

Interoperability is a key issue in the development of current e-Government services. A working paper by the Commission of European Communities [73] emphasized its role, not only as a technical issue concerned with linking up computer networks, but also as a fundamental requirement to share and re-use knowledge between networks, and re-organize administrative processes to better support the services themselves. Still in [73], three levels of interoperability were individuated: technical, semantic and organizational. The first one refers to the topics of connecting systems, defining standard protocols and data formats. The second one concerns the exchange of information in an understandable way, whether within and between administrations, either locally or across countries and with the enterprise sector. The third one refers to enabling processes to co-operate, by re-writing rules for how Public Administrations (PA’s) work internally, interact with their customers, and use Information and Communication Technologies (ICT).

On practical grounds, the integration of services is a basic requirement of service-oriented systems, which aim at gathering and transforming processes - needed for a particular user - into one single service and the corresponding back-office practices. Beside the elimination of duplicate activities on the customer side and the substitution of documents required to be enclosed with application in reusing and sharing of information that already exists within the organizations, duplicated and useless activities within the back-office process are eliminated as well (e.g. every activity is performed only once, every data is stored only in one place if possible). The overall processing time and costs are markedly reduced and all aspects of service quality are thoroughly improved.

One of the key questions in developing service-oriented systems is how to structure and design services in order to improve their quality and efficiency and not make them even more time consuming and complicated.

### I.3 Limitations of existing approaches and research problems

Several KM approaches exist to represent PA knowledge, and a number of research projects is currently under progress [33], [36], [34], [85], [57], [76], [91]. They face more or less the same problems: there is no generic domain analysis for the overall PA system at any level of granularity; there are no generic PA models for processes and objects; there are no ontologies for modelling PA objects and relationships; there are no standard vocabularies for describing concepts. Due to this shortage, the researchers have to build from scratch PA ontologies to be used as test-beds for demonstrating the functionalities of their systems. The main focus of these initiatives is not to build a PA domain ontology, but rather to test and validate specific technological solutions. This results in proposing ad hoc description for the PA domain and far from being considered reusable.

Moreover, existing approaches usually address specific service-oriented models, where the provider’s point of view plays the central role. However, the e-Government scenario is composed by several actors. Each of them deals with different kind of knowledge, conceptions, processes; in other words they have different viewpoints. Such viewpoints differently influence and relate to the services.

The e-Government domain involves the supply of services which need to be integrated among multiple PA’s – or more generally actors. Each actor should keep its autonomy in the description of its domain. Integration will allow all actors to satisfy their needs by means of external services.
This includes looking for and identifying the right service.

Finally, SWS technology offers a promising solution for the interoperability and integration issues. However, they do not completely address these issues. Many services cannot be executed in one single step (one shot). This happens when there are multiple execution paths for the services, depending on the evidence provided by the user. In general way, the administration may repeatedly contact the user during the service execution, and ask for more evidence placeholders.

All of the following problem statements should be investigated in order to find a solution:

- How can a conceptual model be defined to capture the whole e-Government service-supply structure including all involved actors, relationships and routines? Can the model be used as a starting point (framework) for describing different project-domains?

- How can a reference model capture these multiple aspects in order to design and structure qualitative and efficient services (or SWSs)? Does it exist a methodology that drive the knowledge elicitation in all of the existing viewpoints?

- How can the different views be compared or be made comparable, in order to avoid misunderstandings between the parties? How can the reference model be adapted for a distributed scenario?

- Are there several ways to perform a need request? If there are, how can they be described in order to satisfy both users and providers?

- How can the shortcomings of current SWS approaches be addressed with respect to the integration and interoperability issues?

I.4 Research Approach

In this thesis, we present a KM approach based on SWS technology and the following e-Government requirements:

- the PA worker - and in general a domain expert - does not directly use the SWS infrastructure to represent knowledge internally. For instance, organizations will likely adopt their own workflow paradigm to describe their processes [28].

- The PA work routines involve interactions with non-software agents, such as citizens, employees, managers and politicians. Multiple viewpoints need to be considered.

- In real cases, component services are not atomic, and cannot in general be executed in a single-response step; they may require to follow an interaction protocol with non-software agents that involves multiple sequential, conditional and iterative steps. For instance, a service may require a negotiation between the user and the provider.

- Web service description is an important but restricted aspect of an e-Government service-supply scenario.

We argue that a more complex semantic layer for managing government services needs to be modelled - and a middleware system designed on such a model - in order to meet the requirements of real-life applications. In particular, we identify three knowledge levels:

- Guidelines, describing the context that bounds the creation and evolution of services: legislations, policies, and strategies influencing the development and management of an e-Government service-supply scenario.

- Configuration, describing the context in which services are supplied: requirements, resources, actor’s role, business processes, and transactions of an e-Government service-supply scenario.
I.5. Thesis Contributions

- **Service delivery**, adopting Semantic Web Service (SWS) technology as the base for the description, discovery, composition, mediation, and execution of (Web) services.

As a result, the integration of e-Government applications with SWS’s requires a framework which maps and combines the above knowledge levels. The aim of our work is to provide an approach and define such a framework.

## I.5 Thesis Contributions

In our thesis, we provide a framework with which most PA’s – or generally organizations – can identify, from which they can work when designing and delivering e-Government services. This general framework can be adapted and applied as appropriate. It is considered from two distinct dimensions: conceptual modelling and creating an infrastructure for semantic interoperability. The following main contributions regard the first dimension.

- **Contribution to the epistemology of the e-Government service supply-scenario.** The e-Government domain is complex. It cannot be described by a unique conceptual model. As a foundation of our work, we consider clear conceptual models describing distinct aspects connected to such a domain and providing a sound base for the definition of our framework.

  In our approach, we segment the knowledge identified by such conceptual models in the three knowledge levels introduced above. Moreover, we introduce the contextualization – i.e. describing various notions of context, non physical situations, topics, plans, beliefs, etc. as entities – of the e-Government scenario in terms of descriptions. In particular, we define two clean separations: *Context vs Services* and *Context vs Vocabulary*.

  The former distinguishes between the description of the environment where the services are provided, used, and managed by different actors, and the description of the actual services that can be invoked.

  The latter distinguishes between descriptive entities – that are independent views on a scenario by different involved actors – and the actual objects they act upon - representing the vocabulary of different involved actors. This separation allows the adoption of different – and in some cases already existing – vocabularies for multiple viewpoints.

- **An Ontological Framework for Service-Oriented e-Government Applications.** We define a framework composed by three ontologies (Core Life Event Ontology, Domain Ontology, and Service Ontology) that maps a distributed e-Government service-supply scenario where multiple actors are independent nodes describing their own knowledge, and provides mechanisms of knowledge sharing and mismatch resolution. It is a general purpose tool that domain experts can adapt and extend using different levels of granularity on the bases of scenario characteristics. The result is a reusable, extensible, and flexible model.

- **A Core Life Event Ontology (CLEO).** It is the heart of our framework allowing the description of the configuration and guidelines knowledge levels, and mapping the e-Government Conceptual Models. The objective of this ontology is to enable e-Government actors to represent the knowledge they want to describe with the scenario in their own “language” and based on concepts which are familiar to them, excluding the use of technical concepts. It contains a number of generic concepts and method-independent definitions, and enables the adoption of several SWS approaches. Thus, it is extremely useful for reuse purposes.

- **Contribution to PA Autonomy and Cooperative Development.** Each organization should keep its autonomy describing its own domain. Actually, distinct organizations could use or describe the same concepts differently. This implies addressing the issues of mediation between heterogeneous sources, but allows the cooperative development of an e-Government application. The structures of CLEO and the Domain Ontology enable such a approach.
• **Contribution to describe interactions among agents.** The transaction description is a too often neglected component of business process. In our work, we extend an existing framework for designing multi-agent scenario based on economical transactions by adapting it to describe generic interactions between e-Government actors. Moreover, we provide a mechanism that allows to describe complex interaction protocols that cannot be represented by current SWS standards. As a result, non-software agents (e.g. citizen) may interact with the multi-agent environment provide by SWS technology in order to achieve their tasks. These two modules are tightly connected and part of CLEO.

• **Contribution to knowledge elicitation in the e-Government domain.** CLEO introduces a knowledge elicitation methodology that first helps domain experts to create a full description of a specific e-Government context using models close to their experience and specific languages of the different involved domains, and then drives the application developers to implement SWS descriptions inferring requirements from the context description.

### I.6 Thesis Organization

In this section, we describe how all chapters in our thesis are organized. Figure 2 depicts the flow of the chapters.

![Figure 2: Structure of the Thesis](image)

Chapter 1 provides an overview of the main concepts connected to the ontologies, developing methodology and languages for representing ontologies. Chapter 2 describes the existing approaches for knowledge modelling of services. This literature review is directly relevant to our thesis, whereby first we acquaint ourself with the most central concepts for constructing a semantic-based framework for service-oriented systems: ontologies and SWS’s. In Chapter 3, we review the status of existing knowledge-based approaches of service-oriented e-Government applications. We highlight the major shortcomings of existing approaches and formulate the specific objectives of our research.

In Chapter 4, we describe the core of our research proposal. Starting from the requirements that drive our approach, we describe the conceptual models that are the base of our work, specify the particular modelling approach and tools we adopt for mapping the conceptual models onto ontologies, and introduce the resulted ontological framework.

In Chapter 5, we carry on the description of our work providing the formalization of the ontologies that compose the framework introduced in the previous chapter. In particular, we focus on the Core Life Event Ontology (CLEO) – that is the core of the framework – by detailing its independent modules.

The following chapters provide two distinct examples of application of our ontological framework. In Chapter 6, we describe the elicitation methodology based on the ontological framework, and we use it in order to model a real-life case study. The latter also provides an applicability evaluation study of our ontological framework.
In Chapter 7, we outline the high level architecture of a semantically-enhanced middleware based on our ontological framework. We adopt a use case narrative description for showing how a generic KMS extending the middleware can use and transfer the retained knowledge. We refer to a specific life event for providing an evaluation of the reasoning capabilities of our ontological framework. The proposed middleware system is not developed yet, but this chapter introduces the first stage of our next future work.

I.7 Relation to previous published work of the author

The content of this thesis relates to the following previous works by the author:


- A. Gugliotta, L. Cabral, J. Domingue “Knowledge Modelling for Integrating E-Government Applications and Semantic Web Services”. Accepted as long paper at the AAAI Spring Symposium Series, Stanford University, California, USA, 27-29 March 2006.

Moreover, since January 2005, I develop my research activity in conjunction with the Knowledge Media Institute (KMI), the Open University, Milton Keynes (UK), directed by prof. Enrico Motta. In particular, I am a member of the Semantic Web Services group, led by prof. John Domingue, and I am involved in the Work Package 9 of the DIP project (Data, Information and Process Integration with Semantic Web services FP6 - 507483).
First Part
As introduced in Section I.1, an ontology is an explicit specification of a conceptualization that may belong to a semantic layer for communicating and representing knowledge.

This chapter contains a general introduction to the ontologies and the most important related concepts that have been considered in our work. In particular, we show the main components, the design methodologies, and the main steps of an ontology life-cycle. We provide a brief overview of the main developing methodology and formalisms for representing ontologies.

1.1 What is an ontology

1.1.1 Informal definitions

The ontologies are a powerful conceptual tool to model the knowledge. They allow the elicitation, formalization and interchange of knowledge.

The term “ontology” comes from the philosophical field where it is used to show a systematic description of the existence. In the context of Artificial Intelligence (AI) the term “ontology” is used with different meanings. A terminological clarification has been made in [54]. The initial definition has been proposed by Gruber in [49]:

**Definition 1.1: Ontology** The description of some concepts and relations between them, within a specific domain.

and then slightly modified in [11]:

**Definition 1.2: Ontology** A formal and explicit specification of a shared conceptualization.

As argued in [41]

- **explicit**: meaning the types of the used concepts and their constraints are explicitly defined;
- **formal**: the ontology should be machine-readable;
- **shared**: the represented knowledge of an ontology should capture the consensual knowledge, that is not a private type of knowledge – of a single individual – but accepted by a group;
- **conceptualization**: referring to an abstract model of a domain identifying the main elements of that domain.

From [92], an ontology defines a common vocabulary among different systems that is composed by a set of terms and constraints on that terms. The constraints specify the semantics of the terms restricting the number of interpretations.

An ontology can be defined starting from the terms representing the application domain we want to describe. When the knowledge of a domain is represented with a declarative formalism,
the set of objects is named universe of discourse. The set of these objects and the linking relations are reflected in the vocabulary that the knowledge-based application uses. In that ontology, the definitions associate entity names of the universe of discourse with a text describing the meaning of that names, and some formal axioms constraining the interpretation and well formed use of the terms.

In [54], another definition of ontology has been introduced. Independently from the particular used vocabulary and the real occurrence of a specific situation, [54] firstly introduces the conceptualization as a formal structure of the reality (or part of it), how it is interpreted and organized by an agent. In this way, different situations regarding the same objects and described by different vocabularies can be represented with the same conceptualization. Formally, a conceptualization for a domain D is the set $C = \langle D, W, R \rangle$, where W is the set of possible worlds on D, and R represents a set of conceptual relations on the couple $\langle D, W \rangle$. As a consequence of that definition, the conceptualization holds the description of the meaning for all the possible relations between the objects of the domain. Thus, the definition of the meanings can be represented by relations; the idea is to define a set of relations including all the scenarios in every possible world. Finally, [54] supplies the following definition of ontology:

**Definition 1.3: Ontology** An ontology is an engineering artifact composed by a specific vocabulary describing a particular reality, and a set of explicit assumptions regarding the meaning of the available vocabulary terms.

Practically, in [54] an ontology is a logic theory that explains the imputed meanings of the terms of a formal vocabulary.

Informally, most of the researchers consider that the role of an ontology is to capture the knowledge of a domain and supply agreements on its comprehension.

The definition of ontology is tightly connected with the concept of taxonomy:

**Definition 1.4: Taxonomy** A taxonomy is an acyclic and transitive relation.

Thus, an ontology contains the modeling primitives – such as concepts, relations, and axioms –, and is usually organized in a taxonomy defining a partial order on the concepts.

### 1.1.2 A formal definition

An ontology is the set of symbols $O := (L, F, G, C, H, S)$ and consists of [65]:

- A lexicon $L$: containing the set of symbols (lexical entries) for concepts $L^C$, and a set of symbols for template slots $L^S$. The union of these two sets is the lexicon $L := L^C \cup L^S$.

- Two reference functions $F, G$, where $F : 2^{L^C} \rightarrow 2^C$ and $G : 2^{L^S} \rightarrow 2^S$. $F$ and $G$ link the sets of the lexical entries $L_i \subset L$ to the set of concepts and template slots. In general, a lexical entry can refer to several concepts or template slots and a concepts or template slot can refer to several lexical entries. The inverse functions are: $F^{-1}$ and $G^{-1}$.

- A set of concepts $C$ (or classes): for each $c \in C$ at least exists a declaration in the ontology, that is it is part of a taxonomy.

- A taxonomy $H$: the concepts are taxonomically linked by acyclic and transitive relations $H, (H \subset C \times C)$. $H(C_1, C_2)$ shows that $C_1$ is subconcept of $C_2$.

- A set of template slots $S$: it is a relation; it is referred by its lexical entries, and specified by a couple $(C, R)$ with $C, R \subset C$. A template slot $S$ adds a range of restrictions – following an object-oriented style – to a concept description $C$. An instance $i_1$ of $C$ can be linked to another instance $i_2$ through $S$, only if $i_2 \in R$. The functions $d$ and $r$ applied to $S$ respectively produce the domain $C$ and the range of concepts $R$. 
1.2 Main components of an ontology

Ontologies formalize the knowledge of a domain by means of the following components [82]:

- **Concepts.** Also named classes, are used with a wide meaning. Actually, a concept may represent an object, a notion, or an idea. Can be abstract or tangible, simple or complex, real or fictitious. Briefly, a concept can be any entity which we can state something on; therefore it can also be the description of a task, function, action, strategy, reasoning process, etc. Formally, a concept is described by a term (generally a symbol), an extension, and an intension. The extension of a concept is the set of objects that the concept can be applied to: for instance, the extension of “car” includes: “the green car parked at the end of the street” or “the ford fiesta”. The intension of a concept is the set of properties, features, and attributes specifying the semantics of the concept, that is the set of features shared by these objects. For instance, the intension of the concept “car” includes the features of “a street vehicle with engine, generically with four wheels”.

Specific elements can be considered in the description of a concept: **Metaclasses** that are classes as instances of other ones; **Slots/Attributes** which belong to a specific concept; for instance, the attribute *age* may belong to concept *Person*; **Facets** that allow the specification of an attribute; e.g. default value, type, cardinality, operational definition.

- **Taxonomies.** They are widely used to organize ontological knowledge in the domain using generalization/specialization relationship through which simple/multiple inheritance could be applied. Their semantic may be based on the following definitions: subclass of, partitions, disjoint composition, etc.

- **Procedures.** They represent operational definitions to infer values of arguments, or execute formulas and rules.

- **Relations.** Represent a kind of interaction among domain concepts. They are formally defined as any subset of the product of $n$ sets, that is $R : C1 \times C2 \times ... \times Cn$. A relation is described by a term, an extension, and an intension. The extension of a relation is the set of possible tuples of the relation used objects. For instance, the extension of the relation “parent” includes: “Giovanni is my father” and “Luca and Maria are the parents of Cristina”. The intension specifies the types of the linked concepts. For instance, the intension of the relation “parent” includes: “the growing of the child and all the involved activities and duties”. Examples of binary relations are: “is-a” and “links to”

- **Functions.** They are a special type of relation, where the value of the last argument is unique for the list of $n - 1$ previous arguments. Formally, they are defined such as: $F : C1 \times C2 \times ... \times Cn-1 \rightarrow Cn$. An example of binary function is “mother of”, while an example of ternary function is “the price of a used car” that compute the price of a second-hand car base on the car model, the making year, etc.

- **Axioms.** Assertions onto the model, that are always true and specify the semantics of the concepts. They generally describe how the vocabulary (concepts and relations) can be used to reason on the domain. They are included in an ontology as information constraints, correctness checking, or knowledge inference. Particularly, they may express the kind of relation among concepts, the signature and the cardinality of a relation, algebraic properties of a relation (symmetry, transitivity), and other conceptual properties, such as exclusivity, generality, or identity.
• **Instances, Individuals, Facts, and Claims.** The instances represent the elements of the domain. The individuals are any element of the domain but not a class. The facts represent a relation between two elements. The claims are assertions through an instance.

• **Production Rules:** follow a if-then structure and are used to express a set of actions or heuristics.

• **Inference Mechanisms:** describe how the previously defined structures (static) can lead to a reasoning process (Figure 1.1). The following features can be performed by an inference engine: automatic classification, detecting exceptions (e.g. attribute Attribute1 is defined for concept C1 and concept C2, being C1 subclass of C2 and the inference engine analyses whether the definition of Attribute1 in concept C1 overrides the definition of Attribute1 in concept C2 or not), inheritance (monotonic, non monotonic, simple and/or multiple), executing procedures, performing constraint checking by using axioms, reasoning with rules (forward and backward chaining).

---

![Figure 1.1: Relationships between knowledge representation and reasoning components of an ontology](image)

### 1.3 Criteria for designing ontologies

Only after several design decisions the domain of interest representation can be realized by means of an ontology. To drive and value the design process is needful identifying the set of objective design criteria that help the knowledge sharing and the interoperability among shared-conceptualization-based programs [50]. These criteria should be based on the goals of what we are developing rather than a-priori notions. In the following, we report some of the most important design criteria:

• **Clarity.** An ontology should communicate the intensional meaning of the defined terms. The definitions should be objective. While the motivation to introduce a concept may depend on social situations or computational requests, the definition should be independent on the social and computational context. Beside this, all the definitions should be documented in natural language.
• **Coherence.** An ontology should be coherent, that is it should allow to infer only what is consistent with its definitions. The axioms should be logical consistent and the coherence should be also applied to the informally defined concepts, that are described within the natural language documentation and examples.

• **Extendability.** An ontology should be designed foreseeing the future usages of the shared vocabulary. It should offer a conceptual foundation for a set of anticipated tasks, and the representation should allow to extend and specialize the ontology. In other words, we should be able to easily define new terms starting from the existing vocabulary without revising the existing definitions.

• **Minimal code dependency.** The conceptualization should be specified at knowledge level without dependencies from particular code at level of symbols. A dependency from the code is obtained when the choices of representation are made on the base of notation and implementation convenience. The code dependency should be minimized.

• **Minimal ontological commitment.** An ontology should include the minimal number of definitions describing the domain, but allowing the ontology specialization and instantiation on the base of particular needs. The ontological commitment is based on the consistent use of the vocabulary; thereby it can be minimized specifying the weaker theory and only defining the necessary terms for a theory-consistent knowledge communication.

### 1.4 The life-cycle for developing ontologies

The process of ontology development refers to which activities are performed when building ontologies. In the literature, several works concern individuating the fundamental steps and guidelines to design and develop an ontology [47]. In spite of the collective experience in developing and using ontologies, a standard building methodology does not exist. In the following, we summarize the life-cycle adopted in our work. We distinguish three categories of activities: management, development, and support (Figure 1.2).

The **management** phase includes activities such as scheduling, control and quality assurance. The scheduling activity identifies the tasks to be performed, their arrangements, and the time and resources needed to their completion. The control activity guarantees that scheduled tasks are completed in the manner intended to be performed. The quality is ensured by the last activity.

In the **development** phase, the preliminary step is the specification, i.e. detecting the aim of the ontology. Before starting, all the motivations for creating the ontology and its intended use should be clear. Beside this, the set of ontology users should be identified; afterwards, the actual building process can start. The latter is usually decomposed into the following steps: conceptualization, formalization, implementation, evaluation, and maintenance.

• **Conceptualization.** Describing with a conceptual model the ontology, matching the specifications defined in the previous step. In this phase, the domain key concepts and relations are identified and described, providing precise and unambiguous definitions, and detecting the ways to refer to them.

• **Formalization.** Transforming the conceptual model into a formal or semi-computable model.

• **Implementation.** Implementing a computable model in an ontology language.

• **Maintenance.** Updating and correcting the ontology if needed.

Finally, the **support** activities include:

• **Knowledge Acquisition.** Referring to relevant existing documents and domain experts to acquire knowledge about the domain.
1. Ontologies

Figure 1.2: General life-cycle for building an ontology

- **Documentation.** Reporting all the executed activities, including a detailed description and the motivation.
- **Evaluation.** Investigating the usability of the ontology judging its content from a technical point of view.
- **Integration.** Reusing other ontologies already available.

Generally, the ontology life-cycles are evolving prototyping processes, and refinement steps are allowed.

1.5 Methodologies for building ontologies

Basically, a series of methods and methodologies for building ontologies have been reported in literature [47] [78]. In the following, we introduce four of the most known and used methodologies:

- **TOVE:** also named Gruninger and Fox’s methodology, it has been used to build the TOVE ontologies [51][52], which are the pillars of the Enterprise Design Workbench, a design environment that permits the user to explore a variety of enterprise designs. It is well-founded for building and evaluating ontologies, even though some management and support activities are missing. In the following we report the processes identified:
  - Identify motivating scenario. Such scenarios describe a set of the ontology’s requirements that the ontology should satisfy after being formally implemented.
  - Elaborate informal competency questions. Given the set of informal scenarios, a set of informal competency questions are identified.
  - Specify the terminology using first order logic. From the informal competency question, the developer extracts the terminology that will be formally represented by means of concepts, attributes and relations in a first-order logic language.
1.5. Methodologies for building ontologies

- Write competency questions in a formal way using formal terminology. Informal competency questions are written as an entailment of consistency problems.
- Specify axioms using first order logic. Using axioms to specify the definitions of terms in the ontology and constraints in their interpretation.
- Specify completeness theorems. Defining the conditions under which the solutions to the questions are complete.

- **ENTERPRICE**: also named Uschold and King’s method [93][92]. The following processes must be performed:
  - Identify the purpose and scope. The goal here is to clarify why the ontology is being built, what its intended uses are, and what the relevant terms on the domain will be.
  - Build the ontology. It is broken into three activities: (i) knowledge capture adopting three possible strategies: bottom-up – first most specific concepts and then generalizing them →, top down – first the most abstract and then specialization →, and middle-out – identifying first the core of basic terms, and then specifying and generalizing them as required; (ii) coding; (iii) integrating existing ontologies to use ontologies that already exists.
  - Evaluate the ontology.
  - Document the ontology.

- **METHODOLOGY**: it enables the construction of ontologies at the knowledge level and includes: the identification of the ontology development process, a life cycle based on evolving prototypes, and techniques to carry out each activity in the management, development, and support phases 1.4. Peculiar to this approach is the explicit definition of the ontology life cycle; it identifies the set of stages through which the ontology moves during its life time, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.).

- **On-To-Knowledge**: this approach has been developed to build ontologies to be used by the knowledge management application [88]. Therefore, it proposes to build the ontology taking into account how the ontology will be used in further applications. Consequently, ontologies developed with this methodology are highly dependent of the application. The methodology also includes the identification of goals to be achieved by knowledge management tools, and is based on an analysis of usage scenario. In the following we report the processes proposed by this methodology:
  - Feasibility Study. It is applied to the complete application and, therefore, should be carried out before developing the ontologies.
  - Kickoff. The result of this process is the ontology requirements specification document that describes: the domain and the goal of the ontology, the design guidelines, available knowledge sources, and potential users and use cases as well as applications supported by the ontology.
  - Refinement. It produce a mature and application oriented “target ontology” according to the specifications given in the previous phase.
  - Evaluation. It serves as a proof of the usefulness of the developed ontology and their associated software environment.
  - Maintenance. It is important to clarify who is responsible for the maintenance and how it should be carried out.

In [47], the authors present a comparison among the above and other methodologies. The conclusion reached from that analysis are:

- None of the approaches covers all the processes involved in the ontology building.
Most of the approaches are focused on development activities.

Most of the approaches present some drawback in their use.

Most of the approaches do not have a specific tool that gives them technological supports.

In particular, about the presented methodologies:

- METHODOLOGY is the approach that provides the most accurate description of each activity.
- On-To-Knowledge describes more activities than the other approaches.
- The strength of TOVE methodologies is its high degree of formally, but it is not completely specified.
- The ENTERPRICE approach has the same omissions of TOVE an it is less detailed.

Methods and Methodologies have not been created only for building ontologies from scratch. Re-engineering methods allow to re-using existing ontologies that, for instance, are not implemented with the same, language, representation paradigm, or conventions. METHODOLOGY includes methods for solving some of these problems. Ontology learning methods [6] have been thought up to decrease the effort made during the knowledge acquisition process 1.4. Ontology alignment methods [72] establish different kinds of mappings (or links) between existing ontologies that models, in different ways, the same kind of knowledge or domain. This option preserves the original ontologies. Contrariwise, the ontology merging methods [72] propose to generate a unique ontology from the original ontologies. In general, it is more suitable to establish ontological mappings between existing ontologies on the same topics than to pretend to build a unified knowledge model for such a topic. Finally, it can be introduced methods for collaborative and distributed construction of ontologies [38] [37] that allow to agree new pieces of knowledge with the rest of knowledge architecture, which has been previously agreed upon.

1.6 Classification

An ontology may assume several formats, even if it has to necessarily include a vocabulary of terms and some definition of their meaning [71] [92]. The formality degree wherewith a vocabulary is built and the meaning of its terms is ascribed to may change a lot:

- **highly informal**: written in natural language;
- **semi-informal**: expressed by structured and restricted natural language formats to increase the clarity and reduce the ambiguity;
- **semi-formal**: defined by a formal language;
- **highly formal**: defined by a language with a formal semantic and property checking mechanisms such as completeness.

A further classification based on the kind of conceptualization [53][94] groups ontologies in the following categories:

- **Meta-ontologies**: capturing the representation primitives used to formalize the knowledge within a specific family or knowledge representation system.
- **General/common ontologies or upper ontologies**: classifying several existing entity categories of the world and representing very general notions that are independent of a particular problem or domain. Example of these ontologies are: SUO [2], SUMO [3], Cyc [55], and DOLCE [75].
1.7. Formalisms for representing ontologies

Ontologies play an important role in several research fields, such as knowledge management, information and database integration, e-commerce, cooperative systems, and data-mining. Nevertheless the variety uses and involvements of ontologies in these fields, there is a common interest to define and develop ontologies with definite quality standards [89]. A lot of formalisms have been defined guarantying the possibility to choose the best language matching own objectives and aims. The more representative ontology languages can be classified in the following base formalisms:

- Logic (First Order Logic)
- Semantic Nets
- Description Logic
- Frames
- Conceptual Graphs

1.7.1 Logic

The logic is the science supplying to the human the indispensable tools to safety control the strictness of reasonings. The Artificial Intelligence uses the logic to design logic networks and query relational databases. Beside this, the logic may be used as programming language (Logic programming, Prolog).

The classical logic is divided into: prepositional logic and predicate logic. The difference between the two classes resides in the level of expressiveness: the predicate logic allows to express variables and quantifiers, while the prepositional logic can not do that. A First Order Logic language is defined by:

- a syntax: defining the structural characteristics of the language, without ascribing any meaning;
- a semantic: interpreting the syntactic right phrases of the language.

The most representative ontology languages belonging to the First Order Logic are Cyc [55] and Kif [46] that use the predicates as modeling primitives.

1.7.2 Semantic Nets

The semantic nets are a structured representation of the knowledge. Their formalism describes the reference domain objects by means of oriented graphs, which nodes represent the concepts and arcs the relations between concepts. Figure 1.3 shows a simple and intuitive example of knowledge representation with semantic nets.

A fundamental property of the semantic nets is the possibility to manage the inheritance: the properties of a concept are inherited from all the sub-concepts. They also allow the multiple inheritance.
1.7.3 The description logic

The description logic represents the knowledge in terms of concepts and constraints; i.e. it allows to define the concepts by means of descriptions specifying the properties that objects should satisfy for belonging to a concept. It derives from studies and defines a formal semantic on the semantic nets.

The description logic realizes some constraints to the first order logic to obtain better computational properties. However it comes out less expressive than the first order logic because it does not have variables, quantifiers, connectives, function symbols (with arity ≥ 1), predicate symbols with arity ≥ 3, and equality predicate; but it uses other operators that can be seen as fixed combination of the above classical one's.

A description logic knowledge-base system is generally composed by three components:

- A terminological module (MT): a language defining concepts and roles (terms);
- An assertional module (MA): defining ground assertions for the terms;
- A definitional module (MD): specifying inclusion relations between terms;

The MT module is the specific part of the description logic. An example of these languages is LOOM [64], that it is a knowledge representation system developed by the Information Sciences Institute, University of Southern California – under the sponsorship of DARPA –, with the aim of developing advanced tools to knowledge representation, and artificial intelligence reasoning. Actually, LOOM is both a language and an environment for building and managing knowledge-based intelligent applications and expert systems.

1.7.4 Frames

The frame is an information container representing stereotype situations; it is a data structures describing domain objects (classes). Every frame has a unique name and defines the feature of the objects through a set of slots (attributes). Such languages are different at different levels, but they are all based on a notation that specifies the concepts and the relations. The main common aspect of these languages are the following:

- **Object-Oriented**: all information regarding a concept are stored with the concept itself.
- **Generalization/Specialization**: supplying a natural way to group concepts in hierarchies (where the top concepts are the most general concepts), and to share attributes and sub-concepts.
- **Reasoning**: every language offers a different reasoning approach, that is the capability to formally express the existence of a piece of knowledge (previously not expressed) from the existence of other pieces of knowledge.

![Semantic Net Diagram](image-url)

Figure 1.3: Example of Semantic Net: the collocation of the blue whale in the animal kingdom.
1.8. Application fields of ontologies

- **Classification**: given an abstract description of a concept, a lot of these languages allow to express if a concept is subsumed from that abstract description.

The first two items of the above description allow to make human-understandable the language, while the last two allow the system to behavior as it knows what is represented.

Some examples of these languages are: Ontololingua [40], FLogic [61], and OCML [70].

### 1.7.5 Conceptual Graphs

Conceptual graphs provide a powerful multi-paradigm knowledge representation and inference environment having been originally devised by Sowa [87] from philosophical, psychological, linguistic, and artificial intelligence foundations in a principled way. Conceptual graphs also exhibit the entity-relational and object-oriented features of contemporary enterprise and web applications. They take advantage of a graphical formalism to represent the knowledge. This formalism can be brought back to the following notations: display form, linear form, and predicate calculus. In the following, we show how the concept “the cat is on the mat” is represented with the above three notations:

- **Display form**: the concepts are represented by rectangles and relations by circles; the direction of the arrows determines the direction of the reading 1.4.

  ![Figure 1.4: Display Form](image)

- **Linear form**: the concepts are represented by square brackets and relation by circle brackets.

  $\{Cat\} \rightarrow (on) \rightarrow \{Mat\}$.

- **Predicate Calculus**:

  $\exists x : Cat \exists y : Mat \text{on}(x, y)$.

All concepts have referents, which refers to a particular instance, or individual, of that concept. For example consider the concept: $\{Cat : Clyde\}$. This reads as “The cat known as Clyde”, and also happens to show that a conceptual graph can consist of only one concept. A concept that appears without an individual referent has a generic referent, and should be denoted as $\langle \text{Type} - \text{Label} > : * \rangle$.

1.8 Application fields of ontologies

Ontologies can be used in several fields, and one’s of the most important and topical is the Web systems. Ontologies are now ubiquitous in many enterprise-wide information-systems: they are used in e-commerce, knowledge management and in various application fields such as bioinformatics and medicine. Moreover, they constitute the backbone for the Semantic Web, which is discussed in the next chapters. In the following, we report some of the practical applications of ontologies, starting from the simplest (taxonomies) to the more complex (structured ontologies with inference rules) [68].

Simple ontologies:

- **Controlled Vocabulary**: allowing the use of the same terms about particular concepts at different levels: users, developers, and systems; this is a first level of interoperability.
• **Site organization and navigation support**: a simple taxonomy within an ontology can be used to develop menu, sub-menu and in general the structure of a Web site. In this way, starting from the more general to the more specific level a user can step-by-step increase his/her idea about the following pages.

• **“Umbrella” structures from which to extend content**: starting from existing and general ontologies we can derive specific concepts of a domain.

• **Browsing support**: the contents of a Web site can be described with the concepts of an ontology; for instance introducing meta-data defined in an ontology. Once a page (or service) is meta-tagged with a term chosen from a controlled vocabulary, then search engines may exploit the tagging and provide enhanced search capabilities.

• **Search Support**: a user can adopt terms of a taxonomy to query a system.

• **Sense disambiguation support**: if the same term appears in multiple places in a taxonomy, an application may move to a more general level in the taxonomy in order to find the sense of the word.

Structured ontologies:

• **Consistency checking**: if an ontology contains properties and constraints about possible values, an application can check the consistency of some values and eventually notify the error.

• **Completion**: an application can require simple information to the user (e.g. the user’s gender in a medical application); the ontology allows to add more information on the base of the user information (e.g. if the gender is “male” the ontology reject the possibility that the user is pregnant).

• **Interoperability support**: in the simple case of considering controlled vocabularies, there is enhanced interoperability support since different users/applications are using the same set of terms. In simple taxonomies, we can recognize when one application is using a term that is more general or more specific than another term and greater facilitate interoperability. In more expressive ontologies, we can have a complete operational definition for how one term relates to another term and thus, we can use equality axioms or mappings to express one term precisely in terms of another and thereby support more “intelligent” interoperability.

• **Support validation and verification testing of data (and schemes)**: if an ontology contains class descriptions, these definitions can be used as queries to databases to discover what kind of coverage currently exists in datasets.

• **Encode entire test suites**: an ontology can contain a number of definitions of terms, some instance definitions, and then include a term definition that is considered to be a query (e.g. find all terms that meet the following conditions). Markup information could be encoded with this query to include what the answer should be, thus providing enough information to encode regression testing data.

• **Configuration support**: an ontology can describe the components of a particular system; defining relation among the properties of the components, a change in a property can generate a modification in a property of related components.

• **Knowledge Capture support**: an ontology can represent a knowledge capturing process: the steps to follow, and the entity to describe; e.g. an organization can use ontologies to better capture organizational knowledge.

• **Support structured, comparative, and customized search**: the properties defined for a particular term can be used to compare different instances and provide a detailed set of specifications about the items a user is looking to find. More sophisticated ontologies can be generated that
mark which properties are most useful to present in comparative analysis so that users may have concise descriptions of the products instead of comparisons in complete detail. Thus, ontologies with markup information may also be used to prune comparative searches.

- **Exploit generalization/specialization information**: If a search application finds that a users query generates too many answers, one may dissect the query to see if any terms in it appear in an ontology, and if so, then the search application may suggest specializing that term.

## 1.9 Languages for Building Ontologies

Ontolingua [40] is perhaps the most representative of ontology languages. Other *traditional languages* have also been used for specifying ontologies: CycL [56], LOOM [64], OCML [70], FLogic [61], etc.

In the recent years, new languages for the web have been created: XML [12], RDF [60] and RDF Schema [13]. Other languages for the specification of ontologies, based on the previous ones, have also emerged: SHOE [63], XOL [58], OIL [39], DAML+OIL [8], and OWL [86].

These languages can be grouped in the following three categories (Figure 1.5):

- The class of the *traditional languages* that come from the Artificial Intelligence and Knowledge Engineering [89]. These languages can be divided into: predicate Languages, frame based, and description logic (see Section 1.7).
- The class of web standards that are used to exchange knowledge through the Web.
- The class of web-oriented languages that are based on the above web standards.

![Figure 1.5: Ontology Language Classification](image)

In the sequel, we take into consideration OCML and OWL languages; the former is the language we actually adopt for building our conceptual model, the latter is the reference standard for the web-oriented languages. The aim is to compare their expressiveness and inference mechanisms.

We do not give a complete description of the two languages. For both, we firstly present a general description of which context the language has been developed in, and its formalism and main objectives. Then we show examples representing the main entity and relation of a specific domain. To ease the comparison we adopt the same domain for all the proposed examples: Public Administration. Finally, we provide a complete comparison by means of a table. The latter is filled using “yes” to indicate that it is a supported feature in the language, “no” for non supported features, “y/n” for non supported features, but could manage to support it by doing something, “?” when no information is available and N.D. for features which are not restricted, but could be implemented in order to support them. We distinguish between components describing domain knowledge and inference knowledge. In particular, we refer to the introduced descriptions of ontology components (Section 1.2).
1.9.1 OCML

OCML [70] stands for Operational Conceptual Modeling Language. It is a frame-based language that provides mechanisms for expressing items such as relations, functions, rules (with backward and forward chaining), classes and instances. In order to make the execution of the language more efficient, it also adds some extra logical mechanisms for efficient reasoning, such as procedural attachments. A general tell&ask interface is also implemented, as a mechanism to assert facts and/or examine the contents of an OCML model.

Several pragmatic considerations were taken into account in the development of OCML. One of them is the compatibility with standards, such as Ontolingua, so that OCML can be considered as a kind of “operational Ontolingua”, providing theorem proving and function evaluation facilities for its constructs.

Example 1.1: Defining the concept of ChangeOfResidenceService

- The change of residence can be represented as a class of objects:

  \[
  \text{(def-class ChangeOfResidenceService () "Changing residence service")}
  \]

  where \text{def-class} states that we are defining a new class; the \text{string} is optional and supplies a text description of the class.

- To enrich the definition stating that ChangeOfResidenceService is subclass of the concept \text{HomeRelatedLifeEvent} we should do:
  - Defining the HomeRelatedLifeEvent concept:
    \[
    \text{(def-class HomeRelatedLifeEvent () "Life Events connected to the house management")}
    \]
  - Defining ChangeOfResidenceService as subclass of HomeRelatedLifeEvent:
    \[
    \text{(def-class ChangeOfResidenceService (HomeRelatedLifeEvent))}
    \]

- To define an instance of the ChangeOfResidenceService class, we should do:

  \[
  \text{(def-instance ChangeOfResidenceService-instance ChangeOfResidenceService)}
  \]

- To define a relation for stating which are the processes of the service, we should do:

  \[
  \text{(def-relation subprocess (?ChangeOfResidenceService ?ChangeAddressService) "the instance ?ChangeOfResidenceService has subprocess the the instance ?ChangeAddressService") :constraint (and (ChangeOfResidenceService ?ChangeOfResidenceService) (ChangeOfResidenceService ?ChangeAddressService)))}
  \]

  where \text{:constraint} states the classes of the two involved instances; such a constraint may be used to specify the relation.

1.9.2 OWL

The Web Ontology Language OWL [86] is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF [60] and is derived from the DAML+OIL [8] Web Ontology Language. The goals of the OWL language are:

- formalization of a domain by defining classes and properties of those classes;
- definition of individuals and assertion of properties about them;
reasoning about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

OWL provides three increasingly expressive sub-languages designed for use by specific communities of implementers and users:

- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1;

- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class);

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Since there is no tool that supports OWL Full, in the rest of this section we discuss modelling primitives provided by OWL Lite / OWL DL.

A class defines a group of individuals that belong together because they share some properties. There are two built-in classes: *Thing* is the class of all individuals and is a superclass of all OWL classes and *Nothing* is the class that has no instances and is a subclass of all OWL classes. Class hierarchies may be created by making one or more statements that a class is a subclass of another class.

Moreover, properties can be used to state relationships between individuals or from individuals to data values. Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties. A domain of a property limits the individuals to which the property can be applied. The range of a property limits the individuals that the property may have as its value.

Finally, individuals are instances of classes, and properties may be used to relate one individual to another.

Besides these primitives related to the RDF Schema, OWL Lite includes primitives related to equality or inequality. For example, two classes (properties) may be stated to be equivalent. Equivalent classes have the same instances. Equivalent properties relate one individual to the same set of other individuals. Equality can be used to create synonymous classes (properties). Two individuals may be stated to be the same. These constructs may be used to create a number of different names that refer to the same individual. Moreover, an individual may be stated to be different from other individuals and a number of individuals may be stated to be mutually distinct.

There are special identifiers in OWL Lite that are used to provide information concerning properties and their values. One property may be stated to be the inverse of another property. Properties may be stated to be transitive and/or symmetric.

Moreover, properties may be stated to have a unique value. This feature is shorthand for stating that the property’s minimum cardinality is zero and its maximum cardinality is 1. Properties may be stated to be inverse functional, which has also been referred to as an unambiguous property.

OWL Lite allows restrictions to be placed on how properties can be used by instances of a class. The restrictions *allValuesFrom* and *someValuesFrom* are stated on a property with respect to a class. The first one means that this property on this particular class has a local range restriction associated with it. The second one means that a particular class may have a restriction on a property that at least one value for that property is of a certain type.
OWL Lite includes a limited form of cardinality restrictions (minCardinality and maxCardinality), since it allows statements concerning cardinalities of value 0 or 1.

Further, OWL Lite contains an intersection constructor but limits its usage. OWL uses the RDF mechanisms for data values. It supports notions of ontology inclusion and relationships and attaching information to ontologies.

Example 1.2: Defining the Change of Residence Service

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF
   xmlns="http://a.com/ontology#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:ID="ChangeOfResidenceService">
    <rdfs:subClassOf rdf:resource="#HomeRelatedLifeEvent"/>
  </owl:Class>

  <!- Defining the properties Subprocess and DossierNumber -->
  <owl:ObjectProperty rdf:ID="subprocess">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      defining a
    </rdfs:comment>
    <rdfs:domain rdf:resource="#ChangeOfResidenceService"/>
    <rdfs:range rdf:resource="#ChangeAddressService"/>
  </owl:ObjectProperty>

  <owl:DatatypeProperty rdf:ID="DossierNumber">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#ChangeOfResidenceService"/>
  </owl:DatatypeProperty>

  <!- Defining an instance of ChangeOfResidenceService -->
  <ChangeOfResidenceService rdf:ID="EventiVitaEx_Instance_2">
    <NumPratica rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1289bis</NumPratica>
  </ChangeOfResidenceService>
</rdf:RDF>
```
1.10 Comparison of languages

The trade-off between the degree of expressiveness and the inference engine of a language (the more expressive, the less inference capabilities) makes it difficult to establish a scoring of languages. Moreover, we claim that different needs in KR exist nowadays for applications, and some languages are more suitable than others for the specific needs of a given application.

We refer to the comparison proposed in [47] that helps to understand better the similarities and differences between languages and the capabilities of them. It does not establish a score of languages (in the sense of “language X is better than language Y”), as different, as sometimes incompatible, feature may be needed for different ontology-based application.

In particular, we report here the comparison between OWL and OCML. These languages are the two mainly candidate for formalizing our conceptualizations. The former represents the reference standard for web-oriented applications; the latter is a powerful language that easily allows to represent, reasoning, and execute operational knowledge: functions, procedure, and rules. Since in the e-Government domain the representation of operational knowledge (e.g. protocols, internal procedures, policy rules, etc.) is an important aspect, we chose OCML as reference language. Moreover, as showed in Table 1.1, the two languages have a similar expressiveness, and an OCML to OWL translator already exists\(^1\).

Cells in the table are filled using “yes” to indicate that it is a supported feature in the language, “no” for non supported features, “y/n” for non supported features, but could manage to support it by doing something, “?” when no information is available and ND for features which are not restricted, but could be implemented in order to support them. We distinguish between components describing domain knowledge and inference knowledge. In particular, we refer to the introduced descriptions of ontology components (Section 1.2).

---
\(^1\)http://plainmoor.open.ac.uk:8080/ksw/pages/owl.html
<table>
<thead>
<tr>
<th>CONCEPTS</th>
<th>OCML</th>
<th>OWL</th>
<th>OCML</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>sound</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>instance attributes</td>
<td>yes</td>
<td>complete</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>class attributes</td>
<td>yes</td>
<td>y/n</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Facets</td>
<td>automatic class</td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>type constraints</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>exceptions handling</td>
</tr>
<tr>
<td>cardinality constraints</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>procedural Knowledge</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TAXONOMY</td>
<td>monotonic</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>subclass-of</td>
<td>yes</td>
<td>non-monotonic</td>
<td>y/n</td>
<td>no</td>
</tr>
<tr>
<td>disjoint-decomposition</td>
<td>yes</td>
<td>single</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>exhaustive-decomposition</td>
<td>y/n</td>
<td>y/n</td>
<td>multiple</td>
<td>yes</td>
</tr>
<tr>
<td>partition</td>
<td>yes</td>
<td>y/n</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RELATIONS</td>
<td>executions</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>binary relations</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>n-ary relations</td>
<td>yes</td>
<td>y/n</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>relation hierarchy</td>
<td>yes</td>
<td>yes</td>
<td>Forward</td>
<td>yes</td>
</tr>
<tr>
<td>integrity constraints</td>
<td>yes</td>
<td>no</td>
<td>Backward</td>
<td>no</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>OTHER COMPONENTS</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 1.1: Definition of the main components of domain knowledge (left side) and reasoning mechanisms of OCML and OWL (right side)
1.11 Using ontology languages in applications

In [47], the authors give some advice on the use of languages for different kind-of ontology-based applications:

- In e-commerce applications, representational needs are not too complex; basically they need concepts and attributes, and n-ary relations between concepts. Reasoning needs are usually higher – hence, languages based on description logic are very useful – and an efficient query answering is also important in this environment – provided by most of the introduced languages.

- When using Problem Solving Methods (PSM) and domain ontologies together, OCML [70] is recommended, as it provides explicit support for this integration as well as reusable libraries. It is an operational modeling language (domain ontology) and solves the issue of PSM prototyping easily.

- In the context of Semantic Web and exchanging ontologies between application, languages based on XML (RDF, OIL, DAML+OIL, OWL) are easily read and managed.

- The creation of upper-level ontologies requires high expressiveness but not great needs for reasoning support. They are generally specified in description logics languages.

- Applications that need intelligent integration of heterogeneous information sources or information retrieval applications widely use languages based on description logics.
1. Ontologies
Knowledge Modelling of Services

The next Web generation promises services that are self-described and amenable to automated discovery, composition, mediation, and invocation. A technology promising to achieve this result is the Semantic Web Services (SWS’s). SWS’s are web services with a formal description i.e. semantics. A prerequisite to this, however, is the emergence and evolution of the Semantic Web, which provides the infrastructure for the semantic interoperability of Web Services. Web Services will be augmented with rich formal descriptions of their capabilities, so that can be utilized by applications or other services without human assistance or highly constrained agreements on interfaces or protocols. In this chapter, we describe the infrastructure of Semantic Web Services along three orthogonal dimensions: activities, architecture and service ontology. Then, we consider some languages for describing services. In particular, we describe WSDL [17] – i.e. the language for describing Web Services – and the two main approaches for SWSs: WSMO [32] and OWL-S [20]. Finally, we conclude introducing IRS-III [29] that is an implemented infrastructure, which allows the description, publication, and execution of SWS according to the WSMO conceptual model. We adopted this well known framework as base for the development and use of our SWS descriptions.

2.1 Web Services

Web Services are changing the way applications communicate with each other on the Web [16]. They promise to integrate business operations, reduce the time and cost of Web application development and maintenance as well as promote reuse of code. By allowing functionality to be encapsulated and defined in a reusable standardized format, Web services have enabled businesses to share (or trade) functionality with arbitrary numbers of partners, without having to pre-negotiate communication mechanisms or syntax representations.

A Web Service (WS) is a software program identified by an URI, which can be accessed via the Internet through its exposed interface. The interface description declares the operations which can be performed by the service, the types of messages being exchanged during the interaction with the service, and the physical location of ports. A binding then defines the machine and ports where messages should be sent. Although there can be many ways of implementing WS’s, we basically assume that they are deployed in Web servers such that they can be invoked by any Web application or Web agent independently of their implementations. In addition WS’s can invoke other WS’s.

The common usage scenario for Web services (Figure 2.1 [16]) can be defined by three phases: Publish, Find, and Bind; and three entities:

- The service requester, which invokes services;
- The service provider which responds to requests;
- The registry where services can be published or advertised.

A provider publishes a description of the service it provides to a registry. This description (or advertisement) includes a profile on the provider of the service (e.g. company name and address);
a profile about the service itself (e.g. name, category); and the URL of its service interface definition (i.e. WSDL description, Section 2.4.1). When a developer realizes a need for a new service, he/she finds the desired service either by constructing a query, or browsing the registry. The developer then interprets the meaning of the interface description (typically, through the use of meaningful labels or variable names, comments, or additional documentation) and binds to (i.e. includes a call to invoke) the discovered service. The application is known as the service requester. At this point, the service requester can automatically invoke the discovered service using WS communication protocols (i.e. SOAP [96]).

The key to the interoperability of WS’s is an adoption of a set of enabling standard protocols. Several XML-based standards have been proposed to support the scenario previously described (Figure 2.2 [16]).

**XML schema** [10] provides the underlying framework for both defining the Web Services Standards, and variables/objects/data types etc that are exchanged between services.

**SOAP** [96] is W3C’s recommended XML-data transport protocol, used for data exchange over web-based communications protocols (http). SOAP provides the envelope for sending messages over the Internet/Internet. SOAP commonly uses HTTP, but other protocols such as Simple Mail Transfer Protocol (SMTP) may be used. SOAP can be used to exchange complete documents or to call a remote procedure. SOAP messages can carry an XML payload defined using XML-S, thus ensuring a consistent interpretation of data items between different services.

**WSDL** [17] is the W3C recommended language for describing the service interface. Two levels of
2.2 Semantic Web

The vision of the Semantic Web is to provide a Web where all published material is understandable by software agents [90]. It can also be seen as a vast source of information, which can be modelled with the purpose of sharing and reusing knowledge. This allows for the automatic retrieval of information and the establishment of business cooperation. Examples of activities that can be automated with the Semantic Web include an information retrieval task where a requester wants to know the address of the nearest car dealer, or the establishment of a business contract where a requester wants to book and buy the tickets for a conference trip.

To perform such activities on the current Web, requesters enter keywords in a search engine or go directly to Web sites of services that they know provide the service that they need. These approaches typically adopt heuristics to generate machine understandable representations of the Web resources which in turn are used as indexes for the retrieval. However, most Web resources’ content (e.g. text, image) is written by humans in languages tailored to human and not machine understanding. In this context, current methods for index generation cannot capture the semantic of the resources, but only the syntactical interpretation of their content. As complex and refined the index generation method may be, the resulting resource representations are thus always approximations of the original semantic of the resource. As a result, it is impossible for these methods to ensure that the retrieval will always be satisfactory with respect to the requester’s needs. The Semantic Web vision aims at getting rid of the approximation. The mean is to systematically attach machine understandable representations to all published material.

The Semantic Web provides the necessary infrastructure for publishing and resolving ontological descriptions of terms and concepts. In addition, it provides the necessary techniques for reasoning about these concepts, as well as resolving and mapping between ontologies, thus enabling semantic interoperability of Web Services through the identification (and mapping) of semantically similar concepts.

One important first step into building the Semantic Web vision is to specify the machine
understandable language(s) in which the description of the published material is to be written. Figure 2.3 shows the layers of the Semantic Web [90].

![Semantic Web’s Layer cake](image)

The principle of the stack is that the higher level languages use the syntax and semantics defined by the lower level languages. As a result, the higher the language, the more expressive the language and the more complex the reasoning can be. Concretely, the layers are meant to be used as follows, starting from the bottom:

- **Definition of basic types:** (i) URI (Unique Resource Identifier), all Internet resources available on the World Wide Web are referenced by a URI. (ii) Unicode is an encoding system that specifies a machine understandable code for letters, digits, punctuation and some control characters. The code defines the alphabet that can be used to write machine understandable words and sentences.

- **Providing some basic syntax and naming mechanisms:** (i) XML [12] allows for the annotation of free text with tags which provide meta information about the text. (ii) NS (or XML namespaces) is a W3C standard naming mechanism for the tags defined with XML. (iii) XML Schema [10] is a W3C standard that allow the definition of structured concepts.

- **Allowing a richer data modeling by providing the notion of relationships:** (i) RDF [60] allows to write statements composed of three elements: subject, object and predicate where the predicate expresses the relationship between the subject and the object. Each subject, object and predicate has a unique URI. (ii) RDF Schema [13] builds on XML Schema and specifies a typing mechanism which is similar to the class mechanism of the object-oriented paradigm.

- **Providing knowledge and languages to describe that knowledge.** An ontology vocabulary describes knowledge about domains (Chapter 1). An ontology is described with a language (Chapter 1.9).

- **Reasoning about the existing knowledge.** The logic layer provides for language constructors and tools that allow for reasoning about ontologies to establish the consistency and correctness of specific concepts or to infer new concepts that are not explicitly stated. Assertions and rules are examples of language constructs defined in this layer.

- **Ensuring consistency and correctness of the knowledge.** The proof layer provides tools for generating the logical path of reasoning to establish consistency and correctness of concepts. They typically require the ability to describe rules.

- **Trusting the Semantic Web.** The Semantic Web does not assert that all statements found on the Web are “true”. Further, all statement occur in some context. As a result, each
application needs this context to evaluate the trustworthiness of the statements. The trust layer provides the mechanisms that are able to demonstrate the truthfulness or the quality of a resource with respect to a specific context of use. This is typically done through systems of authentication which require that logical reasons for trusting the data are provided.

Concrete work is currently underway to provide languages for each layer. Starting from the bottom, the three first layers are completed, the fourth layer is well on its way (with the OWL language standardized and applications being developed) and the other ones are being tackled. However, current solutions do not always agree completely with the vision [77]. Some adjustment may still be required in the layer cake. Moreover, further work has been conducted to provide query languages and retrieval engines for each of the four first layers, as described in [43].

2.3 Semantic Web Services

Web services (Section 2.1) are designed to provide interoperability between diverse applications. Composing services dynamically to create new functionality is necessary when the required task cannot be realized directly by the existing services. It is an ongoing research activity to automate this process, but accomplishing this goal with a human controller as the decision mechanism can be achieved. The main problem is the gap between the concepts people use and the data computers interpret. This barrier can be overcome using Semantic Web (Section 2.2) technologies.

Semantic descriptions of Web services are necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains. Existing technologies for Web services only provide descriptions at the syntactic level, making it difficult for requesters and providers to interpret or represent nontrivial statements such as the meaning of inputs and outputs or applicable constraints. This limitation may be relaxed by providing a rich set of semantic annotations that augment the service description.

The benefits of the integration include increased visibility of Web services, because open ontology frameworks allow for semantically expressive advertising on the Web that may be found by Web crawlers. They include better usability because of more expressive Web service descriptions. They include a smooth evolution from Web services for human users such as targeted by current industry (quasi-)standards toward Web services for personalized machine agents that assist the user.

In particular, the Semantic Web will allow giving richer descriptions of Web services (e.g., semi-structured data, types, inheritance, and semantic constraints). The key role of the broker may disappear, it may still be viable as a kind of search machine for Web services (with meta search engines on top), but it will lose its central role, because everyone may publish semantic descriptions and crawlers may find them. Personalized machine agents will take over the role of a service requestor from the human user. And, they may also do the composition for the human user.

A Semantic Web Service is defined through a service ontology, which enables machine interpretability of its capabilities as well as integration with domain knowledge.

The deployment of Semantic Web Services will rely on the further development and combination of Web Services and Semantic Web enabling technologies. There exist several initiatives (e.g. http://dip.semanticweb.org or http://www.swsi.org) taking place in industry and academia, which are investigating solutions for the main issues regarding the infrastructure for SWS.

Semantic Web Service infrastructures can be characterized along three orthogonal dimensions [16]:

- **usage activities**, defining the functional requirements, which a framework for Semantic Web Services ought to support;

- **architecture**, defining the components needed for accomplishing these activities;
• service ontology, aggregating all concept models related to the description of a Semantic Web Service, and constituting the knowledge-level model of the information describing and supporting the usage of the service.

These dimensions relate to the requirements for SWS at business, physical and conceptual levels.

2.3.1 The usage activities of a SWS

From the usage activities perspective, SWS are seen as objects within a business application execution scenario. The activities required for running an application using SWS include: publishing, discovery, selection, composition, invocation, deployment and ontology management, as described in the sequel.

The publishing or advertisement of SWS will allow agents or applications to discover services based on its goals and capabilities. A semantic registry is used for registering instances of the service ontology for individual services. The service ontology distinguishes between information which is used for matching during discovery and that used during service invocation. In addition, domain knowledge should also be published or linked to the service ontology.

The discovery of services consists of a semantic matching between the description of a service request and the description of published service. Queries involving the service name, input, output, preconditions and other attributes can be constructed and used for searching the semantic registry. The matching can also be done at the level of tasks or goals to be achieved, followed by a selection of services which solves the task. The degree of matching can be based on some criteria, such as the inheritance relationship of types. For example, an input of type Professor of a provided service can be said to match an input of type Academic of a requested service.

A selection of services is required if there is more than one service matching the request. Non-functional attributes such as cost or quality can be used for choosing one service. In a more specialized or agent-based type of interaction a negotiation process can be started between a requester and a provider, but that requires that the services themselves be knowledge-based. In general, a broker would check that the preconditions of tasks and services are satisfied and prove that the services postconditions and effects imply goal accomplishment. An explanation of the decision making process should also be provided.

Composition or orchestration allows SWS to be defined in terms of other simpler services. A workflow expressing the composition of atomic services can be defined in the service ontology by using appropriate control constructs. This description would be grounded on a syntactic description such as BEPL4WS [24]. Dynamic composition is also being considered as an approach during service request in which the atomic services required to solve a request are located and composed on the fly. That requires an invoker which matches the outputs of atomic services against the input of the requested service.

The invocation of SWS involves a number of steps, once the required inputs have been provided by the service requester. First, the service and domain ontologies associated with the service must be instantiated. Second, the inputs must be validated against the ontology types. Finally the service can be invoked or a workflow executed through the grounding provided. Monitoring the status of the decomposition process and notifying the requester in case of exceptions is also important.

The deployment of a Web service by a provider is independent of the publishing of its semantic description since the same Web service can have serve multiple purposes. But, the SWS infrastructure can provide a facility for the instant deployment of code for a given semantic description.

The management of service ontologies is a cornerstone activity for SWS since it will guarantee that semantic service descriptions are created, accessed and reused within the Semantic Web.

2.3.2 The architecture for SWS

From the architecture perspective, SWS are defined by a set of components which realize the activities above, with underlying security and trust mechanisms. The components gathered from
The discussion above includes: a register, a reasoner, a discoverer, a matchmaker, a composer or decomposer, an invoker, and a mediator.

The reasoner is used during all activities and provides the reasoning support for interpreting the semantic descriptions and queries. The register provides the mechanisms for publishing and locating services in a semantic registry as well as functionalities for creating and editing service descriptions. The discoverer and matchmaker will mediate between the requester and the register respectively during the discovery and selection of services. The composer is the component required for executing the composition model of composed services. The invoker will mediate between requester and provider or decomposer and provider, while the mediator will manage the resolution of mismatching between request and offer. These components can have different names and a complexity of their own in different approaches.

2.3.3 The service ontology of SWS

The service ontology is another dimension under which we can define SWS, for it represents the capabilities of a service itself and the restrictions applied to its use. The service ontology essentially integrates at the knowledge-level the information which has been defined by Web services standards, such as UDDI and WSDL with related domain knowledge.

This would include: functional capabilities such as inputs, output, pre-conditions and post-conditions; non-functional capabilities such as category, cost and quality of service; provider related information, such as company name and address; task or goal-related information; and domain knowledge defining, for instance, the type of the inputs of the service. This information can, in fact be divided in several ontologies. However, the service ontology used for describing SWS will rely on the expressiveness and inference power of the underlying ontology language supported by the Semantic Web.

2.4 Languages for describing Web Services

In this section, we take a look at existing approaches for describing Web Services (Section 2.1). Currently, several Web Service related languages are being discussed. They range from supporting sessions and transactions, quality of service issues, to publishing and subscribing to events. Considering the issue of actually invoking a Web Service in a more flexible way, three approaches are closely related, which we discuss in the following sections:

- **Web Service Description Language (WSDL):** describing the syntactic information of a service (the operations it supports, the transport and messaging protocols on which it supports those operations, the network endpoint of the Web service);

- **Service Choreography and Orchestration Languages:** describing the syntactic information for inter-operation between services. Examples of these languages are: the Business Process Execution Language for Web Services (BPEL4WS) [24], which is a formal specification for business processes using web services, and the Web Service Choreography Interface (WSCI) [35] specification.

- **Semantic Web Service Description Languages:** describing the syntactic as well as the semantic information for automatic description, composition, mediation, and invocation of Web services. OWL-S and WSMO are the most salient initiatives to describe semantic web services.

In the following, we briefly describe WSDL, and the two main SWS approaches: OWL-S and WSMO.

2.4.1 A language for describe Web Services: WSDL

Web services need to be described. A service requester needs to analyze a service for his requirements. Moreover, a web service needs to provide the following information:
the operations it supports;

- the transport and messaging protocols on which it supports those operations;

- the network endpoint of the Web service.

WSDL stands for Web Services Description Language [17]. It is an XML-based language for describing Web services. It specifies the location of the service and the operations (or methods) the service exposes:

- The <types> element defines the data types that are used by the web service. For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

- The <message> element defines the data elements of an operation. Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

- The <portType> element is the most important WSDL element. It defines a web service, the operations that can be performed, and the messages that are involved. The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language. WSDL defines four operation types:
  - One-way, the operation can receive a message but will not return a response
  - Request-response, the operation can receive a request and will return a response
  - Solicit-response, the operation can send a request and will wait for a response
  - Notification, the operation can send a message but will not wait for a response

- The <binding> element defines the message format and protocol details for each port. It has two attributes - the name attribute and the type attribute. The name attribute (one can use any name he/she wants) defines the name of the binding, and the type attribute points to the port for the binding.

A simplified fraction of a WSDL document is shown in Example 2.4.1. In this example the port-Type element defines “CitizenAddress” as the name of a port, and “getAddress” as the name of an operation. The “getAddress” operation has an input message called “CitizenName” and an output message called “Address”. The message element defines the parts of each message and the associated data types. Compared to traditional programming, CitizenAddress is a function library, “getAddress” is a function with “CitizenName” as the input parameter and “Address” as the return parameter. The type attribute points in this case to the “CitizenAddress” port.

**Example 2.1: A WSDL example: get citizen address from his/her complete name**

```xml
<message name="CitizenName">
  <part name="term" type="xs:string"/>
</message>

<message name="Address">
  <part name="value" type="xs:string"/>
</message>

<portType name="CitizenAddress">

<operation name="getAddress">
  <input message="CitizenName"/>
  <output message="Address"/>
</operation>
</portType>

<binding type="CitizenAddress" name="b1">
```
2.4. Languages for describing Web Services

One important aspect is the composition of processes. Composition is the task of combining and linking existing Web Services and other components to create new processes. It adds value to the collection of services, by orchestrating them according to the requirement of the problem. WSDL allows us to capture the various methods and its functionality is required to specify the methods supported by a service. However, it does not support constraints among those methods.

Besides this, the main disadvantage of the WSDL language is that it describes only the syntactic information of a service. The elements of document types must be populated with correct values so that they are semantically correct and are interpreted correctly by the service requesters and providers. This requires the definition of a vocabulary that enumerates or describes valid element values. For example, one vocabulary can contain a list of product names or products that can be ordered from a manufacturer. Further examples are unit of measures as well as country codes.

2.4.2 Ontology-based Web Service Description Languages

In this section, we elaborate on ontologies for describing web services that describe the syntactic as well as the semantic information. Actually, the current languages for describing web services, WSDL and their composition on the level of business processes (for instance BPEL4WS) lack semantic expressiveness that is crucial for capturing service capabilities at abstract levels. FLOWS [23], METEOR-S [99], WSDL-S [69], WSMO [32] and OWL-S [20] aim at representing web services that make use of ontologies.

The First-order Logic Ontology for Web Services (FLOWS) [23] does not strive for a complete representation of web services, but rather for an abstract model that is faithful to the semantic aspects of service behavior. The FLOWS model provides infrastructure for representing messages between services; the focus here is on the semantic content of a message, rather than, for example, the specifics of how that content is packaged into an XML-based message payload. FLOWS also provides constructs for modeling the internal processing of Web services.

METEOR-S [99] focuses on workflow management techniques semantic Web services. A key feature in this project is the usage of semantics for the complete lifecycle of semantic Web processes, which represent complex interactions between Semantic Web Services: development, annotation, discovery, composition and orchestration. A key research direction of METEOR-S is exploring different kinds of semantics, which are present in these stages.

WSDL-S [69] aims to provide a lightweight approach for creating semantic Web service descriptions. It provides simple extensions to WSDL to add semantics, thereby allowing semantic descriptions of Web services. The WSDL-S service ontology is aligned with WSDL. WSDL-S allows integration of semantic and non semantic descriptions of Web services.

Finally, WSMO and OWL-S represent the main efforts and reference models in this field. They take different approaches: WSMO stresses the role of mediation in order to support automated interoperability between services, while OWL-S stresses action representations to support planning processes that provide automated composition. In the rest of this section, we deepen the aspects associated with these two models.
OWL-S

OWL-S (previously DAML-S [19]) consists of a set of ontologies designed for describing and reasoning over service descriptions [20]. OWL-S approach originated from an AI background and has previously been used to describe agent functionality within several Multi-Agent Systems as well as with a variety of planners to solve higher level goals.

It supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S markup of Web services will facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation. OWL-S partitions a semantic description of a web service into three upper ontologies (Figure 2.4 [20]):

- **Service profile**: describing what the service does by specifying the input and output types, preconditions and effects.
- **Process model**: describing how the service works; each service is either an Atomic-Process that is executed directly or a Composite-Process that is a combination of other sub-processes.
- **Grounding**: containing the details of how an agent can access a service by usually specifying a binding to a WSDL operation.

![Figure 2.4: Top level of the OWL-S service ontology](image)

The Profile is used to describe services for the purposes of discovery; service descriptions (and queries) are constructed from a description of functional properties (i.e. inputs, outputs, preconditions, and effects - IOPEs), and non-functional properties (human oriented properties such as service name, etc, and parameters for defining additional meta data about the service itself, such as concept type or quality of service). In addition, the profile class can be subclassed and specialized, thus supporting the creation of profile taxonomies which subsequently describe different classes of services.

OWL-S process models describe the composition or orchestration of one or more services in terms of their constituent processes. This is used both for reasoning about possible compositions (such as validating a possible composition, determining if a model is executable given a specific context, etc) and controlling the enactment/invocation of a service. Three process classes have been defined: the composite, simple and atomic process. The atomic process is a single, black-box process description with exposed IOPEs. Inputs and outputs relate to data channels, where data flows between processes. Preconditions specify facts of the world that must be asserted in order for an agent to execute a service. Effects characterize facts that become asserted given a successful execution of the service, such as the physical side-effects that the execution the service has on the physical world. Simple processes provide a means of describing service or process abstractions such elements have no specific binding to a physical service, and thus have to be realized by an atomic process (e.g. through service discovery and dynamic binding at run-time), or expanded into a composite process. Composite processes are hierarchically defined workflows, consisting of atomic, simple and other composite processes. These process workflows are constructed using a number of different composition constructs, including: Sequence, Unordered, Choice, If-then-else, Iterate, Repeat-until, Repeat-while, Split, and Split+join.
2.4. Languages for describing Web Services

The profile and process models provide semantic frameworks whereby services can be discovered and invoked, based upon conceptual descriptions defined within Semantic Web (i.e. OWL) ontologies.

The grounding provides a pragmatic binding between this concept space and the physical data/machine/port space, thus facilitating service execution. The process model is mapped to a WSDL description of the service, through a thin grounding. Each atomic process is mapped to a WSDL operation, and the OWL-S properties used to represent inputs and outputs are grounded in terms of XML data types. Additional properties pertaining to the binding of the service are also provided (i.e. the IP address of the machine hosting the service, and the ports used to expose the service).

WSMO

The Web Service Modeling Ontology (WSMO) is a formal ontology for describing various aspects related to Semantic Web Services [32]. WSMO defines four main modeling elements for describing several aspects of Semantic Web Services: ontologies, goals, Web services, and mediators. In what follows, we will describe all these elements, insisting on their importance in reaching a truly Semantic Web Service technology.

In WSMO, ontologies represent key elements, having a twofold purpose: firstly they define the informations formal semantics and secondly, they allow to link machine and human terminologies. The WSMO ontologies give meaning to the other elements (Web Services, goals and mediators), and provide common semantics, understandable by all the involved entities (both humans and machines).

In WSMO, requesters of a service express their objectives as goals, which are high level descriptions of concrete tasks. A goal defines a capability; i.e. the state of the desired information space and the desired state of the world after the execution of a given web service. A capability is composed of the following elements:

- **Pre-conditions**: describing what a web service expects for enabling it to provide its service. They define conditions over the input.
- **Post-conditions**: describing what a web service returns in response to its input. They define the relation between the input and the output.
- **Assumptions**: they are similar to pre-conditions, however, also reference aspects of the state of the world beyond the actual input.
- **Effects**: describing the state of the world after the execution of the service.

![Figure 2.5: Meta-ontology for SWSs: WSMO [32]]
Every requester expresses its goal in terms of its own ontology, which, on one hand provides the means for a human user to understand the goal, and on the other hand, allows a machine to interpret it as part of the requesters ontology. Another advantage of using the goals is that the requester only has to provide a declarative specification of what it wants, and does not need to have a fixed relation with the Web Service or to browse through an UDDI registry for finding Web Services that provide the appropriate capability. A goal can import existing concepts and relations defined elsewhere, either by extending or simply re-using them as appropriate.

In order for this goal to be accomplished, the requester (by means of its information system) has to find an appropriate Web Service which may fulfill the required task.

Similar to the way the requester declares its goal, every WSMO Web Service has to declare its capability (that is, what it is able to accomplish) in terms of its own ontology. Moreover, it also outlines the interfaces of the service: the choreography and the orchestration (Figure 2.6 [83]). The former specifies the behavior interface for Web service consumption. A user of the Web service has to support this for consuming a Web service. A choreography description has two parts: the state and the guarded transitions. A state is represented by an ontology, while guarded transitions are if-then rules that specify transitions between states.

The orchestration defines how a Web service makes use of other WSMO Web services or goals in order to achieve its capability. It is not still clear which is the WSMO approach to represent orchestration but the idea is to adopt a description close to choreography: the state and the guarded transitions.

If the requester of the service and the Web Service that offers it use the same ontology the matching between the goal and the capability can be directly established.

Unfortunately, in most of the cases they use different ontologies, and the equivalence between the goal and the capability can be determined only if a third party is consulted for determining the similarities between the two ontologies. Another problem that may appear is the impossibility of the requester and of the provider of the service to communicate with each other, the reason for this being the heterogeneity of their communication protocols. For these reasons, WSMO introduces the fourth key modeling element: the mediators, which have the task of overcoming the heterogeneity problems. Four kinds of mediators are defined:

- OO-mediators: provide translation and harmonization between ontologies that are used by the Web services or any other WSMO component;

- GG-mediators: provide a way to match goals at different levels of granularity. For example, a GG-mediator may take the responsibility to refine the goal buy the ticket to the goal buy a train ticket upon recognizing that there is a subclass relation between the two concepts;

- WW-mediators: resolve the interoperability issues between Web Services at all levels: data, process, and protocol. Problems are solved at the level of both the single Web service choreography, and the orchestration of multiple Web services;

- WG-mediators: handle partial matches between goals of the client and the functionalities provided by Web services.
Concerning the needs for mediation within Semantic Web Services, WSMO distinguishes three levels of mediation:

- **Data Level Mediation**: mediation between heterogeneous data sources; within ontology-based frameworks like WSMO, this is mainly concerned with ontology integration.

- **Protocol Level Mediation**: mediation between heterogeneous communication protocols, i.e. translation between technical transfer protocols (e.g. SOAP, HTTP, etc.).

- **Process Level Mediation**: mediation between heterogeneous business processes; in WSMO, this is concerned with mismatch handling on Web Service Interface description for information interchange between Web Services and clients.

With regard to this, WSMO Mediators create a mediation-orientated architecture for Semantic Web Services, providing an infrastructure for handling heterogeneities that possibly arise between WSMO components and implementing the design concept of strong decoupling and strong mediation. Figure 2.7 shows the general structure of WSMO [18].

### 2.5 IRS-III Framework

The IRS project has the overall aim of supporting the automated or semi-automated construction of semantically enhanced systems over the Internet. In particular, IRS-III [29] is a framework allowing the publication, configuration and execution of multiple, heterogeneous web services, compliant with WSMO (Section 2.4.2).

IRS-III has three main classes of features which distinguish it from other work on semantic web services. Firstly, it supports one-click publishing of “standard” program code. In other words, it automatically transforms programming code (currently it supports Java and Lisp environments) into a web service, by automatically creating an appropriate wrapper. Hence, it is very easy to make existing standalone software available on the net, as web services. Secondly, by extending the WSMO goal and web service concepts, clients of IRS-III can directly invoke web services via goals - that is IRS-III supports capability driven service invocation. Finally, IRS-III services are web service compatible: standard web services can be trivially published through the IRS-III.

The main components of the IRS-III architecture are the IRS-III Server, the IRS-III Publisher and the IRS-III Client, which communicate through the SOAP protocol.

Publishing with IRS-III entails associating a specific web service with a WSMO web service description. When a web service is published in IRS-III all of the information necessary to call the service, the host, port and path are stored within the choreography associated with the web service. The IRS publishing platform is furthermore responsible for the actual invocation of a web service; additionally, it automatically generates wrappers which turn standalone code into a web service. The platform also copes with the syntactic level differences between the various web service platforms e.g. AXIS and Apache [4].

IRS-III was designed for ease of use, in fact a key feature of IRS-III is that web service invocation is capability driven. The IRS-III Client supports this by providing a goal-centric invocation.

![Figure 2.7: WSMO Mediator Structure](image)
mechanism. An IRS-III user simply asks for a goal to be solved and the IRS-III broker locates an appropriate web service semantic description and then invokes the underlying deployed web service.

2.5.1 Solving mismatch problems: data, process, and goal mediation

An important aspect of SWS technology is solving mismatch problems among heterogeneous domains. In the following, we report the approach proposed by IRS-III [29] for the mediation problem [14].

The Goal, Web Service and other Mediator descriptions associated with a Web service can refer to an OO-mediator in order to use ontologies which do not match. IRS-III handles data mediation by executing the mapping rules provided by an OO-Mediator (Figure 2.8 [14]). In IRS-III, the source and target components of an OO-mediator are ontologies. Furthermore, the source and target can be the domain ontologies of associated Goals or Web Services.

Figure 2.8: Data Mediation

WG-mediators, GG-mediators and WW-mediators have a data mediation capacity for transforming inputs between source and target components by using mediation services and have different roles within the process mediation.

The IRS-III approach splits the concept of process mediation into two distinct concepts: goal and process mediation. The former regards mismatches that occur during the process of selection of Web services for solving a Goal; the latter regards mismatches that occur during the invocation or composition of a Web Service.

We consider the two following examples about Goal Mediation. Figure 2.9 [14] shows a graphical illustration of the mediation taking place between a Goal and a Web Service via a WG-mediator. In this example, the Goal requested by the application takes two inputs (first and last names), which are transformed by the mediation service into one input (name) used by the target Web Service.

Figure 2.9: Mediation between a Goal and a Web Service. Two inputs of Goal are transformed into one input of the Web Service.

Since a mediation service can return only one output, IRS-III use a set of mediators between the goal (source) and the web service (target) in order to provide the required number of inputs to the target component as shown in Figure 2.10 [14]. In this example, each mediation service transforms (e.g. splits) the goal input (name) in one of the required inputs of the target component (first-name, last-name). The IRS-III engine can match the inputs and outputs for providing values as required.
As example of *Process Mediation*, we illustrate in the following the role of a GG-mediators during the orchestration of a Web service (Figure 2.11 [14]).

The provider of a Web service describes the orchestration through control-flow mechanisms, for instance: *(sequence G1 G2 M1)*. The *sequence* control command executes the given sub-goals (G1 and G2) in sequence. Figure 2.11 [14] shows the graphical representation of the GG-mediator connecting G1 to G2. This mediator supports the data flow between the sub-goals and the necessary transformations. The source goal (G1) produces one output (E1), which is transformed by the mediation service in one input (E2) used by the target Goal (G2). During the execution of the orchestration the input values (SC, TC, A) received by the current invoked Goal are sent onto the sub-goals through matching, then the associated GG-mediator (M1) are used to connect and forward results between sub-goals providing the necessary transformations through the mediation service.

**WW-mediators** can be used in a similar way to GG-mediators. In this case, the WW-mediator can provide mappings between the input values of the current Web Service and the Web Services in the orchestration.
Approaches to e-Government

Although service-oriented computing is a relatively new field, many e-Government applications have been developed and various approaches have been proposed. As introduced in Chapters 1 and 2, we are focusing on aspects associated with the knowledge representation (ontologies), and, in particular, the semantic description of services (SWS’s).

In this chapter, we take into consideration the e-Government projects investigating the possibility of describing services in order to develop service-oriented architecture. In particular, we only highlight here all of the project aspects that inspired the design criteria of our approach. We distinguish three kinds of approach on the basis of the adopted technologies:

- **Web information systems**: providing Web architecture to access PA services, where the use of semantic – and in particular ontologies – is absent or limited to represent service information (metadata).
- **Ontology-based applications**: considering e-Government projects where ontologies are involved.
- **Semantic Web Service-based applications**: exploring the possibility of using a SWS technology for the description, interoperability, and integration of different public administration services.

Finally, we summaries the relevant aspects and shortcomings of the reported works, before introducing our approach.

### 3.1 Web Information Systems in E-Government

#### 3.1.1 eGov Project

Within the eGOV project [33], an integrated platform for realizing on-line one-stop government was developed. This one-stop government platform allows the public sector to provide citizens, business partners and administrative staff with information and public services based on life events and business situations hence increasing the effectiveness, efficiency and quality of public services. More specifically, the eGOV on-line one-stop government platform:

- Implements the concept of on-line one-stop government. Allows public authorities to provide integrated services. Service integration is based on the open technology of Web Services;
- Achieves interoperability with legacy systems of different public authorities by means of open standards such as XML;
- Uses ontologies in order to enhance the navigation of the end-users to the e-Government portal. The navigation is based on a life-event hierarchy, which are implemented by means of an RDF schema [13].

From a technical point of view, the eGOV project identified and addressed three key development and implementation modules:
• eGOV one-stop government portal and network architecture;

• The eGOV middleware containing the content and service repository, the service creation environment and the service runtime environment;

• A common format for the data flow between the portal and the service repositories in different public authorities, which is termed Governmental Markup Language (GovML) [59]. The final GovML data structure consists of three sub-vocabularies, two for describing public services and one for life events.

Generic description data vocabulary for public services defines a common standard for the content of all of public authorities at a national level. Such governmental content is created only once, at a national level. This type of content could be normally based upon a governmental law, so it can be adopted by all of public agencies of a country.

Specific description data vocabulary for public services caters for the creation of content related to a public service provided by a specific public authority. It can be considered as a specialization of the generic description vocabulary, because the values of some elements of this vocabulary depend on the public authority, which provides the public service.

Data vocabulary for life events and business situations defines a set of elements necessary to describe any everyday life event or business situation. Elements of this vocabulary are a subset of the generic description data vocabulary for public services.

3.1.2 EU-PUBLI.com Project

The EU-PUBLI.com project [36] attempts to achieve cooperation amongst European agencies by designing and implementing a cooperative system that can interconnect, at application level, the different information systems, in order to (semi-) automate inter-country macro-processes providing complex e-Government services.

The basic vision consists on defining an overall architecture, respecting the autonomy of the single involved agency. The choice of a Service Oriented Architecture (SOA) ensures an high level of flexibility to the system. The cooperation of different agencies is achieved by making them responsible for exporting some views of its own information system as e-Services.

A Cooperative Gateway sub-system represents “where” and “how” e-Services are deployed; it includes the definition on how different cooperating organizations are organized and connected and how pre-existing legacy applications can be integrated in a common cooperative process.

An Orchestration Engine sub-system is the responsible of coordinating all the e-Services involved in a cooperative process; through “cooperation process definitions” (technically referred to as orchestration schemas), it dynamically finds and links suitable e-Services.

The core of the architecture is represented by an Information Manager: this sub-system stores both (i) e-Service definitions and (ii) orchestration schemas. Moreover, it stores and manages all of information needed to convert and map different legal frameworks.

Finally the Employees Front-End sub-system is responsible for the presentation to end users of the results of cooperative process executions.

Currently, both e-Service descriptions and orchestration schemas in the EU-PUBLI.com architecture are based on specific XML dialects.

3.2 Ontologies in E-Government

3.2.1 SmartGov Project

The overall aim of the SmartGov project [85] is to specify, develop, deploy and evaluate a knowledge-based platform to assist public sector employees to generate on-line transaction services. The SmartGov project, through its software platform, aims to minimize the reliance on IT skills to develop e-Government services.

The framework for e-Government services includes reference models for:
3.2. Ontologies in E-Government

- the processes behind the design and delivery of E-Government services;
- co-operation in public authorities, both internal and external;
- social acceptance of E-Government services.

The framework is underpinned by the E-Government services ontology. This is intended to provide a common understanding of the principles of E-Government services, an understanding from which people can communicate, discuss and build models of their own.

The key aspects of the SmartGov system are a knowledge base housing the domain knowledge required for e-Government services, and a SmartGov agent to allow the system to communicate with other IT systems.

Ontologies

The E-Government Services Ontology is the core of the framework for the SmartGov processes, business process models and social aspects. This ontology is based on two other ontologies: Enterprise Ontology and Meta Ontology.

The Enterprise Ontology is a useful ontology of generic business activities. Recognizing that many of these activities are common with public authorities, the E-Government Service Ontology can be built around it. The Enterprise Ontology defines concepts within four broad categories: activity, organization, strategy and marketing. It also imports a standard ontology of time. Some of the concepts formally defined within the Enterprise Ontology are listed below:

- Activity (Activity, Execute, Effect)
- Organization (Person, Machine, Legal Ownership)
- Strategy (Purpose, Hold Purpose, Risk)
- Marketing (Actual Customer, Sale, Competitor)
- Time (Time Interval)

The Meta Ontology provides the basic building blocks that are used to construct the E-Government Service Ontology. These are primitives that are defined outside the context of the ontology and for the purposes of the ontology are assumed to have no other meaning than the ones is assigned to them. Since the ontology is based upon the Enterprise Ontology, the Enterprise Meta ontology is the most reasonable starting point for the Meta ontology that is required for the SmartGov project. The terms used in the Enterprise Meta ontology are given below:

- Entity: a fundamental thing in the domain being modelled;
- Relationship: the way that two or more Entities can be associated with each other;
- Role: the way in which an Entity participates in a Relationship;
- Attribute: a Relationship between two Entities (the "attributed entity" and the "value" entity) in which, within the scope of the model, for any particular attributed Entity, the Relationship may exist with only one value Entity;
- State of Affairs: a situation; it consists of a set of Relationships between particular Entities; it can be said to hold, or be true (and conversely to not hold and be false);
- Achieve: the realization of a State of Affairs, i.e. being made true
- Actor Role: a kind of Role in a Relationship whereby the playing of the Role entails some notion of doing or cognition
- Actor: an Entity that actually plays an Actor Role in a Relationship
3. Approaches to e-Government

The development of the E-Government Services Ontology for the SmartGov project consisted of:

- Gathering the data (Interviews, Web documents, Word frequency counts of documents, Workshops);
- Defining the concepts;
- Structuring the ontology;
- Refer back to the experts.

3.2.2 ICTE-PAN Project

The ICTE-PAN Platform [57] consists of some customizable modules – building blocks (e.g. collaboration workflow management system, electronic argumentation system, electronic forms system, etc.).

Therefore, for every specific Government to Government (G2G) collaboration process that is intended to be supported with the ICTE-PAN Platform, it is necessary to configure and customize it appropriately. This configuration process requires the selection of the appropriate modules, which are needed for supporting this process, and then their customization and their integration according to the specific needs of the process.

In order to support the development of Collaborative Environments for Public Administration, based on the ICTE-PAN platform, a methodology for modelling collaborative operations in Public Administration (PA) was developed.

Ontologies

There is a big variety of cooperative activity (CA) types in the PA, which differ in the kinds of elements contributed by the participants, and the kinds of associations allowed among them. For this reason it is necessary during the definition of the activities of a collaborative process in the Process View, for each CA to proceed to modelling this aspect of it; therefore it is necessary to define the kinds of elements (e.g. issues, alternatives, arguments, programs, projects, tasks, etc.) which can be contributed by the participants in this CA, and also the kinds of associations which are allowed to be made among these elements (e.g. an alternative can be associated with an issue). Based on these definitions, the ICTE-PAN Platform for each case creates the appropriate electronic environment for the execution of this CA. In order to support the above definitions, an Ontology is required for the domains of PA policies and programs development, monitoring and evaluation, and also PA decision-making, consisting of the main concepts (i.e. categories or kinds of discussion elements) used in these domains and their associations.

Using this Ontology it is easily define the nature of each CA, by selecting a small subset of the kinds of elements and associations of the Ontology to be allowed in this CA. In this way, a high level of flexibility and adaptability to particular collaborative process requirements can be achieved, and a large variety of CA types of the PA can be supported.

Also eight CA "templates" have been defined, corresponding to the eight basic CA. Each of them corresponds to a specific subset of the kinds of elements and the associations of the above Ontology. These templates can be used both as typical examples of CAs in PA, and also for the quick definition of new CAs; a new CA can be based on any of these templates: in this way all the kinds of elements and the relations of the template are automatically inherited by this new CA and defined to be allowed and usable in it.

3.3 Semantic Web Service technology in E-Government

3.3.1 OntoGov Project

OntoGov project [76] investigates the possibility to apply ontologies in the e-Government domain in order to achieve self-managing systems, i.e. systems that can continually update themselves.
3.3. Semantic Web Service technology in E-Government

(at least to a certain extent automatically) according to the changes in the domain. On the other hand, the OntoGov project aims at improving back-office processes by taking into account the whole process life-cycle.

The goal of the project is to develop a framework that allows for change propagation and traceability, contributing in this way to the bridging of decision making with technical realization. For example, when a passport issuance service has to be changed due to changes in national and European legalization, the OntoGov system will enable to perform a change while keeping the consistency of the whole system.

This project adopts ontologies to model the business logic as well as the dependencies between different stakeholders that define this business logic in a collaborative way.

In particular, OntoGov defines the following ontologies:

- **Meta Ontology** that contains entities needed to describe e-Government services and it is based on the OWL-S (Section 2.4.2) and WSMO (Section 2.4.2) ontologies. This ontology has been introduced to model the dependency between a business rule and the service implementing it and to take into account the other specificities of the E-Government services. Similarly to the OWL-S ontology, the Meta Ontology consists of two parts: (i) the profile that is used for the service discovery and (ii) the process model that is used to describe the process flow. The former extends the OWL-S service profile ontology in several ways; particularly it introduces an explicit link between the service description and the elements of the Legal Ontology, additional entities connected to the description of the business model (e.g. properties defining required resources: persons or equipments), and standard metadata for the E-Government domain – since ontologies may advance metadata solutions. The process model is not described reusing the OWL-S process ontology, but it is combined with the existing work of the business process community. Similarly to the OWL-S process ontology, it distinguishes between the services and the control constructs, but it introduces a set of attributes connected to the e-Government: reference law, security level, cost, etc. A relevant difference in comparison to OWL-S process ontology is related to the conditions of a service. While OWL-S uses preconditions and effects to refer to the changes in the state of resources, OntoGov adopts the WSMO approach: preconditions for defining what a service expects for enabling it to provide its service, and postconditions for defining what the service returns in response to its input. Indeed, they establish the relationship between inputs and outputs.

- **Legal Ontology** that models laws in the form of ontology. In order to find a service that has to be updated after a change in a law, OntoGov establishes a reference between the service description and the law. Based on the analysis of the structure of the legal documents in three different countries (Switzerland, Greece, and Spain), OntoGov concludes that the legal documents have very similar structure independently of the country it is defined for. Therefore, it was possible to extract the general structure of a law and the represent it in a form of the Legal Ontology.

- **Domain Ontology** that contains domain specific knowledge, i.e. entities used in the properties of the Meta Ontology.

- **Service Ontologies** that describe concrete services. For each service, a service ontology that includes all of previously mentioned ontologies is defined, and it might include other Service Ontologies. It specializes concepts from the Meta Ontology (e.g. atomic service), establishes relationships between these concepts (e.g. what is the next or previous atomic service) and relates them to the domain entities from the Domain Ontology.

Ontology evolution can be defined as the timely adaptation of an ontology and a consistent propagation of changes to the dependent artefacts. Since the evolution is driven by the set of changes that have to preserve the consistency, the approach requires: (i) the explicit specification of changes that can be applied and (ii) the consistency definition. For the ontology evolution, OntoGov defines a set of ontology changes that includes elementary changes (e.g. AddConcept) and some more complex changes (e.g. MoveConcept).
Information about changes can be represented in many different ways. To communicate about changes, we need a common understanding of a change model and of a log model. Therefore, we introduce the Evolution Ontology and the Evolution Log. The Evolution Ontology is a model of ontology changes enabling better management of these changes. The Evolution Log tracks the history of applied ontology changes as an order sequence of information (defined through the Evolution Ontology) about particular change. Note that only a common understanding of the changes (achieved through the Evolution Ontology) and of a log (achieved through the Evolution Log) enables the synchronization between the evolving Domain/Service ontology and the dependent artefacts (e.g., applications based on this ontology) that have to incorporate or adapt to those changes. Further, the Evolution Log based on the formal model of ontology changes (i.e., on the Evolution Ontology) enables the recovery from 'failure' since it makes possible to undo and redo applied changes in an unlimited way.

In order to help PA finds out the right changes needed to synchronize the service with a law, OntoGov introduces the so-called Lifecycle Ontology. It describes the information flow and the decision-making process in the PA. It is intended to support the transition from knowledge acquisition to implementation. Therefore, it includes entities for documenting design decisions and underlying rationale. In this way, it gives concrete clues on how a service has to be modified.

3.3.2 Terregov

Terregov project [91] is an on-going project at an early stage of development. It aims to address the issue of interoperability among services provided by government agencies, down to the regional and local level. It is based on the following principles:

- Web Services are the basic building blocks for interoperability between different administrations;
- Business Processes (eProcedures) are built by orchestrating the execution of Web Services and are themselves published as Web Services;
- Semantic, Ontology and Natural Language processing (NLP) are an excellent support for the Civil Servants in the discovery of information and services;
- A collaboration environment for Civil Servant is essential (Community of Practice);
- Compatibility with specific local e-Government rules and openness to Open Source is fundamental;

Its architecture is composed by a framework and intelligent agents that will offer configuration/reconfiguration of service workflows by selecting competing Web Services on the basis of WS performance, and by composing dynamic workflows based on semantic descriptions.

The architecture can be split up into two internal developments: those that provide interoperability to the system and between the system and external components, and those that are built for being used or managed by civil servants and other users of the system. We focus on the first aspect, because more relevant to the topics of the present thesis; particularly we outline the Semantic Web Service Registry, and the eProcedures.

Semantic Web Service Registry is in charge of registering Web Services as well as allowing searches for these services based on semantic concepts, dealing with the requirement for Web Services to dynamically discover each other. This discovery issue raises the need for using semantics into the process. Then, the web services definition will use ontology concepts for their representation, based on existing standards. Terregov’s Web Services are stored semantically using OWL-S (Section 2.4.2). A Semantic Web UDDI Registry stores semantic descriptions of TERREGOV Web Services, and make them available to clients. This module replicates its semantic services advertisements in a regular off-the-shelf UDDI registry, so that non-TERREGOV-enabled client can still browse, search and find TERREGOV Web Services. The core of the Semantic Web Registry module is the Semantic Search Engine; the purpose of this module is to search published
3.4. What needs to be done?

Terregov Web Services based on semantic concepts. This search is based on inference using rules and semantic concepts defined in an ontology. The result of the search will be the advertisements or the unique identifier of the Web Service matching the search request.

eProcedure is a procedure involving public administrations that can be initiated by a Civil Servant; in a digital environment an eProcedure is represented by a formal description of a workflow, whose steps involve the execution of services. Assuming that within an e-Government scenario, the services are implemented as Web Services, the eProcedure Module is the module responsible for managing eProcedures involving the orchestration of Web Services calls. A possible solution to formally describe eProcedures is the use of the BPEL [24] workflow description language. Current efforts in Terregov project are supporting the idea of having a BPEL-compliant workflow engine, where both predefined workflows, and dynamically created ones, are stored and executed.

3.4 What needs to be done?

Starting from the analysis of the above projects, we came to the following conclusion:

- eGov, and EU-PUBLI.com define architectures based on Web services interfacing PA legacy systems. Web service technology allows to introduce interoperability between PA back-offices and create a scalable and flexible environment where PAs can cooperate to offer innovative services to citizens and businesses. The goal of these projects is to achieve one-stop E-Government by: (i) providing all of relevant information needed to perform task; (ii) requiring only the minimal set of information from citizens in order to avoid redundancy (e.g. one has to fill in similar forms with the same data several times when he/she changes the residence). XML dialects are used to define metadata and orchestration of services.

- SmartGov and ICTE-PAN projects use ontologies for representing e-Government knowledge. They developed two ontologies describing the profile of a service; the two approaches are really different: the former supplies an ontology trying to cover all of e-Government service-supply aspects that is used to delivery e-services by domain experts; the latter defines some customizable modules that allow to create specific ontologies describing cooperation scenarios between distinct PAs.

- OntoGov and TerreGov adopt SWS approach for describing services provided by PAs. However, they do not completely take advantage of SWS technology. OntoGov develops an own ontology for describing e-Government services mixing aspects of the two main approaches OWL-S and WSMO. The motivation is to satisfy some e-Government requirements that are not addressed by OWL-S or WSMO, but, in this way, OntoGov cannot use algorithms, tools, and future improvements provided by these two proposed standards. Beside this, OntoGov focuses on life-cycle management of the described services and mediation aspects between heterogeneous sources are not considered. TerreGov adopts OWL-S for describing, and discovering services but it will use BPEL language (lacking semantic expressiveness) for describing composition of services (eProcedure).

All of these projects provide significant inputs for our research. We extend their results by proposing a reusable semantic-based framework allowing

- the description of e-Government service-supply scenarios representing all the involved concepts, roles, constraints, relations, etc. by different actor’s viewpoints (e.g. end-users, public and private organizations, managers, politicians, etc.), and

- the integration of existing standard for SWS. The first point allows the configuration and re-configuration of e-Government applications; this point allows the description, composition, mediation, and invocation of Web services.

Objectives of our approach are the following:
3. Approaches to e-Government

- **General purpose.** The aim of our work is not to represent all of existing concepts and relations connected to the e-Government domain. As in the ICTE-PAN project, the idea is to create some modules driving domain experts to develop domain ontologies describing the specific scenario and helping developers to implements SWSs on the base of the domain expert’s representations. In particular, these modules outline a generic service-supply scenario that domain experts can adapt and extend using different levels of granularity on the bases of scenario characteristics. The result is a re-usable, extensible, and flexible model.

- **Multi-Viewpoints approach.** All of the introduced projects adopt a service-oriented approach. The service provided by organizations is the central concept. For instance, in the eGov and OntoGov projects, the user viewpoint is described by a taxonomy of life events simply representing the way to arrange the services into the portal. In our vision, the life event concept plays a central role prompting the supply of services by several organizations and representing the point of contact among different actor viewpoints. It represents the starting means for our multi-viewpoints description of the involved scenario knowledge.

- **Contextualization.** Our approach allows to contextualize – i.e. describing various notions of context, non physical situations, topics, plans, beliefs, etc. as entities – an e-government scenario in terms of descriptions. In particular, we distinguish between descriptive entities – that are independent views on a scenario by different involved actors – and the actual objects they act upon - representing the concepts of the actor’s vocabularies. This captures that multiple overlapping (or alternative) contexts may match the same world or model, and that such contexts can have systematic relations among their elements.

- **PA Autonomy and Cooperative Development.** The domain standardization (introduced by the different e-Government projects) can help, but it does not necessarily unify the aims and languages of all the involved organizations and actors. Each of them should keep its autonomy describing its own domain. Actually, distinct organizations could use or describe the same concepts differently. This implies addressing the issues of mediation between heterogeneous sources, but allows the co-operative development of an e-Government application.

- **Business Process and Interaction description.** The process flow of e-Government processes can be modeled using standard control structure and tasks. Different projects adopt different approaches. For instance, OntoGov adopts the OWL-S process model while TerreGov will adopt BPEL. Unlike the other approaches, we introduce an Interaction description that is a useful means of introducing model checking to the requirements gathering process as well as a key but too often neglected component of business process. Actually, we distinguish between a plan describing processes and organising concepts within an actor’s viewpoint and interaction describing mutual actions involving two different actor’s viewpoints.

- **Delegation.** Service integration will allow organizations to delegate the execution of some of their tasks to external organizations. This includes looking for and identifying the right organization. In our approach, we explicitly define how declare delegate tasks. This aspect is strictly connected to the above interaction description representing the protocol to consume the delegation.

- **SWS standards.** SWS technology addresses the integration and interoperability issues between services provided by heterogeneous organizations. However, some e-Government requirements cannot be represented by existing SWS approaches. In our approach, we clearly distinguish between e-Government scenario description – addressing the e-Government requirements – and SWS descriptions. The two levels are integrated without affecting but taking advantage of SWS standards. Moreover, this level separation allows to adopt WSMO as well as OWL-S approach.

In the following chapters, we provide a detailed discussion about our framework.
II

Second Part
A Semantic-Based Framework

A single conceptual model is often described as providing an abstract framework to establish common understanding, identification of issues and a context for discussion. A framework suggests a safe foundation and structure within which to build, and several conceptual models can fit inside the same framework. Disparate models can fit together and be coherent inside a unifying framework.

The field of government services is large. Starting from specific requirements and defined conceptual models, our goal is to provide a general framework with which most Public Administrations (PA’s) – or generally organizations – can identify, from which they can work when designing and delivering e-Government services. Such a general framework can be adapted and applied as appropriate. Features of our proposal have been stated in Section 3.4.

We argue that a complex semantic layer for managing government services needs to be modeled – and a middleware system designed on such a model – in order to meet the requirements of real-life applications. In particular, we identify three knowledge levels:

- **Guidelines**, describing the context that bounds the creation and evolution of services: legislations, policies, and strategies influencing the development and management of an e-Government service-supply scenario.
- **Configuration**, describing the context in which services are supplied: requirements, resources, actor’s role, business processes, and transactions of an e-Government service-supply scenario.
- **Service delivery**, adopting Semantic Web Service (SWS) technology as the base for the description, discovery, composition, mediation, and execution of (Web) services.

As a result, the integration of e-Government applications with SWS’s requires a framework which maps and combines the afore described knowledge levels. The aim of our work is to provide such a framework. Our approach is grounded on a technological paradigm capable to fit a general distributed organization of knowledge, with focus on the supply of services. The proposed framework should be considered from two different dimensions:

1. **Conceptual modelling**: this is a double stages process that first creates a conceptualization of the reality in terms of conceptual models, and then uses ontologies to represent the semantic structure of involved knowledge, enabling knowledge use and reuse. The result is an ontological framework for service-oriented e-Government applications.

2. **Creating of an infrastructure for semantic interoperability**: software modules are used to implement the functionalities of a middleware system that enables the automated interpretation and paves a common ground for services. The result is a semantically-enhanced middleware for service-oriented e-Government applications.

The main contributes of this thesis regard the first issue, on which we shall focus in the rest of this chapter. The creation of an infrastructure for semantic interoperability is discussed in Chapter 7.

The structure of this chapter follows the developing methodology we adopted for building the ontological framework: Section 4.1 states the criteria and choices driving our approach, and
Section 4.2 presents the conceptual models that are the foundations of our work. These two sections represent the specification and conceptualization phases (Section 1.4) of our building methodology. Then, Section 4.3 specifies the modelling approach and knowledge modelling tools we used in order to map the above conceptual models onto ontologies. Finally, in Section 4.4, we present the architecture and three ontologies that have been developed and combined into a sound ontological framework.

4.1 Requirements

In this section, we specify the requirements driving our development approach. In order to describe a conceptual model – and the mapping ontology framework – to be applied in the e-Government field [76] and to address the objective stated in 3.4, we defined the following set of criteria:

- **Modelling primitives.** The conceptual model intends to capture the necessary modelling primitives that both provide adequate expressive power and are well understood thereby allowing the semantics to be precisely specified and complete inference to be viable.

  In the e-Government domain a high degree of flexibility and expressiveness is required. For example, the ability of an ontology to express constraints (e.g. only adults can require a move house service) would be an advantage.

- **Rules.** Two types of rules might be distinguished: axioms as a standard axiom schemata (e.g. symmetry, transitivity, inverse) and specific rules.

  For instance, if an organization supplies services, then a service is supplied by an organization. Therefore, supplies and isSuppliedBy are inverse properties. Further, if an organization supplies a service that is about some topic, then this organization has knowledge about this topic. It can be formalised in a form of IF-THEN rules\(^1\):

\[
\text{\forall x} \ (\text{Organization} \ x \ \text{supplies} \ s) \rightarrow (\text{Service} \ s \ \text{isAbout} \ t) \rightarrow (\text{Organization} \ x \ \text{hasKnowledge} \ t))
\]

- **Meta-Models.** Since the e-Government domain is big, it cannot be described by a unique conceptual model. Several conceptual models 4.2 can cover its different aspects. In order to capture and combine the main elements of these conceptual models and to match the reusability requirement of our approach, we propose a meta-model, i.e. a model expressing the modeling process [42]. Particularly, the part of the modeling process to model is the creation of the conceptual model. The obtained meta-ontologies are domain-independent ontologies specifying the schema to be followed by the modeling process and the general concepts and relations that may be extended and adapted.

  Applying a specific scenario to the meta-model, the result is a model for a specific application (Figure 4.1 [42]). Starting from meta-ontologies, the resulted ontologies describe application-dependent concepts, relations, axioms, etc.

  In e-Government it would be useful to adopt meta-ontologies describing a service-supply scenario. These ontologies can be further reused and extended for developing specific-scenario ontologies.

- **Modularization.** It is a common engineering practice to extract a well encapsulated body of information in a separate module that can later be reused in different contexts. It is inefficient and error-prone to always build the ontology from scratch. Rather, these models

\(^1\)This rule is written in OCML language (Section 1.9.1)
should be built by reusing smaller, well-defined components. An ontology can reuse modules from other ontologies through modularisation.

Module inclusion might be supported by allowing an ontology to include other ontologies, thus obtaining the union of the definitions from all included models. All definitions from an included ontology are automatically available in the including ontology.

The ontological framework adopts modularisation including modules from existing ontologies and defining new ones. According to meta-modelling, the new modules are building blocks for scenario-specific ontologies; thus, they can be configured and customised appropriately. This configuration process requires the selection of the appropriate modules and then their customisation and integration according to the specific needs of the scenario allowing representations at different levels of granularity.

For instance, an ontology describing the moving house scenario might include any ontology describing domain specifics terms such as “address”, “citizen”, ..., and the ontology defining meta-concepts connected to the description of involved services.

- **Lexical layer.** The idea of the E-Government initiative of European Union is to bridge the gap between citizens and the state as well as between different nationalities. Therefore, prerequisite for such communication/interoperability is multi-vocabulary support.

  For example, to automate the procedure of moving house, it may be necessary to describe terms in different “languages”. The existing place of residence and the new one might be in different countries that have different official languages; different organizations might use different terms for expressing the same meaning. The whole procedure can only be performed automatically if a common understanding of used terms is achieved. For example, the service has to recognise that the terms “address” and “indirizzo”, or “address” and “location” have the same meaning.

- **Efficient reasoning.** Inference mechanisms for deduction of information not explicitly asserted is an important characteristic of ontology-based systems. A lot of knowledge in the e-Government domain has to be modeled at the instance level. Therefore, an ontology reasoning system has to provide efficient means for reasoning at the specific level.

- **Compliance with standards.** The compliance of e-Government services with standards for semantic web services is absolutely necessary, since this is the only realistic way that all of these distributed heterogeneous services can be connected.

- **Supporting Web Services modelling.** We apply SWS technology in the e-Government field. Therefore, it is important to check whether and with which degree the ontological framework and its infrastructures provide support for modelling of web services.
It is important to note that some of the above criteria are only related to the choice of the ontology language and some of them to the tools that allow developing ontologies in the language or reasoning on ontologies.

### 4.2 e-Government Conceptual Models

There are several definitions of conceptual model. They depend on the intended use of the model. In our vision, we refer to it as an abstract definition of how to describe and develop a domain of interest: a model of modelling\(^2\). It points out the building blocks that are used in models of the domain, the relationships between the building blocks, and how to build models.

In this section, we introduce the conceptual models representing the foundation of our work. As the e-Government field involves several aspects, we do not refer to a unique model. The main elements described by these existing – and in some cases well known – conceptual models are combined and mapped in our ontological framework. The following models clarify our approach respectively outlining the elements of a government service-supply scenario, specifying the actors and roles of an e-Government application, introducing the life event metaphor for our multi-viewpoint approach, and defining the considered aspects of e-Government business processes.

**The Government Service-Supply Scenario Conceptual Model.**

Figure 4.2 shows a general view of an government service-supply scenario [85], representing the main concepts, roles, elements, actors, and existing relations among them.

We distinguish two main actors: *users* (citizens and businesses), and *Public Administration* (PA). The latter co-operates with other public and private organizations to jointly deliver *services*. Public services meet the user’s *needs*. Users use PA services creating *resources* (e.g. documents, information, etc.) to consume them. This resources are – electronically or physically – provided to the PA when the users requires or during the accomplishment of a service. Further resources are provided by connected organizations and managed by the PA providing the service. The user needs influence the *legislation*. The legislation influences the PA work, the private and public organizations, the *policy* and *strategies* driving the creation of services – and thus indirectly the services.

![Figure 4.2: The government service-supply scenario conceptual model.](image_url)

\(^2\)We can speak of conceptual meta-models.
The e-Government System Conceptual Model.

In relation to the previous model, we defined the following one emphasizing the basic structure of an e-Government system. It focuses on the description of existing roles and the process for the design, development, delivery, and maintenance of e-Government services (Figure 4.3).

In particular, the main actors involved in an e-Government system are the following:

- **Politician** defines the laws regulating and influencing the service-supply scenario.
- **PA Manager** organizes and supervises public services. On the basis of the laws, he defines policies and strategies about the implementation of new services or the alteration of existing ones.
- **PA Domain Expert / Service Worker** possesses the necessary background knowledge for designing public services. He possesses a very good knowledge about the e-Government domain. This knowledge is needed for the design of services. Based on the interpretation of a law and the strategies defined by a manager, a domain expert describes a service as the sequence of activities that have been done - which represents a business process - and the interaction with other actors (users and/or other organizations) - which represents a transaction for using the service or delegating part of its functionalities. During the development of an service, the domain experts may have to collaborate with the developers to communicate to them their domain knowledge.
- **Developer** possesses the necessary technological knowledge for the development of an electronic public service. He designs the system from scratch, defining system architecture, database schema, user interface and functionality. He translates the description provided by domain experts in terms of activities that implement the service (e.g. the implementation of Web Services and their description with SWS). He also provides the necessary interfaces for data exchange between the electronic service platform and the back-end systems.
- **End User** makes use of the service. End user may be citizens, businesses, PA workers, and other PA’s.

Due to the changes in the political goals of a government, changes in the environment, or changes in the needs of the people, the politicians might make the revision of a law. The change in the law causes a modification in the policies/strategies in which this law has been implemented. Domain expert must understand the changes and reconfigure the service description.

It is important to note that the vocabulary could be different at different levers: indeed, a politician uses quite different languages and concepts as compared with a programmer, and, overall, end-user knows what he/she wants to achieve (moving house, getting married, etc.), but does not know exactly which services match his/her needs.
The Life Event Metaphor Conceptual Model.

The life-event metaphor considers government operation from the perspective of everyday life. For instance, Table 4.1 shows the categorization proposed by the Italian Government [27].

<table>
<thead>
<tr>
<th>Citizen Oriented</th>
<th>Business Oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Being citizen</td>
<td>Opening activity</td>
</tr>
<tr>
<td>Having a baby</td>
<td>Modifying activity</td>
</tr>
<tr>
<td>Having a family</td>
<td>Improving activity</td>
</tr>
<tr>
<td>Health</td>
<td>Ending activity</td>
</tr>
<tr>
<td>Living</td>
<td>Sponsoring activity</td>
</tr>
<tr>
<td>Studying</td>
<td>Employee Management</td>
</tr>
<tr>
<td>Working</td>
<td>Owning Estate</td>
</tr>
<tr>
<td>Retiring</td>
<td>Paying taxes</td>
</tr>
<tr>
<td>Paying taxes</td>
<td>Recording brand and patent</td>
</tr>
<tr>
<td>Doing or suffering a complaint</td>
<td>Import and Export</td>
</tr>
<tr>
<td>Using transportation services</td>
<td>Doing or suffering a complaint</td>
</tr>
<tr>
<td>Leisure time and culture</td>
<td>Protect the environment</td>
</tr>
<tr>
<td>Practicing sport</td>
<td></td>
</tr>
<tr>
<td>Going abroad</td>
<td></td>
</tr>
<tr>
<td>Nature and Environment</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Life Events proposed by Italian Government

One life-event has to comprise all of the services as well as processes necessary to solve the user’s problem. In this way, all services necessary to solve a particular problem or situation, are either linked or integrated into one single service [74].

Figure 4.4: The way in which public services are grouped and integrated into life events.

In existing approaches (Section 3.4), the life event is simply the way to arrange the portal contents using a well known user metaphor (Figure 4.4). Actually, the life event is the point of contact among all the actor viewpoints defined in above conceptual models. The various views naturally focus on different aspects of a life event (Figure 4.5). Several viewpoints can be connected to the same life event. We may distinguish multiple classes of end-users (generic citizen, elderly or young people, PA employee, etc.), public organizations (national, regional, or local level), private organizations, etc. Moreover, organizations may supply
services in cooperative or competitive way; e.g. a regional public organization may supply a service involving several local and private organizations. The same event can be described by all of these viewpoints.

Considering the moving house life event, the citizen’s viewpoint will include the description of his/her need to move house from the old address to the new one; the PA’s viewpoint (domain experts) will describe the offered services for storing the change in its and eventually other organizations legacy systems and helping the user moving house; the manager’s viewpoint will define some policies and strategies influencing the service implementations, such as which organizations should be involved in the automatic change of address notification, or helping specific class of citizens with the introduction of moving house benefits; the politician’s viewpoint will describe the laws ruling the moving house life events.

Business Process: Plan and Interaction Conceptual Models

A business process is the description of steps necessary to carry out a business activity regardless of the systems involved. It provides a high level view of the steps involved and can be used to model, benchmark and document existing or future designs.

An executable business process is a kind of business process whose lifecycle is controlled by one or a combination of business process management systems. A business process is long running. Its execution may be limited to minutes or hours, like the session of a web-based application, or spans days, months, or years.

A business process activity (task) represents a short-lived interaction between users or, in some cases, systems. It can be viewed as one step in an executable business process.

The process flow of e-Government processes can be modelled using the standard control structures and objects describing business processes. However, while in private organisations the decisions for process definitions are mainly based on time, cost and quality criteria, government processes must be in accordance with the existing law and regulations from different levels (state, region, and municipality). Therefore it is very important to document the laws and regulations the process is based upon not only for the whole process but maybe also for specific activities.

Beside this, in our approach, we clearly separate two important aspects of the business process: the Plan, defining the direct-decomposition of a process into tasks and their organization within an actor’s viewpoint, and the Interaction, describing the mutual actions (transitions) between two distinct actors in order to achieve a task (delegated task) (Figure 4.6).

In the next paragraph, we detail the Interaction Model because it is an important aspect that is too often neglected in other e-Government approaches.

Note that Web service orchestration and choreography definitions, available on the W3C glossary [48] and introduced in Section 2.4.2 for the WSMO approach, can be brought back to the Plan and Interaction definitions, respectively. However, those definitions refer to only two limited aspects (connected to Web service description) that should be described in an e-Government scenario.
Other aspects can be modeled with Plan and Interaction models but not with orchestration and choreography. For instance, Web service orchestration does not (and must not!) consider the interaction with the human user. The description of the process involving a negotiation with the human user (e.g., repetitively invoke a Web service until the result satisfies the user) is knowledge that should be described by a Plan at the e-Government application and not at the Web service level.

In other words, orchestration and choreography are respectively particular Plan and Interaction descriptions at a technical and Web service-specific knowledge level.

**Interaction Conceptual Model**

The interaction conceptual model describes the category of business process to be conducted between two actors: the service provider and the requester.

Interaction knowledge provides the vocabulary used to describe the different mutual actions that may be established between distinct service requester and provider viewpoints. It provides requesters (resp. providers) with the vocabulary to describe the kind of interaction they agree to participate in (resp. are able to support). Further, from the requesters point of view, the interaction vocabulary that is put at their disposal is also a way of knowing what possible interactions actually exist, as well as the alternative existing solutions that they are perhaps not aware of.

There are many different possible ways to perform business. The vocabulary must allow for the description of the kind of business interaction the provider is able to supply and the requester wants to enter into.

In the e-Government field, we can refer to a four-stage framework for the classification of e-services regarding to the level of on-line sophistication of these services provided by the European Commission [22]:

- **Information**: includes on-line information about public services (e.g. information necessary to start the processes to obtain the public services available on the websites).

- **One-Way Interaction**: downloading of forms (e.g. downloadable or printable forms to start the processes to obtain the public services on the websites) is available.

- **Two-way Interaction**: comprises processing of forms, including authentication (electronic forms to start the processes to obtain the public services on the websites). The communication with institutions and persons in charge and the notification about the case progress is also included in this stage.
4.2. e-Government Conceptual Models

- **Transaction**: comprises full electronic case handling of processes by the service provider, including decision, notification, delivery and payment if necessary.

The services of each lower stage are included in the next higher stage.

Polovina and Hill [80] proposed a framework for designing multi-agent scenario based around transactions. They have revealed how transactions transcend the myriad quantitative and qualitative aspects of an organization, its transactions with other organizations and the environment. Transactions recognise the optimal relationships that occur through a mutually beneficial exchange in resources. Their transaction framework thereby provides organizations with an underlying basis by which they can interrogate their divergent systems, build an organizational memory and thereby enable effective organization knowledge management.

The basis of such a framework is the conceptual graph (Section 1.7.5), being Polovina’s Economic Accounting model that in turn is based on Geerts & McCarthy’s respected Event Accounting model [79] [45]. We see that this model captures the full essence of a transaction (Figure 4.7).

![Figure 4.7: Polovina’s accounting model. The concepts are represented by rectangles and relations by circles; the direction of the arrows determines the direction of the reading. A concept that appears with a specific referent (e.g. Economic Resource: *c) represents an instance.](image)

- **Transaction.** A transaction is an exchange of resources between two or more parties, who range from individuals to large organizations. Each party enters into the transaction in the belief that each of them benefits from giving up lesser valued resources for more valued ones; i.e. a win-win situation. The value placed on each resource can be both quantitatively and qualitatively defined, thus “every debit has a credit” extends beyond the monetary value exchange of each resource.

- **Economic Event.** This is an event that, together with the other economic events that make up a given transaction, triggers the exchange of economic resources.

- **Economic Resource.** This represents each resource that is available to be exchanged.

- **Inside and Outside Agent.** This is the view from the organization’s own perspective. An outside agent is the complementary to the inside agent. The transaction occurs when both the inside and outside agents’ needs have been satisfied.

In our approach, we adopt such a framework for describing interactions within the e-Government service-supply scenario. We adapted it to model generic transition events (and not only economic events) representing a transition of a resource (not only economic) from one party to the other. In fact, we should consider both *Communications* (Two-way Interaction) and *Transactions*: they differ on the number of transition events that can be defined: one for the communication and at least two for the transaction description. Moreover, we distinguish three kinds of communication:

- **Notify:** the requester gives information to the provider.
• Apply: the requester asks for a service that cannot be consumed on-line by supplying the necessary information to start the service.
• Enquiry: the requester retrieves information from the provider.

Beside this, we extended the Polovina’s model with the description of the activation conditions of a transition event, in order to model the sequence of transitions within an interaction.

4.3 Meta-Modelling the Conceptual Model

An explicit specification of a conceptual model is an ontology (Chapter 1).

The aim of our building methodology is to combine and map the introduced conceptual models (Section 4.2) onto meta-ontologies following the stated requirements (Section 4.1).

In this way, the resulting ontologies are reusable models for e-Government applications; they describe the global, uniform view of the scenario, using commonly accepted and standardized concepts and properties (attributes and relations), and possible domain-specific extensions. Their concepts/properties are either mapped or being mapped onto those in the application-specific models.

The meta-modelling approach is the basis for the cooperative and distributed development of an application-specific knowledge level allowing involved actors to keep their autonomy in the description of their domains: they follow the proposed schema (meta-graph) to create ontologies extending and adapting the meta-ontologies. All of the obtained ontologies form the application-specific knowledge level: each of them can define one or more actor’s viewpoints and may refer to other application ontologies.

For instance, in Figure 4.8, distributed nodes (representing working groups of the cooperative development) describe one or more actor’s viewpoints: Node A describes the end-user viewpoint; Node B and Node C represent two distinct organizations; the former describes politician, manager, and domain expert viewpoints, the latter only the manager and domain expert viewpoints. Note that different vocabularies can be used to describe different viewpoints within the same node.

4.3.1 Ontologies for Meta-Modelling

To facilitate the building of meta-ontologies we refer to existing reference ontologies. We extend them and reuse some of their modules to create our ontologies. Actually, we refer to DOLCE [75] as upper ontology for describing domain concepts, its Description & Situation module [44] as approach for knowledge contextualization (i.e., representing various points of view on a scenario, possibly with different granularity), and WSMO [32] as meta-ontology for describing Web services.

![Figure 4.8: Use of meta-models for cooperative building of application-specific knowledge level.](image-url)
Upper ontologies: DOLCE, Description&Situations

Also called foundational, serve as starting points for building domain ontologies, to provide a reference point for easy and rigorous comparisons among different approaches, and create a framework for analyzing, harmonizing and integrating existing ontologies and metadata standards. They are conceptualizations containing specifications of domain-independent concepts and relations, based on formal principles from linguistics, philosophy and mathematics. Upper ontologies are ultimately devoted to facilitate mutual understanding and interoperability among people and machines.

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) belongs to the WonderWeb project Foundational Ontology Library (WFOL) and is designed to be minimal in that it includes only the most reusable and widely applicable upper-level categories, rigorous in terms of axiomatization and extensively researched and documented [75]. Figure 4.9 [75] shows the taxonomy of DOLCE basic categories.

DOLCE is based on a fundamental distinction between enduring and perduring entities. The main relation between Endurants (i.e. objects or substances) and Perdurants (i.e. events or processes) is the participation: an endurant “lives” in time by participating in a perdurant. For example, a person, which is an endurant, may participate in a discussion, which is a perdurant. A persons life is also a perdurant, in which a person participates throughout its duration. Qualities can be seen as the basic entities we can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths or electrical charges. Spatial and temporal qualities encode the spatio-temporal attributes of objects or events. Finally, Abstracts do not have spatial or temporal qualities, and they are not qualities themselves, e.g. (quality) regions or sets. In particular, regions are used to encode the measurement of qualities as conventionalized in some metric or conceptual space.

DOLCE has been chosen due to its internal structure – rich axiomatization, modularization, explicit construction principles, careful reference to interdisciplinary literature, commonsense-orientatedness. In addition, being part of the WFOL, DOLCE will be mapped onto other foundational ontologies – possibly more suitable for certain applications – and be extended with modules covering different domains (e.g., legal and biomedical); with problems and lexical resources (e.g., WordNet-like lexical). Internal consistency and external openness make DOLCE specially suited to our needs.
The Description & Situations (D&S) is a module of the DOLCE ontology describing context elements. While modelling physical objects or events in DOLCE is quite straightforward, intuition comes to odds when we want to model non-physical objects such as social institutions, plans, organizations, regulations, roles or parameters. The representation of context is a common problem in many realistic domains from technology and society which are full of non physical objects, e.g. non-physical situations, norms, plans, beliefs, or social roles are usually represented as a set of statements and not as concepts.

D & S results to be a theory of ontological contexts because it is capable of describing various notions of context or frame of reference (non physical situations, topics, plans, beliefs, etc.) as entities. It features a philosophically concise axiomatization.

D&S introduces a new category, *Situation*, that reifies contexts, episodes, configurations, state of affairs, cases, etc. and is composed by entities of the ground ontology (e.g. a domain ontology derived from DOLCE). A Situation satisfies a *Situation Description* (Description), which is aligned as a DOLCE non-physical endurant and represents a conceptualization (as a mental object or state), hence generically dependent on some agent, and which is also social, i.e. communicable.

![Figure 4.10: Description and Situation](image)

Situation Descriptions are composed of descriptive entities, i.e., *Parameters*, *Functional Roles* and *Courses of Events*. Axioms enforce that each descriptive component links to a certain category of DOLCE (the actual objects they act upon): Parameters are valued by Regions, Functional Roles are played-by Endurants and Courses of Events sequence Perdurants.

Figure 4.10 [44] shows the distinctions between *Situation* (representing real world situations) and *Situation Description* (representing the context description), and the roles of their descriptive entities.

This captures that multiple overlapping (or alternative) contexts may match the same world or model, and that such contexts can have systematic relations among their elements.

D&S shows its practical value when applied as reference ontology for structuring application ontologies that require contextualization. As we will see in the remainder of this Chapter, this is the case when describing the e-Government service-supply scenario.

Web Service Modelling Ontology (WSMO)

One of the objectives of our work is to keep a clean separation between the semantic description of Web services and e-Government context, without affecting existing Semantic Web Services ontologies, and allowing the integration of several standards. In our approach, we adopt Web Service Modelling Ontology (WSMO) as the reference ontology for describing Web services.

In Section 2.4.2, we already introduced WSMO; in this paragraph, we outline the rationale for using it. We refer to some designing principles of WSMO that particularly embrace our approach and other standards do not have [32]:

---

### References

[32] [Link to reference]

---

This document is a draft and may contain errors. Please refer to the final version for accuracy.
4.3. Meta-Modelling the Conceptual Model

- **Strict Decoupling.** Decoupling denotes that WSMO resources are defined in isolation, meaning that each resource is specified independently without regard to possible usage or interactions with other resources. This complies with our distributed and cooperative approach.

- **Centrality of Mediation.** As a complementary design principle to strict decoupling, mediation addresses the handling of heterogeneities that naturally arise in distributed environments. Heterogeneity can occur in terms of data or process. WSMO recognizes the importance of mediation for the successful deployment of Web services by making mediation a first class component of the framework.

- **Ontological Role Separation.** User requests are formulated independently of (in a different context than) the available Web services. The underlying epistemology of WSMO differentiates between the desires of users or clients and available Web services. This complies with our multi-viewpoint approach around the concept of life event.

- **Service versus Web service.** A Web service is a computational entity which is able (by invocation) to achieve a goal. A service in contrast is the actual value provided by this invocation. Thus, WSMO does not specify services, but Web services which are actually means to buy and search services. This complies with our clear separation between the description of Web services and the context of e-Government service-supply where Web services are used.

4.3.2 OCML as a knowledge modelling tool

In our approach, we used the Operational Conceptual Modelling Language (OCML) as the knowledge modelling language implementing our conceptual model. In Section 1.9.1, we already introduced some of its features.

OCML has been primarily adopted to provide a concrete modelling support for our framework. Moreover, it provides a support for executing the definitions as well as export mechanism to other representations, including Ontolingua [40] and OWL [86] (compliance with standards).

OCML supports knowledge-level modelling specifications meeting the requirements defined in Section 4.1. A base ontology provides a basic foundation for ontology development and it includes the following modules:

- **Meta:** defining the concepts necessary to describe the OCML language, such as expressions, functional terms, rule, relation, function, assertion, etc.

- **Functions:** defining the concepts associated with function specification, such as domain, range, unary and binary-relations

- **Relations:** defining the concepts associated with relation specification, such as the universe and extension of a relation, partial and total order.

- **Sets:** defining the constructs associated with and necessary to define sets, e.g. empty, set, union, intersection, exhaustive-subclass-partition, cardinality.

- **Numbers:** defining the concepts and mathematical operations required to model mathematical calculations with numbers.

- **List:** defining the concepts necessary to represent and manipulates lists; e.g. list, atom, first, read, append.

- **Strings:** defining the concepts associated with strings.

- **Mappings:** describing the concepts necessary to specify mapping mechanisms; e.g. maps-to, meta-reference, domain-reference, and so forth.

- **Frames:** defining the concepts associated with a frame-based representation of constructs. It includes classes such as class and instance, functions like direct-instances and all-slot-values, and relations like has-one, has-at-most.
4. A Semantic-Based Framework

- **Inferences**: supporting all the inference mechanisms to define functions and relations.
- **Environment**: providing an environmental support to construct OCML models and includes special operator like `exec`, which invokes a procedure from a rule and a procedure such as output to print a message.

OCML allows a rule-based reasoning. It supports the specification of backward and forward rule. In our work we adopted the first type. A backward rule consists of a number of backward clauses. Each of them specifies a different goal-subgoal decomposition. When carrying out a proof by means of backward rules the OCML interpreter (environment) will try to prove the relevant goal by firing the clauses in the order in which these are listed in the rule definition.

Beside this, (i) there exist OCML implementations of both DOLCE, and D&S; (ii) IRS-III [29] – a well-known framework and implemented infrastructure which supports the creation of semantic web service based applications – adopts OCML to store knowledge level descriptions and models the WSMO specification as a set of related knowledge models for the WSMO top level components of Goals, Web Services and Mediators, which are meta-models in corresponding ontologies.

4.4 The Ontological Framework

In this section, we propose an ontological framework that specifies a conceptual model of a distributed e-Government service-supply scenario, where multiple actors are independent nodes describing their own knowledge, and provides mechanisms of knowledge sharing and mismatch resolution. This conceptual model introduces two clean separations:

- **Context vs SWS.** We distinguish between the description of the environment where the services are provided, used, and managed by different actors and the SWS’s allowing the automatic discovery, composition, mediation, and execution of services. The former maps the e-Government application entities and requirements, and, particularly, the aspects that cannot be captured by SWS’s such as interaction with non software agents, multiple viewpoints, distinct infrastructure to represent knowledge internally, negotiation between user and provider. It represents the inputs for the latter that completes the scenario with the technical description of computable entities that are able to achieve a goal. This separation allows to integrate SWS standards and e-Government application without affecting SWS standards.

- **Context vs Vocabulary.** We distinguish between descriptive entities – that are independent views on a scenario by different involved actors – and the actual objects they act upon - representing the vocabulary of different involved actors. This separation allows to adopt different – and in some cases already existing – vocabularies for multiple viewpoints.

As depicted in Figure 4.11, the architecture of our framework is composed by three meta-ontologies describing the above three distinct aspects:

- **Core Life Event Ontology (CLEO)** is the heart of our framework by allowing the description of the configuration and guidelines knowledge of an e-Government service-supply scenario by four classes actors: end-user, PA Domain Expert, PA Manager, and Politician. It represents the contextualization of the scenario.

- The **Service Ontology** allows the description of the service delivery knowledge. Based on the above scenario description, developers provide SWS descriptions addressing integration and interoperability issues; i.e. it completes CLEO describing Web services and their composition and mediation.

- The **Domain Ontology** defines the vocabularies used by different actors in the description of their viewpoints. It represent the vocabulary layer of the framework.
4.4. The Ontological Framework

Figure 4.11: The proposed ontological framework

These three meta-ontologies have been fused to create a sound ontological framework as follow:

- Concepts of the Service Ontology are aligned to concepts of CLEO, and axioms and rules describe inference reasonings to complete SWS descriptions.

- Concepts of the Domain Ontology are used as building blocks for the other two ontologies. Descriptive elements of CLEO and Service Ontology link to concepts of Domain Ontology by means of relations.

Other ontologies can be imported in order to extend or specialize the conceptual model. For instance, an existing ontology can be used to describe the vocabulary of an involved actor; e.g. a legislation ontology can be integrated to supply terms for describing the politician’s viewpoint.

4.4.1 Core Life Event Ontology (CLEO)

As a main result of our work, we created a core ontology named “Core Life Event Ontology (CLEO)”.

A core ontology is a very basic and minimal ontology consisting only of the minimal concepts required to understand the other concepts. It should contain a number of generic concepts and method-independent definitions and it is extremely useful for reuse purposes.

CLEO describes the e-Government service-supply knowledge structure mapping the conceptual models introduced in Section 4.2.

The objective of this ontology is to enable e-Government actors to represent the knowledge they want to describe with the scenario in their own “language” and based on concepts which are familiar to them, excluding the use of technical concepts. In particular it allows to define what kinds of concepts each participant-actor should contribute (e.g. need, offer, policy, and legislation), and which are the allowed associations-relations between these elements (e.g. influences, interactions, and plans).

Because most of the concepts of the scenario represent descriptions and situations, we adopted D&S as reference ontology. Particularly, descriptions represent independent views on the e-
Government service-supply scenario by the various actors involved and may significantly differ in the notions that are used and the granularity of the description (high-level tasks vs. detailed processes). Similarities among such views are to be found on the level of constructs used to describe these views: both of them discuss roles and attributes, parameters, and course of events that can be respectively played by, valued by, and sequences a number of objects of the Domain Ontology.

CLEO has been designed to be modular: Figure 4.11 shows the main modules composing it. Every module can be readily extended and freely reused.

The central concept of CLEO is the life event, which originates the supply of services by the PA’s. Actually, the life event is the point of contact among the actor’s viewpoints. It can be described in terms of state of affairs – containing descriptive elements such as involved actors, resources, attributes and parameters –, plan – describing processes and arranging elements of a description –, and interaction – describing interaction between different viewpoints – descriptions.

The various views naturally focus on different aspects of a life event: the user viewpoint includes the description of his/her needs; the provider viewpoint the description of offers; the manager viewpoint the policies influencing the service implementations; the politician viewpoint the description of laws ruling the scenario.

The user and provider viewpoints represent the configuration knowledge of the scenario. They contain functional and non-functional descriptions for defining needs and offers, and especially use the interaction module to specify the kind of accomplishment (communication, inquiry, notification, or transaction) and describe user-provider and provider-provider (task delegation) interactions. This knowledge layer is mapped to the Service Ontology.

The manager and politician viewpoints represent the guidelines knowledge. This knowledge layer allows to describe the evolution of the scenario; changes at this level should be properly propagated to the other ontologies (concepts) of the framework (scenario description). For instance, a change in a law or a policy can have repercussions onto the manager’s policies and the description of the service process.

Finally, CLEO modules introduce a knowledge elicitation methodology that first helps domain experts to create a full description of a specific e-government context using models close to their experience and specific languages of the different involved domains, and then drives the application developers to implement SWS descriptions inferring knowledge from the context description.

4.4.2 Domain Ontology

The Domain Ontology encodes concepts of the PA domain: organizational, legal, economic, business, information technology and end-user. They are the building blocks for the definition of CLEO and Service Ontology concepts. However, our aim was not to cover all the aspects connected with the e-Government.

Distinct PA’s could use the same concepts differently or vice-versa adopt distinct terms for the same concept; a single Public Administration (PA) may not share the same point of view and have different interoperability needs by other PA’s. Multiple actors can use different vocabularies, also within the same organization.

The standardization can help, but it does not necessarily unify the aims and languages of all the involved actors. It is important that every PA (or actor) keeps its autonomy in the description of its own domain (or viewpoint); as we shall describe in the following, this does not affect our ultimate goals of interoperability and integration.

For these reasons, we designed a structure that resides on two levels of abstraction:

- The generic level represents the meta-ontology describing commonly accepted and standardized concepts and properties of the e-Government domain (e.g. Person, Address, Organization, etc.). This vocabulary defines a common standard for the content of all PA’s at a high level (e.g. national level). Such governmental content is created only once.

Because e-Government concepts range over a wide area, we need to refer to an existing upper ontology. To keep coherence with CLEO, we adopted DOLCE as reference ontology. We extended it ending some of its concepts.
4.4. The Ontological Framework

- The specific level extends the generic level within the specific domain of an involved actor or PA, ending and adapting the existing concepts and relations. This vocabulary provides for the creation of content related to a viewpoint provided by a specific organization. It can be considered as a specialization of the generic level vocabulary, because the values of some elements of this vocabulary depend on the organization, which provides a viewpoint description.

Actually, each instance ontology describes the vocabulary of the respective actor or organization. Every actor or organization is responsible of its ontology.

Existing ontologies can be introduced and used to define a viewpoint vocabulary or to specify scenario-dependent aspects not covered by the generic level. These ontologies can be added to the ontological framework or directly aligned to CLEO.

4.4.3 Service Ontology

The Service Ontology contains the SWS descriptions. In our approach, this ontology plays a double role: (i) completes the framework with the description of e-Government processes and services that are implemented using and integrating Web services; (ii) solves mismatch problems introducing mapping mechanisms between two existing CLEO viewpoints that interact to accomplish a task (user-provider and provider-provider interactions).

The first aspect regards the mapping of computable entities defined in CLEO onto SWSs technical concepts. Concepts such as precondition, postcondition, grounding, orchestration and choreography of Web services are defined at this level by developers. Axioms and rules have been introduced to describe inference reasonings from CLEO descriptions. This aspect addresses the service integration issues.

The second aspect is associated with the domain independence of actor’s viewpoints. In other words, a case of mismatch between two domain models is bridged by defining appropriate mapping mechanisms. The developers play the role of a third party for determining and solving the differences between heterogeneous languages and protocols. This aspect addresses the service interoperability issues.

As introduced in Section 4.3.1, we adopt WSMO as reference ontology for SWSs. Following its definitions, the structure of the Service Ontology comprises three main modules (Figure 4.11):

- Goal Ontology, representing the goals that a user would like to achieve. Goal descriptions represent the computational entities (invocations) associated with CLEO Need descriptions.

- Mediator Ontology, representing all mediators linking elements of Goal and Web Service Ontology and addressing the interoperability issues at protocol, process, and data level.

- Web Service Ontology, representing the capabilities and interfaces of a Web service, and the composition of several Web services. Web service descriptions represent the computational entities (execution) associated with CLEO Offer descriptions.

Further standards may be introduced creating other Service Ontologies redefining the mapping of CLEO concepts.

4.4.4 Ontology Stack

The ontology stack in Figure 4.12 summarizes our mapping effort and represents the inclusion graph of considered ontologies (i.e. a layer includes the upper layers). We use OCML as Knowledge Modelling tool (Section 4.3.2), and its associated Base Ontology as foundation of our development. We adopt DOLCE – extended by its module Descriptions & Situations – as upper ontology, and WSMO as meta-ontology for SWS’s (Section 4.3.1). Based on such a layer, we create our ontological framework (Section 4.4) that is composed by the following ontologies: CLEO, Domain Ontology, and Service Ontology. The three ontologies are used to align concrete Application Ontologies. All
of the descriptions of application ontologies are shared, in such a way that an actor can refer to other ontologies to describe its viewpoint. For instance a developer can compose a web service with existing ones provided by other organizations or link it to existing goals provided by other organizations.

![Ontology Stack](image)

Figure 4.12: Ontology Stack

Our mapping method was a combination of a bottom-up and a top-down approach. On the one hand, ontologies in the lower layers – representing worked out case studies – provided representation requirements for the higher layers, which abstracted their concepts and relationships. On the other hand, the upper layers provided design guidelines to the lower layers.
Formalizing the Ontological Framework

In this chapter, we carry on the description of our work providing the formalization of the meta-ontologies that compose the ontological framework introduced in Chapter 4: Core Life Event Ontology (CLEO, Section 5.1), Domain Ontology (Section 5.2), and Service Ontology (Section 5.3).

Starting from the conceptual models (Section 4.2) and requirements (Section 4.1), we detail the internal structures of the above ontologies, highlighting their modeling and reasoning capabilities. Moreover, we specify the main concepts and relations, and outline the constraints and axioms that bound their definitions. As implementation examples, we report only some OCML (Section 4.3.2) definitions of the introduced elements.

Because CLEO is the core – and the bigger among the three ontologies – of our ontological framework, we dedicate more space to its description. In particular, we will use UML diagrams for offering some views on specific aspects of its modules.

For the Domain Ontology, we provide the description of its particular two-level structure and the building process that led to such a structure, while, for the Service Ontology, we describe its links to CLEO that allow to infer useful context knowledge for the SWS descriptions.

5.1 The Core Life Event Ontology (CLEO)

As introduced in Section 4.4.1, the Core Life Event Ontology (CLEO) provides the minimal concepts and relations required to describe aspects connected to e-Government service-supply scenarios. It is a means for representing knowledge by multiple viewpoints and defining the context in which an application should be developed. The knowledge may include generic service processes as well as situation descriptions, involving human and not human agents.

CLEO is a big ontology: it contains 242 elements among classes, relations, axioms, and rules that compose its 6 modules. Moreover, it refers to specific modules (e.g. Plan Ontology) provided by the D&S ontology (Section 4.3.1), and the Base Ontology of the OCML environment (Section 4.3.2). We cannot detail all of the involved aspects; in some cases, we simply cite or outline them.

CLEO is composed by the following extendible and reusable modules; each of them deals with one particular aspect of our conceptual model:

- **Life Event Description Module**: it is the heart of CLEO, representing the life events and all actor’s viewpoints that provide a description of them. It adopts the modules below to provide a sound description of the context around the life event.

- **State of Affair Description Module**: it gathers the elements (roles, attributes, and parameters) that are relevant in the description of a life event situation.

- **Conception Description Module**: it represents the conceptions that actors may describe in their viewpoint related to a specific state of affair and life event: needs, offers, policies, and legislation.
• **Quality Description**: it details non-functional parameters (e.g. quality of services, security levels, trust policies, etc.) associated with conception descriptions.

• **Plan Description Module**: it allows to describe the tasks associated with the descriptions of a scenario and organizing them into a plan within an actor’s viewpoint.

• **Interaction Description Module**: it plays a double role, representing the interactions between two distinct viewpoints, and capturing possible mismatches in the description of exchanged resources. It is the unique module shared between two distinct viewpoints.

Note that, we can distinguish two classes of modules: **static** – i.e. state of affair, conception, and quality descriptions – and **dynamic** – plan and interaction descriptions. The former gathers and represents concepts, roles, features, and relationships of an actor’s viewpoint; the latter represents the business processes of an actor’s viewpoint, in the two forms we identified in the previous chapter (Section 4.2).

In the following, we show an overall view of the structure of CLEO as a multi-viewpoint tool for representing e-Government service-supply scenarios. It provides a more detailed visualization of CLEO and its modules than the one in Figure 4.11, Section 4.4. Such a description is also used as directory for the rest of the section.

![Figure 5.1: Multi-viewpoint structure of CLEO](image)

Figure 5.1 shows the main elements of CLEO modules rearranged within four viewpoints: user (or service worker), domain expert of a PA (provider), manager, and politician. Each viewpoint is a loose-coupled module and, hence, it can be independently described.
The core concept is the life-event that can be associated with the four viewpoints represented by balloons. Each viewpoint is characterized by the e-government-actor that defines it, the ontology representing the actor’s vocabulary (vocabulary layer), the particular conception that the actor expresses, and other specific elements derived from the following modules: state of affair, plan, and interaction descriptions. The conception description is the core of each viewpoint linking together all of the elements. The red boxes expand their descriptions in term of composing elements. In the cases of user and provider viewpoints, the need and offer descriptions represent the interfaces of the viewpoint towards other viewpoints. The mechanism of need/goal and offer/service decomposition allows to model knowledge at different level of granularity, fitting project-specific requirements, and represent complex interactions that cannot be represented by the current one-shot SWS approaches. Note the role of the interaction module: it allows the user and provider – but also provider and provider – viewpoints “talking together”.

In the rest of the section, we show some specific views of the CLEO modules. We start from the Life Event Description module that allows to describe the viewpoints. Then, we consider the modules that supply static and dynamic descriptions of the life event. Finally, because it is an important aspect of the CLEO approach, we dedicate a paragraph for deepening the aspects connected to the interactions between viewpoints.

Another important aspect depicted by Figure 5.1 is the clean separation existing between the guidelines (Politician and Manager viewpoints) and the configuration (User and Provider viewpoints) knowledge levels.

The former drives the evolution of the scenario. A change in an element of the guidelines knowledge level may produce changes in the other elements of the scenario. Changes can be propagated following the chain created by the influences relations that links the CLEO elements. In our work, we do not detail this aspect. In fact, this is an interesting and complex further research field. A previous work [84] demonstrates that representing legislation knowledge is not an easy task and many difficulties were encountered. However, we create the basis for its development. Future work may regard extending the knowledge structure of the Politician and Manager viewpoints, and creating the appropriate infrastructure for tracing the changes and suggesting the evolution of the configuration knowledge.

In this thesis, we focus on the configuration knowledge level that defines the contexts for the service delivery knowledge level, i.e. the SWS descriptions. Note that the Service Ontology is associated with the need and offer descriptions.

5.1.1 Life Event Description Module

Our approach is based on the life event metaphor, which prompts the supply of services by PA’s. Referring to the Life Event Metaphor Conceptual Model and e-Government System Conceptual Model, we may simply consider how many different views may exist on a life event: the citizen, the PA, the manager, the politician, etc. Note that the concepts used to formulate any given view are clearly separate from the actual objects they act upon and often independent from the concepts appearing in other views.

Life-Event and Life-Event-Description are center concepts of CLEO, respectively referring to the Situation and Description concepts of the D&S ontology (Figure 5.2).

A Life-Event is a situation of the real world setting for at least one event that occurs in that situation. Several taxonomies and standards of life events may be described and connected. For instance, the Switzerland has two national life events standards. A life event of one standard may be linked to the corresponding one (having the same semantic) of the other standard by means of the symmetric relation has-variant-name. In this way, description provided for a life event may be valid also for the linked one.

---

1Note that more than one conception description may be provided within a viewpoint
2Figure 5.1 shows only the links between the main elements
A life event satisfies one or more Life-Event-Descriptions. The latter is mainly defined by the following elements: E-Government-Actor, State-of-Affair-Description, Conception-Description, and Plan-Description.

Because several state of affairs, conceptions, plans can coexist within the same life event description (e.g. two final state of affairs that can be reached through two distinct plans) or reused in other one, we introduced the following two relations for linking the above elements:

- cleo-conceives, specifying which conception-description the e-government-actor conceives in the initial-state-of-affair-description of a life-event-description; it provides a static description of the life event, because it depicts a situation.

- cleo-achieves, specifying which final-state-of-affair-description of a life-event-description the e-government-actor achieves following a plan-description; it provides a dynamic description of the life event description, because it depicts a plan to follow.

Currently, we consider the following four kinds (sub-classes) of life event description, but further views may be added in the future when extensions needs arise: User-Life-Event-Description, Provider-Life-Event-Description, Manager-Life-Event-Description, Politician-Life-Event-Description.

As introduced in the Life Event Metaphor Conceptual Model (Section 4.2), more than one actor belonging to the same class may provide a distinct viewpoint. e.g. two provider-life-event-descriptions by two distinct organizations that cooperate or compete for offering services.
5.1.2 Static Descriptions

State of Affair Description Module

The aim of this module is to depict a representation of a state of affair. A state of affair description provides a particular picture of the life event (or generally a situation). While the life event defines an event, and so it also represents dynamic aspects (e.g. a plan of the event), the state of affair is a “state”, i.e. a static representation. More than one state of affairs can be identified within a life event: e.g. the initial and the final state of affairs.

A state of affair description allows to select the elements of the Domain Ontology that are useful to represent the situation, and specify their role within the situation. Note that a state of affair description is an internal description of a viewpoint, thus it can only select terms of the actor’s vocabulary (no cross viewpoints relations address more modularity). The entities selecting elements of the Domain Ontology may be defined in this module and reused in other ones within the same life event description. From this point of view, the state of affairs play the role of containers that capture some of the concepts – not all of concepts are captured by state of affair descriptions – necessary to describe the actor’s viewpoint and interfacing between the context description of CLEO and the vocabulary layer of the Domain Ontology. The state of affair description is completed introducing the relations among the selected elements. In this way, we can have a complete idea of the situation. State of affair descriptions are really relevant from the user point a view. In fact, users can recognize the depicted situation, and ask for the connected life event and services.

As shown in Figure 5.3, the State-of-Affair class describes a situation connected to a life event, which is setting for all entities involved in that situation. The associated State-of-Affair-Description links to the concepts representing the descriptive counterparts of the entities of the Domain Ontology:

- cleo-role: describing a role played by a thing entity of the Domain Ontology, such as agent, person, object, ...
- cleo-attribute: selecting one of the properties (slots) of a concept of the Domain Ontology and introduced by means of a cleo-role.
- cleo-parameter: defining a constraint bounding the state of affair.

On the base of used descriptive entities, we distinguish some sub-classes of the State-of-Affair-Description. Significant examples are the following:

- **Budget**: defining only resources;
- **Quality**: defining only non-functional-parameters (i.e. quality of service, security, etc.);
- **Service-Available**: defining a situation where a provider offers services;
5. Formalizing the Ontological Framework

Figure 5.4: Top-level taxonomy for the descriptive entities and the main relations between them.

- **Service-Request**: defining a situation where an applicant requires services;
- **Service-Delivery**: defining a situation where a customer is receiving a service;
- **Processed**: defining a situation where one or more activities have been executed.

The relations between descriptive entities are expressed by axioms (class `constraint-expression`). We defined a set of relations that can be used as elements of axioms. Figure 5.4 shows only the main relations.

**Conception Description module**

The *Government Service-Supply Scenario Conceptual Model* and *Life Event Metaphor Conceptual Model* – defined in Section 4.2 – show which concepts each involved actor expresses within the same context and life event. Moreover, the *Government Service-Supply Scenario Conceptual Model* and the *e-Government System Conceptual Model* define the existing influences between the considered concepts. Following these ideas, we created the *Conception Description* module, that represents four kinds of actor’s conception – *legislation*, *policy*, *need*, and *offer* (Figure 5.5), and the relations among them and their elements.

Figure 5.5: The Conception Description module. Gray boxes represent elements of other CLEO modules.

Note that each conception is connected to a specific actor’s viewpoint (sub-class of life event description). In the following, we briefly describe the structure of each conception.
5.1. The Core Life Event Ontology (CLEO)

Legislation Description

The legislation-description is a set of norm descriptions that regulate a life event. The concept norm-description is the descriptive counterpart of a law, directive, or regulation that may be defined in a specialized ontology (e.g. Legal Ontology) representing the vocabulary layer of the politician.

A legislation-description is influenced by a need-description and influences the policy-description and offer-description conceptions connected to a life event and provided by managers and organizations, respectively. Differently, a norm-description can straight influence elements of the above conceptions: e.g. strategy-description and service-description.

Policy Description

The policy-description describes the main high-level purposes and principles connected to a specific life event and provided by public administration managers. It uses at least one strategy-description to plan the steps for achieving a particular subset of purposes. The strategy-activity-description concept represents a single activity of a strategy.

Policies and strategies define required and shared elements for the scenario: common practice to follow, cooperations, agreements, etc. All of these elements are influenced by legislations and norms and influences the involved service providers.

Both policies and strategies can be arranged by a plan, and thus define task descriptions that refer to activities defined in the Domain Ontology. The latter may be extended with existing managerial ontologies specifying strategies and activities features in order to create an adequate vocabulary layer.

Need Description

The need-description concept expresses what a user would like to achieve within a specific state of affair of a life event. It defines the activity of expressing a need by an actor (class need-task). To represent a need, we identify the following elements:

- interaction-description: representing the kind of interaction, the mutual actions, and the exchanged resources associated with a need. More than one interaction can be linked, mapping the situation where different protocols – provided by different organizations – can be followed to accomplish a need.

- goal-description: representing a step that a user should achieve to satisfy his/her need; on the base of the kind of interaction one or more goals can be associated with a need. Each goal should includes functional and non functional requirements connected to the achievement of the goal, and represents the invocation for one of the actions expressed in the interaction description.

- goal-plan-description: representing the workflow (business process) of goals that the user should invoke to accomplish the need. Associated with the interaction description, It allows to represent complex negotiations between users (non software agents) and providers.

- quality-description: representing general non-functional requirements in common with all of the goals of the need description.

The following example 5.1.2 reports the definition of the goal-description class. A goal differs from a need because it represents the request for a service and can be invoked. The slot goal-executes links the goal to the transition of an interaction-description that is invoked by the goal.

---

3We detail this aspect later in the Plan Description Module.
Formalizing the Ontological Framework

Definition 5.1: The goal-description class

(def-class goal-description (egov-description)
  ((defines :type goal-task :cardinality 1)
   (cleo-influences :type norm-description :min-cardinality 0)
   (goal-executes :type transition-event :min-cardinality 1)
   (uses-quality :cardinality 1 :type quality-description)
   (used-by :type need-description :min-cardinality 1)))

The goal class is an important element of the ontology: it represents a gate to the SWS layer represented by the Service Ontology.

Note that we do not define slots and relations connected to functional descriptions. We simply created the links to other elements of CLEO. In this way, we can abstract from a specific SWS approach. Future work can regard the alignment of other SWS approaches to our proposal. For instance, it would be useful simply extend the goal class – creating an inherited class – with the specific features connected to the request of service provided by the chosen SWS approach.

Offer Description

The class offer-description describes what a provider (private or public organization) supplies within a specific state of affair of a life event in order to match the user needs. It defines the activity of presenting an offer by an actor (class offer-task). The offer description has a specular structure and is strictly connected to the need-description concept. To represent an offer, we identify the following elements:

- **interaction-description**: representing the kind of interaction, the mutual actions, and the exchanged resources associated with an offer. It represents the way to consume the services provided with the offer. Multiple interaction-description can be linked, mapping the situation of organizations offering different way (e.g. multiple contracts, protocols, etc.) of consuming an offer. Connected need and offer descriptions (i.e. the offer matches the need) share the same interaction-description.

- **service-description**: representing a service that can fulfill the offer; more than one services can be defined within the same offer, mapping the situation where an organizations can provide a package of different services accomplishing the same task but matching different classes of users, economic conditions, etc. Each service defines the functionalities provided by an offer, and represents the execution for one of the actions expressed in the interaction descriptions.

- **service-plan-description**: representing the way to organize services within the offer; it is not a proper workflow of services, but defines a container for gathering and organizing – for instance scheduling – services that share the same aim, qualities, etc.

- **quality-description**: representing general non-functional requirements in common with all of the services of the offer description.

A complex-services-description concept is a service-description that allows to represent the eventual decomposition of a service into sub-services by means of a plan description. The latter defines the business process of the service. Sub-services may be known a-priori – in this case we can speak of composition of services – or their functionalities may be delegated to not known external services by means of a need description – in this case we can speak of integration of services. Actually, a service may be decomposed in terms of service-description or need-description concepts.

The following example 5.1.2 reports the definition of the service-description and complex-service-description classes. A service differs from an offer because it represents the provision of a service (e.g. Web service) and it can be executed. The slot service-executes links the service to the transitions of an interaction-description that are executed by the service.
5.1. The Core Life Event Ontology (CLEO)

Definition 5.2: The service-description and complex-service-description classes

(def-class service-description (egov-description) ?sd
 (defines :type service-task :cardinality 1)
 (used-by :type offer-description :min-cardinality 1)
 (service-executes :type transition-event :min-cardinality 1)
 (uses-quality :cardinality 1 :type quality-description)
 (cleo-influenced-by :type (or (norm-description)
                                (strategy-description))
                    :min-cardinality 1)))

(def-class complex-service-description (service-description) ?sd
 (uses-service :type service-description :min-cardinality 1)
 (uses-need :type need-description :min-cardinality 1)
 (uses-plan :type decomposition-plan-description :cardinality 1)))

The afore defined classes represent the second gate to the SWS layer (Service Ontology). Following the same reasoning we did for the goal-description concept, we do not define slots and relations connected to functional description and we simply created links to other elements of CLEO.

Finally, note that the decomposition of services is a peculiarity of SWS descriptions (Section 5.3). Our aim was not to substitute them but introduce two different levels of representation: (i) the decomposition is not described and demanded to the Service Ontology or (ii) the decomposition is represented at this stage and later mapped to the SWS descriptions. Different factor should be considered in the choice of the adequate level of representation: e.g. the specific SWS approach, the skills of the developers and domain experts, etc.

Quality Description

The quality-description concept has been introduced to gather all of the non-functional properties that bound a particular description.

It is used within the offer, service, need, and goal descriptions. The description defined at the goal and service level inherits non-functional properties defined at the need and offer level, respectively.

5.1.3 Dynamic Descriptions

Plan Description module

This module describes plans that enable to organise task descriptions in order to represent business processes associated with life event and conception description modules. We refer to the Plan Conceptual Model defined in Section 4.2.

In our approach, we take advantage of a number of concepts from the Ontology of Plan, which is a module of D&S ontology. It allows the division of tasks into elementary and control and the construction of complex tasks from elementary ones among other features. In other words, we can describe both simple (e.g. workflow) and complex (e.g. scheduling) plans adapting to the needs and skills of the different actors: users, manager, organizations, etc. However, further specific approaches used by involved organizations may be adopted simply extending this module.

We distinguish two main elements: the cleo-plan-description and the cleo-task. The former represents a proper plan; all of the plans defined in CLEO inherit from it. The latter represents a generic element (activity) of the plan; all of the tasks defined in CLEO inherit from it. Note that a cleo-task is a descriptive entity that sequences events (egov-event) of the Domain Ontology.

Figure 5.6 depicts the elements of the module and their links to the concepts of the Ontology of Plan. To better describe the structure of a plan, we adopt, as example, the concepts of the goals-plan-description used by need-description to represent the workflow of goal-description concepts.
88 5. Formalizing the Ontological Framework

Figure 5.6: The Plan Description module and the goals-plan-description example. Top-right gray boxes represent concepts of the D&S ontology. Mid-left gray boxes represent elements of the Conception Description Module for the goals-description-plan example. The three bottom classes aggregate control and description constructs of a plan.

Figure 5.7: The goals-plan-description example.

Figure 5.7 shows an example of goal-plan-description that uses a sequential-task (complex task) composed by a list of goal-task that have to be sequentially performed. Each goal-task represents the activity of expressing a goal, and it is defined by a goal-description.

Links between the tasks are represented by means of the following couples of D&S relations: predecessor and successor, and direct-predecessor and direct-successor. The latter couple is used to represent the actual flow, when complex and control task are introduced.

The model we adopt for representing plans is based on the constraints that some CLEO description defines a task, and then the plans organise such tasks. However, the result is not easy to visualize, especially because we expect a plan where the descriptions and control constructs are the elements. For this reason, we create the description-construct and the control-construct concepts. The former aggregates all of the descriptions of CLEO, the latter the complex and control tasks of the D&S ontology. Two new relations link these constructs: has-next and has-previous. Using rules, it is possible to infer the links between such concepts starting from a plan description. Figure 5.8 shows the transformation of the goals-plan-description example of Figure 5.7.
5.1. The Core Life Event Ontology (CLEO)

Figure 5.8: The goals-plan-description example: the highlighted arrows shows the new flow of the plan.

**Interaction Description Module**

The Interaction Description module represents an agreement between user and provider – or provider and provider – viewpoints about how to consume services and exchange resources. It is the unique point of contact: a shared module that represents knowledge crossing multiple viewpoints. It allows to capture context elements and requirements that cannot be caught in other CLEO modules, but also check the existing ones. Because the heterogeneity of different viewpoints, it is important to capture and match the eventual different descriptions. We based our approach on the Interaction Conceptual Model defined in Section 4.2.

Figure 5.9: The Interaction Description module.

Figure 5.9 shows the module. We distinguish two types of interaction descriptions: communication-description – i.e. notification, application, and enquiry – and transaction-description. They respectively map the two-way and full transaction interaction levels of e-Government services, and differ on the number of transition events that can define: one for the communication and at least two for the transaction description.

**Transition Event description**

The core of the Interaction Description module is the transition-event. It gathers the elements that allow to represent the sequence of actions, exchanged resources, and eventual data and process
differences between two viewpoints.

- The **transition-medium** concept describes the mean used to transfer the resource: e.g. web, front-desk, letter, mail, e-mail, publication, etc. Note that in some cases (e.g. book purchasing) the transaction can be completely on-line, but the actual value is transferred by means of different medium (e.g. mail delivery).

- The **transition-condition** description defines the necessary elements and the conditions for activating the transition event.
  The first aspect to consider is the **triggering condition**: if the triggering condition is explicit, the transition needs an invocation and the valuation of the activation condition, otherwise it needs only the valuation of the condition. We generally consider explicit transitions.
  The condition is composed by the following three elements: **state-condition**, defining conditions on the state (information space) on which the transition operates; **agent-condition**, defining conditions on the profile of the resources performing the transition; **time-condition**, defining temporal constraints for the invocation and execution of the transition.
  As depicted in Figure 5.9, the above concepts have similar definitions: they refer to descriptive entities, and use an axiom (class constraint-expression) for representing relations between entities. Each descriptive entity selects concepts of the vocabulary layer that are involved in the transition:
    - **state-element**: concept necessary for achieving the transition; e.g. information, resources, acknowledgment message, etc. It differs from the transition resource that only represents the transferred value. To quote an example, in an enquiry communication, the transferred value may be the result of a database query, while the state of the transition may include the research parameters provided by the requester.
    - **agent-element**: the transition is an event and, therefore, it can be associated with a resource that performs it (Section 5.1.3); e.g. employee, hardware, software, company, etc.
    - **time-element**: temporal parameter for representing constraints of the transition; e.g. transition executable only during office times.

- The **transition-resource** description defines the subject of the transition.
  A transition transfers a resource from a **source** to a **destination** that represent two distinct **transition-agent**. We distinguish two types of transition agents: **inside-agent** and **outside-agent**. The former is the view from the interaction requester’s own perspective, the latter is the complementary to the inside-agent. Note that inside-agent does not automatically correspond to the source, and outside-agent to the destination: in a full transaction description the inside-agent (outside-agent) may be the source (destination) of a transition, and the destination (source) of another transition; beside this, in an enquiry communication, the inside-agent is the destination and the outside-agent the source of the unique transition.
  The value of the transition is represented by one (or more) **resource-element** descriptive entity. The latter may represent concepts from a simple real (e.g. a book) or abstract (e.g. a care) object to common exchange formats such as forms. Its definition is similar to the one adopted for the **state-element**.
  In the economical transaction model defined by Polovina [80] (Section 4.2), the exchange of values should be balanced. The model is accordingly incomplete until both sides of a transaction “balance”, and this has been shown to lucidly represent qualitative transaction. The balance check immediately raises the developer’s awareness of the need to discover the appropriate knowledge for the model that have not been captured in other modules. In our approach, we refer to Polovina’s model; however, differently from the e-Commerce, in

---

4This slightly differs from the Polovina’s definition [80] introduced in Section 4.2
5.1. The Core Life Event Ontology (CLEO)

the e-Government field not all of the interactions are economical and balanced transactions. We generalized such a model simply keeping the “balance-checking” rigour upon the model, without forcing it as a strict constraint.

Because the interaction description is a shared module between two independent and heterogeneous viewpoints, we may expect at least two distinct counterparts of the introduced descriptive entities: one (or more) from the domain ontology of the source viewpoint and one (or more) from the domain ontology of the destination viewpoint. As example, Figure 5.10 depicts the state condition description; the same role (state element) is played by two distinct terms in distinct vocabulary layers. This also explains one of the motivations of splitting the context description from the vocabulary description: an entity defined at the vocabulary layer can play two distinct roles in two distinct context: descriptive entity for the state of affair description and state element for the state condition.

Figure 5.10: The state of a transition (gray boxes) as a shared definition.

This simple mechanism allows to represent data and process mismatches between the shared elements of the two viewpoints. For instance, we may consider a state element ID (identification): if it selects the concept name from the source vocabulary layer, and the concept ID-name from the destination vocabulary layer, then we have a data mismatch; if it selects the name concept from the source vocabulary layer and the first-name and last-name concepts from the destination vocabulary layer, then we have a particular type of process mismatch\(^5\). The aim of this module is not to solve existing mismatch problems, but to describe a context where two different domain meets. Mismatch resolution is delegated at the service execution time and, hence, represented at the service delivery level (Service Ontology).

The definitions of state, resource, and transition-agent elements introduce constraints in order to limit the concepts that can play such roles. For instance, if an entity plays a state element role in a state condition, then should exist at least one state of affair for each involved party where this entity plays a role. Figure 5.10 clarify this aspect.

Testing the constraints, it is possible to (i) check the quality (completeness) of the knowledge that have been identified in the associated viewpoint descriptions, and (ii) obtain a list of entities that may play a state condition role of a transition. In figure 5.11, dashed arrows show the backward reasoning we may adopt for checking or obtaining the entities of the viewpoint that can be used as resource element (or state element) description. Note the importance of the chain of uses relations that we adopt for specifying the descriptions of a context. Following back the chain (e.g. by means of a backward rule), we can obtain all of the descriptive elements of the state of affairs connected to the resource element. Note that the conception description is the point of contact between the interaction description and the life event description.

\(^5\)The destination processes expect knowledge in a different way of the one provided by source processes
To restrict the number of possible elements, we refer to the relations that link the descriptions. For instance, to obtain only the elements of the viewpoint A (i.e. going left in the schema of Figure 5.11), we can check if the life-event-description or state-of-affair-description of the viewpoint A uses the agent that is the source of the transition. Other useful relations refining the research are the cleo-achieves and cleo-conceives. They allow to select only the state of affair descriptions and life event descriptions connected to a specific conception description.

Figure 5.11: The transition resource description. Dashed arrows represent the backward inference path for obtaining elements of the viewpoint A, starting from the resource element.

Axioms and rules represent the above reasoning.

**Modeling interactions between viewpoints**

Figure 5.12 depicts some situations requiring the use of the Interaction Description module within a generic multi-viewpoint scenario.

- (a) The user (U1) follows an interaction description (A or B) to consume a service supplied by a provider (P1). The two interaction description may be provided in order to model the following situations: P1 proposes two different offers to U1; P1 proposes only one offer
but two distinct ways (different contracts, different prices, etc.) to consume it; U1 requires different interactions connected to two distinct states of affair.

- (b) In the same situation, a new user (U2) benefits of the services of P1. U2 does not share the domain with U1 (i.e. they are independent): a new interaction description is needed. F and A may have the same structure (list of actions), but they may differ on the terms used to define the involved resources.

- (c) To accomplish the offered services, P1 delegates two functionalities to two different providers: P2 and P3. Because the functionalities are different, two distinct interaction descriptions (C and D) are introduced to manage the respective service delegations.

- (d) Both P2 and P3 benefit of a service supplied by P4 adopting the interaction description E. In this case they can use the same interaction description because they share the same domain.

The Interaction Description module is tightly connected to the need and offer descriptions of the involved viewpoints. Section 5.1.2 introduces how:

- need-description and offer-description link to a interaction-description;
- goal-description and service description link to a transition-event.

Figure 5.13 shows an example of transaction that highlights the above connections. The transition events allow to: (i) link the goals of a need description to the services of an offer description; (ii) define the elements – i.e. the state and resource elements – that represent the inputs and outputs of a viewpoint within a transaction. As the goals are the invocation of the transition events, the need-goals decomposition drives the transaction.

Figure 5.13: Links between Need and Offer Descriptions through the Interaction Description (gray boxes).

However, Figure 5.13 shows only a simple case. The proposed structure introduces a high level of flexibility and different levels of granularity. In the following, we describe some generic cases that give an idea of the situations that can be modelled (Figure 5.14). They may represent the detailed descriptions of the interactions individuated in Figure 5.12.

- (a) The transition event T is invoked by one goal G and may be executed by two services S1 and S2. Which service will be activated depends on the functional and non functional parameter descriptions of the goals and services. S1 and S2 share the same domain, hence, they do not need two distinct mappings of the exchanged resource R. The latter is played by a single entity (one arrow) in the viewpoint of G and two entities (two arrows) in the viewpoint of S1 and S2. This represents one example of process mismatch: the service viewpoint expects two entities, while the goal viewpoint provides only one.
(b) This case presents a similar situation of (a), but S1 and S2 do not share the same domain. Two transition events (T1 and T2) have been introduced for mapping the two cases. In T1, there is not a process level but a data level mismatch: both goal and service viewpoints use two entities for describing R, but they use different concepts. In T2, we have a problem at the process level as in the case (a), but we have two entities from the goal viewpoint and one from the service one’s. C1 and C2 conditions complete the description of the transitions.

(c) This is a simple case with only one goal and service. The transition event performs a transfer of a transition resource having two resource elements: R1 and R2. The former is represents a data mismatch, while the latter is not affected by mismatch problems.

(d) In this case goal and service are linked by two transition events. It models two ways of consuming the same service. It reuse the transition events defined in the case (b). The transition conditions C1 and C2 may trigger the proper transition that has to be activated.

(e) In this case we have two goals that require two distinct functionalities provided by the same service. This case can be compared with the case (c), where the two functionalities are joined into a unique transition with two resource elements. In this case, the same service has two distinct processes that exec the two required functionalities; in the case (c), there exists only one process that execs both the functionalities. This case represents another kind of process mismatch: two distinct requests from the source viewpoints are mapped to a unique service of the destination viewpoint.
5.2 The Domain Ontology

In our ontological framework, the Domain Ontology provides the building blocks for defining CLEO and Service Ontology concepts. As mentioned in the previous section, it represents the vocabulary layer of multiple viewpoints. In order to keep autonomy in the description of multiple viewpoints, we can not create a unique Domain Ontology holding all the possible concepts.

As introduced in Section 4.4.2 and according to the results provided by the eGov Projects 3.1.1, we designed a structure that resides on two level of abstraction: generic and specific level. Figure 5.15 shows four distinct ontologies (A, B, C, and D) derived from the generic level that are independent each other. They compose the specific level. Each of them describes a particular domain connected to a viewpoint; e.g. legislative terminology, technical terminology of an organization or a field, actor's language, manager's tasks, etc.

![Figure 5.15: The two-levels structure of the Domain Ontology.](image)

It is important to note that the aim of such a structure is to represent an heterogeneous scenario, and not create mappings between different concepts or solve existing mismatch problems. In our approach these tasks are delegated to the service delivery knowledge level; i.e. at service execution time, when the need of solving a mismatch arises.

In the following, we detail the building process we adopted and the structure of the generic level of the Domain Ontology.

5.2.1 Building the Domain Ontology

Because the e-Government concepts range over a wide area, we need to refer to an existing upper ontology. To keep coherence with CLEO, we adopted DOLCE (Section 4.3.1) as foundational ontology.

However, we cannot strictly use DOLCE as generic level of our Domain Ontology. In fact, it contains a lot of concepts that are not relevant in the e-Government field or too detailed for our general purpose aims. Contrariwise, some of the common accepted concepts of the e-Government field are not defined or not completely specified.

For these reasons, we created a three-stage process to fix the existing epistemological gap between DOLCE and the desired generic level. Figure 5.16 depicts such a process. The process consists of the following stages:

1. selecting the concepts of DOLCE that are relevant for our generic level\(^6\);

2. creating a DOLCE e-government mapping ontology that inherits from DOLCE only the concepts selected in the previous stage and their taxonomy structure;

3. using the DOLCE e-government mapping ontology as starting point for creating the generic level of the Domain Ontology.

\(^6\)We also selected the concepts of the D&S module of CLEO; in this way the DOLCE e-government mapping ontology has been used as starting point for the creation of CLEO.
Figure 5.16: The three-stage process for creating the generic level of the Domain Ontology. We distinguish four ontologies: DOLCE (and its module D&S) represented by the gray ovals; the DOLCE e-government mapping ontology represented by white ovals; CLEO and Domain Ontology, extending the previous ontology with bold ovals.

The core of the above process is the creation of the DOLCE e-government mapping ontology. Its classes have the same semantic of the counterpart classes of DOLCE, but limited to the e-Government domain of discourse. We generally use the egov- prefix for distinguishing the classes of the DOLCE e-government mapping ontology from the respective counterparts of DOLCE.

The taxonomy of the DOLCE e-government mapping ontology follows the structure of DOLCE. Precisely, if a concept A of the DOLCE e-government mapping ontology is child of the concept B of the DOLCE e-government mapping ontology, then the counterpart of A in DOLCE is child of the counterpart of B in DOLCE. Thereby, a concept of the DOLCE e-government mapping ontology inherits from its father in the taxonomy of the DOLCE e-government mapping ontology and from its counterpart in DOLCE. As example, we report the definition of the egov-agent concept that corresponds to the agent concept of DOLCE, and is child of the egov-thing concept in the DOLCE e-government mapping ontology taxonomy. The egov-agent definition adds new slots that are not defined in the DOLCE agent concept.

```
Definition 5.3: The egov-agent class
(def-class egov-agent (egov-thing agent) ?x
  ((has-name :type name :min-cardinality 1)
   (has-acronym :type acronym :min-cardinality 1)
   (has-web-address :type web-address :min-cardinality 1)
   (has-email-address :type email-address :min-cardinality 1))
  :constraint (exists (?n ?a ?w ?e) (and (has-quality ?x ?n)
    (name ?n))
    (and (has-quality ?x ?a)
      (acronym ?a))
    (and (has-quality ?x ?w)
      (web-address ?w))
    (and (has-quality ?x ?e)
      (email-address ?e))))
```

This three-stage process is a common technique that allows to extend an upper ontology keeping the links to but without affecting it. Note that the links to the concepts of DOLCE are kept at
level of slot ranges and inherited classes. Keeping links to an upper ontology makes easy future extensions and the integration of existing ontologies. These aspects are particularly important for the CLEO ontology.

5.2.2 The Generic Level of the Domain Ontology

As introduced in the previous section, we built the generic level of the Domain Ontology starting from the DOLCE e-government mapping ontology.

The taxonomy of the DOLCE e-government mapping ontology follows the structure of the top elements of DOLCE; in the following, we show only the root and its direct children.

- **egov-entity**: representing any individual in the e-Government domain; it is the root of the ontology.
- **egov-thing**: representing any DOLCE endurant in the e-Government domain;
- **egov-occurrence**: representing any DOLCE perdurant (events, processes, activities and states) in the e-Government domain;
- **egov-quality**: representing any quality used in the e-Government domain; i.e. qualities are the properties of a egov-entity;
- **egov-abstract**: representing any abstract entity used in the e-Government domain, and, in particular, we refer to egov-region.
- **egov-situation**: representing any DOLCE situation in the e-Government domain.

We extend such a taxonomy adding new concepts within the following broad categories:

- **Activity and Action**: representing events connected to a communication (inform, notify, complain, enquiry, apply, book), a transaction (issue, develop, sale, purchase, payment, rebate, help), a strategy (manage, monitor, promotion, choose), and a service delivery (assistance, advice, benefit, grant, service, public-service, private-service, ...); actions differ from activities because they represent atomic events (give, provide, receive, obtain, present, request, reply).

- **Actor**: representing generic actors of the e-Government filed (legal-agent, person, citizen, family, ...).

- **Organization**: representing the structure (organization, organization-unit, ...), the size (organization-size, ...) and the aims (educational-organization, non-profit-organization, profit-organization, central-government-organization, local-government-organization, county-council, national-government, borough-council, parish-council) of an organization. An organization is a particular legal agent (actor).

- **Quality**: representing possible attributes of the above concepts (identifier, url, postal-address, fax-number, appellation, gender, ethnic group, faith, currency, ...)

- **Time**: we include a standard ontology of time; we aligned its concepts to egov-quality (entity-duration, entity-start-time, entity-end-time, entity-date, ...) and egov-region (time-entity, time-duration, time-point, calendar-time, ...) concepts of the DOLCE e-government mapping ontology.

As a result, we obtained a meta-ontology that can fit different domains and allows to create instance ontologies simply extending and adapting its concepts.
5.3 The Service Ontology

The Service Ontology makes whole the representation of the scenario, modelling the service delivery knowledge level by means of the SWS technology. It allows to complete the descriptions of (i) the services implemented by means of Web services, (ii) the e-Government processes that can be modeled as a composition of Web services (without user interaction), and (iii) the user requests of services. Moreover, it enables to specify the mechanisms to solve existing mismatch problems at data and process level between distinct viewpoints. In other words, it represents the knowledge useful at runtime.

Because representing SWS’s requires the use of technical concepts, the developers are responsible of the creation of this ontology.

As introduced in Section 4.4.3, we adopt the WSMO approach for SWS’s, and thus the derived Service Ontology is composed by three main modules: Goal Ontology, Web Service Ontology, and Mediation Ontology. Note that our work does not consist in improving existing SWS solutions, but enabling their application in the designed ontological framework. Particularly, we refer to the IRS-III (Section 2.5) approach. IRS-III models the WSMO specification as a set of related knowledge models for the WSMO top level components of Goals, Web Services and Mediators, which are meta-models in corresponding ontologies.

Figure 5.17: Intersection between the knowledge levels modeled by the Service Ontology and CLEO; the former represents the service delivery knowledge level (SWS), the latter the configuration knowledge level (e-Government application). The lines labeled with ”goal” and ”service” represent the belongings of these two concepts to both knowledge levels.

Figure 5.17 depicts the intersection between the representations of the SWS’s and e-Government application context provided by the Service Ontology and CLEO, respectively. The two ontologies model two distinct knowledge levels that are provided by distinct kinds of actor, and obviously described by distinct modules. However, they intersect onto two concepts: the goal and service descriptions. Such concepts are their points of contact, and, thereby, the way to actualize the integration between SWS and the e-Government application context descriptions. This conceptual overlapping allows the following integration directions:

- **Using SWS descriptions within the context description:** WSMO compliant descriptions of goal and services can be directly adopted within the context description for describing user’s requests and Web services.
- **Inferring knowledge from the context for creating SWS descriptions:** the description of the context where the service are supplied may represent the base for the definition of WSMO compliant goal and service descriptions. This also involves the definition of the mediation mechanisms.
To carry out the above integration purposes, we firstly derived the concepts of the Service Ontology from WSMO meta-models (referring to the IRS-III implementation) and then aligned such concepts to the CLEO meta-models (Figure 5.18). Axioms and rules enrich the Service Ontology for specifying the above alignment, and inferring knowledge from CLEO in order to complete the WSMO compliant descriptions and handle mismatches. Service Ontology may integrate further SWS approaches, simply adopting the same alignment mechanism (Figure 5.18).

Figure 5.18: Aligning web service concepts of the Service Ontology: the class WSMO-Web-service is defined as sub-class of both CLEO and WSMO web service descriptions. Further SWS approaches simply adopt the same alignment mechanism.

In the following sections, we introduce the three modules that compose the Service Ontology, and outline the introduced axiomatization.

5.3.1 The Goal Ontology module

The Goal Ontology module provides a meta-model for the WSMO component goal, which can be used as goal description in CLEO. It defines the tasks that a service requester expects a Web service to fulfill. In this sense they express the service requester’s intent.

To define the requested service functionalities, the goal links to the class capability that describes preconditions and postconditions – specifying the information space of the service before and after its execution, respectively [32] –, and assumption and effects – representing the state of the world before and after the service execution, respectively [32].

The specific IRS-III implementation extends the WSMO goal – and web service – definitions. The differences are mainly derived from the fact that IRS-III aims to support capability driven web service invocation². To achieve this, it requires that goals and web services have input and output roles.

The class WSMO-goal is the main element of our Goal Ontology module and inherits the properties from the classes goal of IRS-III meta-model and goal-description of CLEO.

This structure allows both actors (users and providers) and developers to employ such a meta-model in their respective descriptions: viewpoints and Service Ontology. Therefore, we took into account the following cases:

1. the actors adopt the meta-model WSMO-goal in their viewpoint descriptions, and specify the respective WSMO slots and concepts (i.e. input, output, capability, and non-functional-properties); the developers straight refer to the existing goal definitions of the viewpoints for completing the Service Ontology (e.g. introducing mediators and mismatches resolution);

2. the actors adopt the meta-model WSMO-goal in their viewpoints, but they do not specify WSMO slots and concepts – or they partly do it; the developers should complete the WSMO-

²Clients of IRS-III (e.g. agents, applications, etc.) can straight invoke Web services via goals
goal description starting from the existing goal definition and inferring some useful knowledge from the context;

3. the actors do not adopt the meta-model WSMO-goal in their viewpoints, but simply define the goal-description classes of CLEO; the developers should create a WSMO-goal meta-model, map it to the goal-description, and infer some useful knowledge from the context for completing its description.

The above cases are considered for every concept of the Service Ontology that is necessary to align to CLEO.

Aligning the Goal Ontology module

The process of alignment involves the following concepts connected to the WSMO-goal meta-model:

- The class WSMO-non-functional-properties. It is derived from the class non-functional-properties of WSMO and aligned to the class quality-description of CLEO. In fact, both describe non-functional requirements that are respectively represented by means of slots and descriptive entities (non-functional-parameter).

- The input and output roles, which are a precise subset of all of the descriptive entities (cleo-role, cleo-parameter, and cleo-attribute) identified within one or more state of affair descriptions.

- The classes WSMO-pre-condition and WSMO-post-condition, which are expressions that constrain the set of states of the information space to a set of valid starting state for executing the service (precondition) and that must be reached by executing the service (postcondition). Thus, precondition and postcondition are connected to the the transitions that are executed by service (and invoked by a goal). Generally, WSMO-pre-condition refers to state elements – that are connected to the activation of a transition –, while WSMO-post-condition refers to resource elements – that represent the transferred value.

- The classes WSMO-assumption and WSMO-effect, which are expressions that constrain the set of states of the world to the set of valid starting states (assumptions), and that must to be reached by executing the service (effects). The elements of the expressions are concepts of the Domain Ontology selected by means of descriptive entities of a state of affair description.

Starting from the alignment of the above concepts, it is possible to complete the alignment of the slots of a WSMO-goal in order to constraint the elements that can be defined and, in some cases, enabling some knowledge inferences from the context. For instance, to suggest possible input roles, we created rules that backward follow the chain of uses relations from the considered goal description to the associated initial state of affair descriptions, in order to obtain all of the defined descriptive entities. The result is refined by means of an axiom that constraints the possible candidates to the only entities that also plays a role as state or resource element of a transition invoked by the considered goal.

5.3.2 The Web Service Ontology module

The Web Service Ontology module provides a meta-model for the WSMO service component, which can be used as service description in CLEO. It represents the functional behavior of an existing deployed Web service.

The class WSMO-web-service is the main element of our Web Service Ontology module and inherits the properties from the classes web-service of IRS-III meta-model and service-description of CLEO.

As for the previously introduced class WSMO-goal, it links to class non-functional-properties, extends WSMO specifications with input and output roles, and defines the provided capability.

Differently, it describes its interface; i.e. how the Web services communicate (choreography) and how they are functionally decomposed in other Web services (orchestration).
5.3. The Service Ontology

The choreography uses a set of rules (guarded transitions) to specify the flow of operations required for realizing the specific functionality of the Web service.

When a Web Service is composite an orchestration has to be provided. Its input values have to be passed to the orchestration and the result of the orchestration has to be passed back to the Web Service. This aspect is still to be further investigate in WSMO. However, IRS-III proposes an approach where the orchestration follows the decomposition of Goals into sub-Goals and uses GG-mediators for connecting sub-goals and mediating the order and types of inputs between them (Section 2.5).

Aligning the Web Service Ontology module

The process of alignment involves the elements connected to capability, choreography, and orchestration. The alignment of non-functional-properties, input and output roles, and capability is based on an axiomatization similar to the one described for the WSMO-goal case. Since the WSMO orchestration is not still defined, we do not create an axiomatization for it. However, we postpone to the Mediation Ontology sub-section the description of a proposed alignment based on the adoption of mediators.

The class WSMO-choreography is derived from the class choreography of WSMO and aligned to the class interaction-description. In fact, it represents a specific mechanism for describing the interactions of a web service from the client’s perspective. In other words, we can straight adopt the WSMO choreography in context representations for describing specific transitions between two distinct viewpoints that involve web services.

In a WSMO choreography, we can distinguish:

• the state signature that defines the state over which the transition rules are executed, and the types of modes the involved concepts and relations may have.

• the transition rules that express changes of states by changing the set of instances of the state signature. A transition rule is expressed by means of guarded transitions; their general form is given below:

\[ \text{if condition then rules} \]

In our alignment, we consider WSMO transition rules as the particular sub-class of CLEO transition-event that expresses the transitions by means of guarded transitions. To complete such a association, the condition of the transition rule is aligned to the condition of the CLEO transition-event.

In the process of alignment of the class WSMO-web-service, the context allows to infer which transition rules of a choreography may be associated with a web service: we simply refer to all of the transitions that are executed by the web service. These transitions can be already expressed by transition-rules or derived from existing transition-events.

The state-signature of the choreography cannot be aligned to the state condition of a transition, because it represents the state of all of the transitions associated with a web service. Again, we can infer this knowledge from the context. Starting from the a service-description, we can obtain all of the state or resource elements associated with the transitions that the service executes, and then check if such elements are input or output roles in order to define the types of mode. The concepts that play such roles are the elements of the state signature.

5.3.3 The Mediation Ontology module

The Mediation Ontology module provides a meta-model for WSMO mediator component. It is a feature of the WSMO approach, and it cannot be aligned to a specific element of CLEO. It represents a bridge between the elements of the Goal and Mediation Ontology modules (Figure 5.17), and handles data and process interoperability issues that arise when handling heterogeneous systems.

The main concept is defined by the class mediator which is sub-classed into more specific types of mediators (wg-mediator, ww-mediator, gg-mediator, oo-mediator). Source and target components
can be any of the WSMO top level components (class `wsmo-entity`). The mediators differ according to the type of source and target components they can handle and whether it uses a mediation service or mapping rules. Thus, mediators are bridges which can provide conceptual mappings or input transformations from source components to target components. IRS-III supports the implementation of Mediation Services (Section 2.4.2) as Goals as well as the explicit declaration of mapping rules. Since mediation services are implemented as Goals they can simply be invoked resulting in the transformation of the relevant input data.

Inferring knowledge for the mediator definitions

The class `WSMO-mediator` is the main element of our Mediation Ontology module and inherits the properties from the class `mediator` of IRS-III meta-model (Section 2.5).

The proposed axiomatization allows to infer from the context description the existence of WG-mediators between a goal and web service. The central element is the transition event of the interaction module. In fact, if a goal of the Goal Ontology executes a transition event that is executed by a web service of the Web Service Ontology, then exists a WG-mediator that links such a goal and web service. To make easy the description of the mediator, we can also infer from the goal and web service descriptions associated with a WG-mediator the possible input and output roles of the eventually mediation service.

The interaction module of CLEO allows to represent existing data mismatches, but not to solve them. The introduction of OO-mediators can accomplish this task. An axiom states the existence of an OO-mediator when two concepts of distinct domain ontologies link to the same state or resource element of a transition event.

Finally, as described in the previous sub-section, the orchestration can be represented by means of GG and WW mediators. In CLEO, we decompose a complex service in service and need descriptions. The former for composing and the latter for integrating services (Section 5.1.2). In particular, we represent the decomposition by means of two plan descriptions: `decomposition-plan-description` and `goals-plan-description`. The former describe the decomposition of the service, the latter the decomposition of a need. Figure 5.19 shows how the above plan descriptions can be easily mapped to a WSMO orchestration decomposition:

- the element `service-description` is mapped to a WW-mediator linking the current web service to the requested web service (represented by the `service-description`).

- the element `need-description` is mapped to a goal that is decomposed in sub-goals linked by GG-mediators. The sub-goal decomposition follows the `goals-plan-description` of the `need-description`.

Figure 5.19: Mapping the CLEO service decomposition to WSMO orchestration by means of GG and WW mediators
A case study in applying the approach.

The ontological framework proposed in this thesis (Chapter 4) is based on a meta-modelling approach (Section 4.3) that specifies the schema to be followed by the modeling process and the general concepts and relations for describing a specific e-Government service-supply scenario.

In this chapter, we propose and detail a specific application of our ontological framework for placing a rigorous emphasis upon the initial knowledge acquisition phase of a knowledge management process.

We define a precise methodology that first helps domain experts to create a full description of a specific e-Government context by using models close to their experience and specific languages of the multiple involved domains, and then drives the application developers to implement SWS descriptions inferring requirements from the context description.

In fact, we argue that it is important to have a much deeper level of understanding of a scenario from the outset, ensuring that fundamental concepts are captured, described, and understood. While conceptual modelling is often a means by which rich, flexible scenarios can be captured, there is an inherent difficulty in specifying a design later in the development lifecycle. This is compounded by the fact that flexibility often leads towards lack of discipline, or consistency, in modelling, thus there is a need for a concept-led, rigorous elicitation process, prior to SWS’s specification.

To introduce the methodology and associated knowledge structures, we worked out a case study within the change of circumstances scenario. Such a case study also provides a detailed evaluation of the applicability to real-life scenarios of our ontological framework.

The rest of the chapter is organized as follow: Section 6.1 describes some of the shortcomings of requirements capture for multi-agent systems development, before explaining the proposed methodology in Section 6.2. Section 6.3 uses an exemplar case study in the Change of Circumstances scenario to explicate the process in detail, and provide an evaluation of the proposed ontological framework.

In order to make the content readable, we report only some meaningful examples.

6.1 Capturing Requirements

In the e-Government service-supply scenarios, the consideration of communication protocols, behaviours, and allocation of tasks and roles is inherently complicated as the problem domain to be modeled is generally quite elaborate itself.

Polovina and Hill [80] illustrate some of the shortcomings of requirements capture for multi-agent systems development. Use case models are a convenient means of defining actors and for documenting the existing processes. Use case notation is flexible enough to capture some of the richer concepts; however there is no inherent model verification, so it is probable that some significant details will be missed from the first iteration. Beside this, use case analysis is a procedure that elicits process-level tasks without challenging qualitative issues. Moreover, while the process
of describing and articulating uses cases serves to elicit the majority of the involved knowledge, generation of an ontology of terms is mostly based on the existing processes together with the domain experts knowledge and experience. Finally, the assignment of behaviours is often based on current practice, rather than from the systematic iteration from a coherent model.

From the above analysis, they argue that a requirements capture process must therefore incorporate the following:

- A means of modelling the concepts in an abstract way that facilitates the consideration of qualitative issues.
- An ability to reveal more system requirements to supplement the obvious actor to agent mappings.
- An explicit means of model-checking before detailed analysis and design specification.
- Improved support for capturing domain knowledge, with less reliance upon domain experts.

6.2 The methodology

In this section, we propose a methodology that improves the capture of e-Government service-supply scenario requirements and knowledge, in a robust and repeatable manner, whilst also eliciting an awareness of significant facets of the scenario much earlier during the knowledge capture phase.

Our methodology embraces the requirements for a capture process defined in the previous section, and extends them in order to create the foundation for SWS descriptions. It is based on the ontological framework introduced in Section 4.4 and formalized in Chapter 5. The modules of CLEO represent the stages of the methodology that have to be followed, while its meta-models define the structure of the knowledge that have to be represented. CLEO indicates how real-life interaction scenarios can be decomposed and translated into models. The Domain Ontology will collect the terms extracted during the elicitation process, and the Service Ontology will contain the final result of the process: the SWS’s specification.

As the ontological framework offers a multi-viewpoint approach, and following the idea of a distribute and cooperative development of the scenario representation (Section 4.3), the proposed methodology involves multiple actors that are responsible of the distinct steps. The methodology is summarized in the following stages:

1. **Life event and actor analysis.** The e-Government scenario is segmented along two orthogonal dimensions: life events, and actor’s viewpoints. Segmentation allows to focus on a reduced and well-delimited sector of the scenario.

2. **Viewpoint analysis.** It represents the distribute and cooperative phase. Each identified actor independently defines its viewpoint on the life event. The adequate class *life-event-description* (user, provider, manager, or politician) has to be adopted for specifying the structure of the viewpoint. The following sub-stages are involved in this phase:

   - (a) **State of Affairs analysis.** Main concepts of the domain are modelled as descriptive entities and used to describe the overall scenario of the problem that is being investigated.
   - (b) **Interaction analysis.** All of the interactions between couples of user-provider and provider-provider viewpoints are identified and described by means of the Interaction Description module. This module is an agreed pattern that enables the two viewpoint’s developers to make a much more knowledgeable decision by considering the entities of a candidate interaction (communication or transaction) in a single integrated, interoperable environment. In other words, it imposes a check rigour upon the previous model and arises the need to discover the appropriate knowledge for the model that have not been captured yet.
6.3. A Change of Circumstance Case Study

- (c) **Conception analysis.** The description of the scenario is improved adding the conceptions (need, offer, policy, legislation) of the actors onto the state of affairs previously defined. In the cases of user and provider viewpoints, the defined conceptions refer to the defined interaction descriptions.

- (d) **Plan analysis.** It describes the processes and dynamics within the viewpoint. Concepts connected to events and tasks are elicited.

All of the identified concepts populate the *generic level* of the Domain Ontology, creating the specific *specific level* ontologies of the viewpoints.

3. **Model Specific Scenario.** Instances of the descriptions and concepts are created. In this way, the model is tested forcing a set of check axioms and rules to refine the representation.

4. **Create SWS descriptions.** The obtained model is used as input of the SWS descriptions provided by developers.

The methodology thus also incorporates an implicit means of validating the resulting model, both as a means to drives iterations during the modelling process, and as an overall validation prior to SWS's specification.

### 6.3 A Change of Circumstance Case Study

To illustrate the methodology and give an evaluation of the proposed ontological framework (Section 5), we describe in detail the elicitation steps with respect to an ongoing case study in the change of circumstances scenario of the healthcare domain [31], [15], within the DIP project, as part of a portal for the Essex County Council (ECC) (UK).

Home-based community care delivery is an example of a complicated, multi-viewpoint social-care system that is plagued with inefficiencies and logistical problems. Social care systems typically comprise a large number of autonomous functions and services, each integrating and communicating with a variety of protocols. Thus the problem domain involves a vast number of quantitative and qualitative issues that must be captured and represented clearly then translated into SWS applications.

We will comment on the requirements and use of the ontological framework within the scenario implemented.

### Data sources

SWIFT and ELMS databases represent the data sources of the present case study.

The SWIFT system is a care system, and holds data for the care the community gives to the people, how they help them making their live easier. It stores data about the person’s life and how the community takes care from him: i.e. the care around the person.

ELMS is just a management system to get the equipment – order it to a warehouse.

The ELMS system and the SWIFT system both have duplicated data. They both hold personal details from the patients. There can be inconsistencies. They are not normalized.

### 6.3.1 Overview of use case

The change of circumstances scenario can be defined by means of the following example:

*A part-time employed single woman moving into a new rented house, in the same local authority area as their previous address, in order to look after her disabled 86 year old mother, whose previous

---

1 Data, Information and Process Integration with Semantic Web Services (http://dip.semanticweb.org)
address was also in the same local authority area. As a result of the mother’s disability they can claim for several benefits.

The mother has several diseases; she’s got a heart disease, which impedes her to make any efforts (e.g.: she can’t lift heavy items) and a knee disease as well, which stops her to do several daily tasks (like climb stairs or get into the bath). Because of these difficulties, they want to hire several devices designed for impaired people’s houses, in order to help her with her daily work.

At some point her disease gets worse, so she needs some equipment at home to do her daily tasks. The doctor refers the person to the OT (Occupational Therapist - an OT works for the “Social Services” department). The OT visits the patient to assess if the person really needs some equipment.

The OT gives the patient a questionnaire to fill out and asks her a lot of questions. Afterwards he/she makes her some tests – an OT is normally a doctor or a nurse. With the results of the tests and the questionnaire, the OT assesses if the patient needs some equipment or services.

E.g. services: Meals on wheels, someone goes and cleans the house, someone goes and sleeps with the patient, etc. The services are recorded on the SWIFT DB.

E.g. equipment: Stair lift, wheel chair, crutch, etc. The equipment is ordered to the ELMS system and recorded there. The SWIFT DB does not store this information.

Assessment of non-equipment: If the OT agrees that the patient needs some services, they are ordered to the SWIFT system and the information stored on the SWIFT database.

Assessment of equipment: If the OT agrees that the patient needs some equipment, there are two possible scenarios:

- If the patient needs the equipment for less than 6 months: the Hospital manages and pays for it - This process does not go trough the Essex County Council (ECC) and the data is not stored on the ELMS system.

- If the patient’s disease is supposed to persist more than 6 months then the equipment is charged to ECC, who manages it. The data is stored on the ELMS system.

We are only going to focus on the long-term (≥ 6 months) diseases or impairments. ECC pays for the equipment if the person’s disease is expected to last more than 6 months.

No one (nor the hospital, nor ECC) charge people for the equipment. It’s for free. But there are normally waiting lists for the equipment. But if people want to get it right now they can purchase the equipment so they don’t have to wait - this data is not stored anywhere then.

Then the OT assesses what equipment the patient needs. Depending on the price of the equipment, different people have to approve it. (e.g.: the OT can be allowed to order some equipment, for some equipments it has to be a Team Manager, an assistant social worker, a social worker, etc. Each kit has its own approver.

Once it is approved (by the OT or another person), the equipment is ordered (data stored in the ELMS database). The order has a budget-code, which is the main identifier of an order. Currently, this order is sent from the OT to the ELMS system in a “free-text notes” secure mail format. Then an operator has to read the mail and manually introduce the relevant data into the ELMS system (Client details, kit issued, dates, etc.).

People do not have to give things back; they can have the equipment for life time. But that is a big waste of money for ECC. Some people do return things, and that it is stored on the ELMS database (returning date). Assessment and decision making on individual client cases are central to the work of professional staff in Community Care.
Records on individual people (“clients”) with whom Community Care has contact are built iteratively to incorporate key facts and notes (“concerns”) and maintained in a proprietary database system (SWIFT).

Staff in different agencies involved in services affected by a Change of Circumstances need to interact with one or more other agency and frequently refer clients to another agency for particular services or other purposes. Case workers in Community Care interact with multiple agencies to activate different services such as health, housing, etc.

When a client’s circumstances change, case workers in Community Care have a co-ordination role, which are frequently centred on tracking changes of the living address of the client.

6.4 Live event and actor analysis

The first stage is to examine the use case in order to identify the life events within it.

In the following we list the two considered life events:

• **Patient Move House**: A patient of the Social Services notifies that he/she changed address. This event triggers some changing of the information stored in the SWIFT and ELMS databases, and checking the eligibility of the patient to old and new services and benefits provided by the involved organizations. In case of eligibility of new services, a new patient assessment is necessary.

• **Patient Passes Away**: A patient of the Social Services dies. The date of deceasing should be set in the SWIFT database, and services and benefits have to be canceled.

The second stage is to define the actors that may describe their viewpoints. In this case study two public administrations were involved:

• **Community Care (Social Services) in Essex County Council**: they typically have a co-ordinating role in relation to a range of services from a number of providers and special responsibility for key services such as support for elderly and disabled people (day centers, transportation). It uses the SWIFT database as its main records management tool.

• **The Housing Department of Chelmsford District Council**: it handles housing services and uses the ELMS database.

Moreover, the end-users are the case workers of the Community Care department that help citizens to report his/her changes of circumstance (e.g. address) to different agencies involved in the process. In this way, the citizen only has to inform the council once about his/her change, and the government agency automatically notifies all the agencies involved.

In this case study, we do not considered politician and manager viewpoints, because the aim is to investigate how agreed Public Administrations can adopt SWS infrastructure for developing an interoperable and integrated service-oriented system (i.e. we focus on configuration and service delivery knowledge levels).

In the present dissertation, we refer to the first life event, and we show examples for each step of the methodology.

As output of this first analysis, we report the following box that shows the definition of the class **patient-moves-house**. The life event is associated with three classes – **case-worker-PMH-WP**, **community-care-PMH-WP**, and **housing-dpt-PMH-WP** – that represent the afore identified viewpoints. Note that the **patient-moves-house** inherits from two more general life events – **move-house-life-event** and **community-care-related-life-event** – that are part of a taxonomy of life events.
A case study in applying the approach.

Definition 6.1: The life event patient-move-house

\[
\text{(def-class patient-moves-house)
\text{(move-house-life-event community-care-related-life-event) ?x}
\text{(satisfies :cardinality 3))}
\text{:constraint (exists (?wp1 ?wp2 ?w3)
\text{(and (satisfies ?x ?wp1)
\text{(case-worker-PMH-WP ?wp1))
\text{(satisfies ?x ?wp2)
\text{(community-care-PMH-WP ?wp2))
\text{(satisfies ?x ?wp3)
\text{(housing-dpt-PMH-WP ?wp3))))})}
\]

6.5 Viewpoint analysis

At this stage, we devise three teams for creating three descriptions according to their viewpoints: one user description for representing user requests, and two provider descriptions for representing available services.

The three teams work independently building and using the respective lexical layers, and interfacing only for coming to an agreement at the Interaction analysis stage. In this way, we can simulate the situation of distributed organizations that, driven by the framework, can autonomously describe their own domain.

Figure 6.1 shows that, for each domain, we refer to two ontologies: one that will contain the terms associated with the legacy systems (SWIFT, ELMS), and one that will contain other specific terms of the domain. All of the above ontologies will form the specific level of the Domain Ontology (Section 5.2). Without losing generality, we assume that the case worker and community care viewpoints share the same ontologies.

The introduction of the three actor’s viewpoints elicits three new terms that extend the generic level of the Domain Ontology: case-worker, county-council, and district-council. They respectively play the roles of end-user and service providers in the analysed context.

The following box shows the skeletons of the case-worker viewpoint descriptions (class case-worker-PMH-WP). At this stage, only the actor’s viewpoint is defined (class end-user-case-worker), but after every step this description will be updated. Finally, it will gather all of the representations associated with the viewpoint.

Figure 6.1: Reference domain ontologies for the considered viewpoints.
6.5. Viewpoint analysis

Definition 6.2: The life event viewpoint definitions at kick-off stage

(def-class end-user-case-worker (end-user) ((played-by :type case-worker)))

Note that the viewpoint analysis is the heart of the methodology. In this phase a large part of the context and lexical layer description is provided. In the following sub-section, we summaries our work, reporting only some meaningful examples.

6.5.1 State of Affairs analysis

The first step of the Viewpoint analysis is to identify the main concepts of the domain, and describe the states of affair where the services are requested and provided within the life event. As introduced in Section 5.1.2, each state of affair defines the involved actors, resources, information, attributes, functional and non-functional parameters, and the relations among them. The aim is to obtain a first representation of the involved knowledge for each viewpoint (Figure 6.2). The concepts identified in this analysis enrich the defined Domain Ontology (Figure 6.1).

In the following, we summaries the analysis of the three viewpoints.

- A **case worker** is involved in two main situations connected to the patient move house life event: (i) collecting patient information, and notifying his/her change of address; (ii) checking the patient eligibility to old a new services and benefits, and eventually opening a new patient assessment.

Such two situations have been mapped onto two couples of initial and final state of affair descriptions: *Change of Address* and *New Patient Assessment*. The initial ones are descriptions of *service-request* state of affair, while the final ones are descriptions of *processed* states of affair (Section 5.1.2).

- In the patient move house life event, a **Community Care** department is involved in situations where it supplies services for retrieving information from the SWIFT database, managing the change of address notification, and eventually revoking patient services and benefits. Such situations have been mapped onto a couple of initial and final state of affair descriptions. The initial one is a description of a *service-available* state of affair, while the final one is a descriptions of a *processed* states of affair (Section 5.1.2).

- In the patient move house life event, a **Housing Department** is involved in situations where it supplies services for retrieving information from the ELMS database, updating the address...
information of a client, and opening new client assessments for providing care equipments. Note that we are using the client concept instead of the patient one, because we changed the domain and thus some of the used terms. As in the community care state of affairs, such situations have been mapped onto a couple of initial and final state of affair descriptions, and the initial one is a service-available state of affair, while the final one is a description of a processed state of affair (Section 5.1.2).

As example, we show the New Patient Assessment initial state of affair from the case worker viewpoint (class case-worker-PMH-assessment-initial-SOA). It describes a situation where a patient speaks with a case worker of a community care department, and supplies to him/her the new address and moving date information. The case worker retrieves more information about the patient from the system, and then notify the new data. Italic words represent elicited concepts of the context that have been used to describe the viewpoint.

<table>
<thead>
<tr>
<th>Definition 6.3: The state of affair description case-worker-PMH-assessment-initial-SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(def-class case-worker-PMH-assessment-initial-SOA (service-request) ?soa</td>
</tr>
<tr>
<td>(uses-role :cardinality 8) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) }</td>
</tr>
</tbody>
</table>

In the previous box, the expressed-by slot shows the elements involved in the state of affair and the relations among them. Note that a service-request state of affair is composed by a set of relations that give a clear idea of the required and supplied resource for accomplishing the requested services.

The following Table 6.1 shows the descriptive entities used in the above description. Note that each descriptive entity captures a concept of the community care domain.

| Each state of affair description represents some new concepts (e.g. assessment) of the Domain Ontology or reuses others that have been previously extracted (e.g. case-worker). The same concept may play distinct roles within the same description or between distinct descriptions. For instance, the list concept identifies the list of existing-services, but also the list of eligibility-equipments. A further example is provided by the case-worker concept that identifies the applicant (requester) of the service, but he/she also plays the role of e-government actor (viewpoint) in the user life event description. This mechanism does not allow to create duplications of the same concept, and describes the different aspects that a concept can play. |

In this example, it is clear the absence of dynamics in the description. The case worker requires and supplies information, but we do not know when and how. As introduced in Section 5.1.3, this aspect will be captured by the associated plan and interaction descriptions.
6.5. Viewpoint analysis

<table>
<thead>
<tr>
<th>Type</th>
<th>Descriptive Entity</th>
<th>Captured Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicant (role)</td>
<td>patient-case-worker</td>
<td>case-worker</td>
</tr>
<tr>
<td>actor (role)</td>
<td>community-care-department</td>
<td>county-council</td>
</tr>
<tr>
<td>actor (role)</td>
<td>housing-department</td>
<td>district-council</td>
</tr>
<tr>
<td>information (role)</td>
<td>patient-information</td>
<td>client</td>
</tr>
<tr>
<td>information (role)</td>
<td>existing-services</td>
<td>list</td>
</tr>
<tr>
<td>information (role)</td>
<td>eligibility-equipments</td>
<td>list</td>
</tr>
<tr>
<td>information (role)</td>
<td>cancel-service</td>
<td>care-item</td>
</tr>
<tr>
<td>information (role)</td>
<td>patient-assessment</td>
<td>assessment</td>
</tr>
</tbody>
</table>

Table 6.1: Descriptive entities of the state of affair description case-worker-PMH-assessment-initial-SOA

6.5.2 Interaction analysis

The State of Affair analysis provided a first description of the three viewpoints of the scenario. We elicited some concepts of the Domain Ontology, and outlined the situations where the services are requested and provided. The present analysis refines the existing descriptions, considering new aspects such as the dynamic of the scenario (i.e. the interaction between viewpoints), the source and the destination of the exchanged resources, the condition for exchanging resources, etc. This means that elements captured by a viewpoint can be introduced into other viewpoints, and the need of balance in an interaction can arise the necessity of further resources. As introduced in Section 5.1.3, the constraints defined in the interaction module impose a check rigour in the definition of the new elements. In particular, they requires that concepts playing a state and resource role in a transition should also play a role in a defined state of affair description. Finally, the existing data and protocol mismatching may be discovered and represented. Figure 6.3 shows the role played by the Interaction module.

![Figure 6.3: The interaction analysis.](image)

The aim is to obtain a quantitative and qualitative representation of the scenario composed by descriptive entities of the situation and interaction descriptions, and the counterpart concepts of the Domain Ontology. Such a representation is the foundation for describing needs, offers, and processes in the following stages of the methodology.

In the patient moves house life event, we described five interactions between the three involved viewpoints: case worker, community care, and housing department. Figure 6.4 depicts the interaction descriptions linking the three viewpoints. The arrows indicates the direction of the transferred value: single-way arrows represent a communication interaction; double-way arrows represent a transactional interaction.

- **Retrieve-patient-info.** It is an enquiry description that adopts its unique transition event
for transferring the patient record resource that contains the complete information about a patient. The event activation condition requests the existence of partial information about the patient for starting the query, and, in particular, at least the name of the patient.

- **Notify-change-address.** It is a notification description that adopts its unique transition event for communicating the information about the patient new address in order to update the legacy system information. The event activation condition requests the existence of information about the moving patient, a notification acknowledgment, and the execution of the transition only after the moving date. The community care department collects the notification and informs the involved organizations.

- **Notify-change-address-HD.** It is a notification description that adopts its unique transition event for forwarding the information about the patient new address to the housing department. The event activation condition requests the existence of information about the moving patient, a notification acknowledgment, and the execution of the transition only after the moving date. This communication follows the above notification.

- **Cancel-service-description.** It is a transaction description that adopts its two transition events for finalizing services that are no more eligible to the patient. The case worker queries the community care department for obtaining the list of services that a patient is currently receiving. From this list, he/she selects the services to finalize.

- **Open-assessment-description.** It is a transaction that exchanges values between the case worker and the housing department in order to supply new care equipments to the patient and thus open an assessment. The case worker queries the housing department for obtaining the list of care equipments that a patient can use. On the base of the available equipments, he/she requires a new patient assessment.

As example, we detail the open-assessment-description. It adopts the following two transition events for retrieving services eligible to a patient and eventually opening a new patient assessment:

- The *list-equipments-event* represents an enquiry returning the list of all of the equipments that the moving patient is eligible to. Its activation condition requests the existence of patient weight and impairment information for querying the ELMS database. As we detail in the follow, this activation condition will arise the necessity of refine a state of affair description.

- The *open-assessment-event* represents a communication for giving the adequate patient information to open a new assessment. Its event activation condition requests the existence of the patient list of requirements, and the assessment information record.
The *inside and outside agent* roles are respectively played by the *case-worker* that requires the transaction and the *district-council* that plays the housing-department role. As we describe two transition events, we consider the respective two *transition resources*:

- The *list-equipments-resource* describes the resource *list-of-equipments* that represents the list of care equipments that can be provided to a patient. The resource is played by the concept *list* both in the source and destination domains. The outside agent is the source and the inside agent the destination of the transition.

- The *open-assessment-resource* describes the resource *patient-info-assessment* that represents the patient information that should be provided. The resource is played by distinct concepts in the source and destination domains (*data mismatch*). The inside agent is the source and the outside agent the destination of the transition.

Note that the inside and outside agents exchange their roles (source and destination) each other in the two transitions. Then, we consider the following two *transition activations*:

- The condition of the first transition is constituted by a unique state condition. It requires the existence of the two following state elements: *patient-weight-element*, and *patient-impairment-element*. The former represents the weight of a patient, and it is a specific attribute of the patient information. The latter represents the impairment of a patient, and it is a specific attribute of the patient information too.

  Checking the constraints of the state element class it arises that the domain concepts selected by the above two elements do not play a role in the state of affair descriptions of the case worker viewpoint. This leads to the necessity of refining the New Patient Assessment initial state of affair (that is the one provided as example in the previous State of Affair analysis), introducing two new attributes that specialize the descriptive entities *patient-information*.

- The condition of the second transition is also constituted by a unique state condition. It requires the existence of the following two state elements: *received-list-equipments*, and *transition-patient-assessment*. The former represents the list of equipments obtained after executing the previous transition; this means that the current transition can be executed only after the previous one. The latter represents the information returned after the assessment opening, and it is played by two distinct concepts *assessment* in the source and destination domains (*data mismatch*). This condition allows to better understand the dynamic of the scenario: i.e. the order of the transitions.

The following box shows the state condition definition of the second transition and one of the two associated state elements. The slot *expressed-by* introduces the conditions on the state. Note that the received list of equipments should not be empty. Moreover, the state element definition represents the data mismatch between the source and the destination domains.
The described interaction analysis introduced a lack in the state of affair analysis. The bug regarded two possible service inputs, and thus the early discovery avoided problems at SWS definition level. For instance, developers could create a goal description using generic information of the patient as inputs and a possible web service that satisfy the goal using weight and impairment information as a goal. This problem could be solved only introducing complex mediators between the goal and web service descriptions.

In this phase, we match elements from the two distinct domains. The introduced state and resource elements arose data mismatching, that will solved later in the Service Ontology with the creation of appropriate OO-mediators.

### 6.5.3 Conception analysis

At this stage of the methodology, we have a life event description segmented into three viewpoints. Each viewpoint autonomously described its state of affairs connected to the life event. The viewpoints are linked by means of interaction description, representing the existing resource transitions between them.

As described in Section 5.1.2, the conception analysis allows to describe what an actor – representing a viewpoint – expresses (conceives) in a particular state of affair. The link between a conception description and one state of affair is created by means of the cleo-conceives relation.

In the specific cases of user and provider viewpoints, the created conception descriptions (need or offer) link to one or more interaction descriptions whose transitions operating on (Figure 6.5). Moreover, goal and service descriptions represent the decomposition of a conception in active/computable steps that links to specific transitions of a interaction description.

In the following, we outline the conception descriptions that have been created within each viewpoint.

- The case worker defines two initial state of affairs associated with Change of address and New patient assessment. In the first state of affair, the case worker conceives the following two need description:
  
  - get-patient-information-need: the case worker enquiries more information about the patient. The description refers to the interaction retrieve-patient-info.
– **notify-change-of-address-need**: the case worker notifies to all the involved organizations the patient change of address. The description refers to the interaction *notify-change-of-address*.

In the second state of affair, the case worker conceives the following two need description:

– **cancel-services-need**: the case worker needs to finalize some services currently provided to the patient, after checking the whole list. The description refers to the interaction *cancel-service-description*.

– **open-assessment-need**: the case worker needs to open a new assessment, after checking the list of care equipments that are eligible to a patient. The description refers to the interaction *open-assessment-description*.

• The **community care department** defines one initial state of affair, where conceives the following four descriptions:

  – **get-patient-information-offer**: the community care department offers to supply information about a patient. The description refers to the interaction *retrieve-patient-info*.

  – **notify-change-of-address-offer**: the community care department offers to update the new address in the SWIFT database, and notify the change of address to the housing department. The description refers to the interaction *notify-change-of-address*, and uses a complex service (*notify-change-address-service*) that is decomposed into a service and need descriptions. The former updates the address information in the SWIFT database (local service, known a priori); the latter expresses the need of notifying the change of address to the housing department (*delegated service*).

  – **notify-change-of-address-HD-need**: the community care department notifies the change of address to the housing department. The description refers to the interaction *notify-change-of-address-HD*. Differently from the need descriptions defined so far, it is not used by a viewpoint description, but a complex service (*notify-change-address-service*).

  – **cancel-services-offer**: the community care department offers to supply the list of services that a patient is currently receiving, and finalizes some services of that list. The description refers to the interaction *cancel-service-description*.

• The **housing department** defines one initial state of affairs, where conceives the following two offer descriptions:

  – **notify-change-of-address-housing-department-offer**: the housing department offers to update the address information of a patient in the ELSM database. The description refers to the interaction *notify-change-of-address-HD*.

  – **open-assessment-offer**: the housing department offers to open a new assessment after supplying the list of care equipments that are eligible to a patient. It represents a transaction offer description, and refers to the interaction *open-assessment-description*. 

![Figure 6.5: The conception analysis.](image)
As examples, we detail the definitions associated with the open-assessment-need and open-assessment-offer. Figure 6.6 depicts the specific situation we are going to describe. Note that is an instantiation of the Figure 5.13 in Section 5.1.3 (Modelling interaction between viewpoints).

The open-assessment-need uses the following two goals: list-equipments-goal, and open-assessment-goal. These goals are respectively execution for the transitions list-equipments-event and open-assessment-event of the transaction description open-assessment-description. The need description uses the plan open-assessment-need-plan for representing the sequence of the two goals. In the following box, we report the definition of the need description and its goal list-equipments-goal.

The interaction description open-assessment-description links the above need to the offer description open-assessment-offer. The latter uses the following two services: list-equipments-service, and open-assessment-service. The former is execution for the transition list-equipments-event; the latter is execution for the transition open-assessment-event.

The service list-equipments-service is a complex service that uses the need description retrieve-list-equipments-need for describing a delegation in terms of the following three goals: finds-items-matching-weight-goal, finds-items-matching-impairment-goal, and list-intersection-goal. The first requires a service that finds care equipments for a patient with a specific weight, the second one finds care equipments for a patient with a specific impairment, and the third one intersects the results of the previous two services. The need description specifies the plan retrieve-list-equipments-need-plan that arranges the three used goals.
In the following box, we show the definition of the complex service `list-equipments-service` and the associated need description `retrieve-list-equipments-need`. Note the decomposition into three goals.

**Definition 6.6:** The service description `list-equipments-service` and its delegation description `retrieve-list-equipments-need`

```lisp
(def-class list-equipments-service (complex-service-description) ?s
  ((used-by :type open-assessment-offer)
   (defines :type list-equipments-service-t)
   (service-executes :cardinality 1)
   (uses-need :cardinality 1)
   (uses-plan :type list-equipments-service-plan)
   (uses-quality :type list-equipments-service-q))
:constraint (exists (?te ?nd)
  (and (service-executes ?s ?te)
       (list-equipments-ev ?te)
       (uses-need ?s ?nd)
       (retrieve-list-equipments-need ?nd))))

(def-class retrieve-list-equipments-need (need-description) ?nd
  ((used-by :type list-equipment-service)
   (defines :type retrieve-list-equipments-need-t)
   (uses-interaction :type retrieve-list-equipments)
   (uses-quality :type retrieve-list-equipments-need-q)
   (uses-goal :cardinality 3)
   (uses-plan :type retrieve-list-equipments-need-plan))
:constraint (exists (?g1 ?g2 ?g3)
  (and (uses-goal ?nd ?g1)
       (finds-items-matching-weight-goal ?g1)
       (uses-goal ?nd ?g2)
       (finds-items-matching-impairment-goal ?g2)
       (uses-goal ?nd ?g3)
       (list-intersection-goal ?g3))))
```

### 6.5.4 Plan analysis

This is the last stage of the viewpoint analysis. Each viewpoint is completed with the description of all of the plans that describe procedures, processes, etc. As described in Section 5.1.3, the plans organize goals or services within a need or offer description, or need and offer within a life event description (viewpoint). This phase allows to elicit new knowledge about tasks and events of the domain, completing the description of the dynamic.

The created plans link the conception descriptions to the final state of affair descriptions of the viewpoints (Figure 6.7):

- Plan and conception descriptions are connected by means of the tasks defined by the conception descriptions and used by the plan;
• Plan and state of affair descriptions are connected by means of the *cleo-achieves* relation;

In our case study, we identified the following plans grouped in the three viewpoints.

• **Case Worker.** We defined four plans connected to the four need descriptions, and one plan that arranges the four need in the viewpoint description.

  The more interesting plans are the ones connected to the transaction need descriptions. The *cancel-services-need-plan* describes a sequence of two tasks associated with the two goals of the need. The second goal-task is iterated in a loop, until the list of services to cancel is empty, or the user decides to exit. The *open-assessment-need-plan* describes a simple sequence of two tasks associated with the two goals of the need.

  The *need-plan of the case worker viewpoint* organizes in a sequence its four needs:

  - Get patient information;
  - Notify change of address;
  - Cancel services;
  - Open assessment.

  Such a sequence of needs represents the four steps that a case worker should follow in order to accomplish all the tasks connected to a patient moves house life event.

• **Community Care.** We defined three plans connected to the three offer descriptions, and one plan that arranges the three offer in the viewpoint description. Beside this, we created a plan for representing the functional decomposition of the notify change of address complex service, and another plan for the need description that represent the delegation expressed in the complex service.

  The more interesting plan is the decomposition of the notify change of address service described by the plan *notify-change-address-service-plan*: the plan defines a sequence of tasks respectively defined by the *update-change-address-service* and the *notify-change-address-HD-need*.

  The plan connected to the transaction offer description *cancel-services-offer* simply contains (as a bag) the two tasks defined by the services of the offer. It does not specify any arrangement between the two tasks.

  The *offer plan of the community care viewpoint* defines the following three offer tasks:

  - get patient information
  - notify change address
  - cancel services

• **Housing Department.** We defined two plans connected to the two offer descriptions, and one plan that arranges the two offers in the viewpoint description. Beside this, we created a plan for representing the functional decomposition of the list equipments complex service, and another plan for the need description that describes the delegation expressed in the complex service.

  The plan connected to the transaction offer description *open-assessment-offer* simply contains (as a bag) the two tasks defined by the services of the offer. It does not specify any arrangement between the two tasks.

  The *list-equipments-service* delegates its functionalities by means of a unique need description. Thereby, its plan simply contains the task associated with the need. The most interesting case is represented by the plan of the need description that we report below as example.
As introduced in the conception analysis, the need retrieve-list-equipment-need contains three goals. Two of them ask for a list of equipments on the bases of client weight and impairment and can be invoked freely; the third one asks for intersecting the two above lists and can be invoked only after the other two. The associated plan introduces the following couple of control tasks: any-order-task and syncro-task. All of the tasks within this couple of tasks can be executed freely. The syncro-task synchronizes the previous tasks before the execution of the last one. The following box shows its definition: \( ?t1, ?t2, ?t3 \) are the three tasks associated with the goals; \( ?a, ?s \) are the any-order-task and syncro-task, respectively. Note that \( ?a \) and \( ?s \) are successors but not direct-successors.

\[
\text{Definition 6.7: The plan description \textit{retrieve-list-equipments-need-plan}}
\]

\[
\begin{align*}
(\text{def-class} & \text{retrieve-list-equipments-need-plan} \ (\text{goals-plan-description}) \ ?pd \\
& ((\text{uses-task} : \text{cardinality} 6)) \\
& :\text{constraint} \ (\exists (?a \ ?t1 \ ?t2 \ ?s \ ?t3) \\
& \quad (\text{uses-task} \ ?pd \ ?a) \\
& \quad (\text{retrieve-list-equipments-any-order-task} \ ?a) \\
& \quad (\text{uses-task} \ ?pd \ ?t1) \\
& \quad (\text{finds-items-matching-weight-goal-t} \ ?t1) \\
& \quad (\text{uses-task} \ ?pd \ ?a) \\
& \quad (\text{finds-items-matching-impairment-goal-t} \ ?t2) \\
& \quad (\text{uses-task} \ ?pd \ ?a) \\
& \quad (\text{retrieve-list-equipments-syncro-task} \ ?s) \\
& \quad (\text{uses-task} \ ?pd \ ?a) \\
& \quad (\text{list-intersection-goal-t} \ ?t3) \\
& \quad (\text{successor} \ ?a \ ?s) \\
& \quad (\text{direct-successor} \ ?a \ ?t1) \\
& \quad (\text{direct-successor} \ ?a \ ?t2) \\
& \quad (\text{direct-predecessor} \ ?s \ ?t1) \\
& \quad (\text{direct-predecessor} \ ?a \ ?t2) \\
& \quad (\text{direct-successor} \ ?s \ ?t3))))
\end{align*}
\]

### 6.6 Model Specific Scenario

At this point the description of the three viewpoints – and thus the context – have been completed. Note that from such a definition, it is possible to start several inference paths in order to explore and retrieve information from the context. This is based on the simple relations that links the modules of CLEO and the rich axiomatization that characterizes it.

Once the generic model has been created, to assess its viability it is tested with some specific scenario. We create instances of the concepts of the Domain Ontology (e.g. the Essex county council, the Chelmsford district council, dummy citizens and case workers, etc.). These instances populate the lexical layer of the ontological framework.

Further instances may specialize all of the descriptions of the context, describing specific cases. These instances are created starting from the lexical layer: we select instances of the Domain Ontology, we create instances of the descriptive entities that are played by such instances, and then we compose them into an instance of a context descriptions. This is a sort of revers path compared with the one we adopted for eliciting the knowledge. For example, an instance of a viewpoint description (e.g. the case worker viewpoint) is built selecting the instances playing a role in the description (e.g. the Jessica, Robert, Essex etc.), creating the instances of the roles of the description (e.g. jessica-patient, robert-case-worker, essex-county-council, etc.), and finally composing the situation following the defined relations (e.g. jessica-patient speaks with robert-case-worker, etc.). The result is a specific description of a scenario. In the following box, we report, as example, an instance of the case worker initial state of affair.
Definition 6.8: The definition of the instance jessica-PMH-change-address-initial-SOA

```
(def-instance jessica-PMH-change-address-initial-SOA (case-worker-PMH-change-address-initial-SOA)
  ((uses-role jessica-patient)
   (uses-role jessica-info)
   (uses-attribute jessica-current-address)
   (uses-attribute jessica-new-address)
   (uses-role robert-case-worker)
   (uses-role essex-county-council)
   (uses-parameter jessica-moving-date)
   (expressed-by (and
     (patient 'jessica-patient)
     (community-care-department 'essex-county-council)
     (current-address 'jessica-current-address)
     (patient-information 'jessica-info)
     (new-address 'jessica-new-address)
     (moving-data 'jessica-moving-date)
     (speaks 'jessica-patient
       'robert-case-worker)
     (cleo-relates 'robert-case-worker
       'essex-county-council)
     (cleo-specializes 'jessica-current-address
       'jessica-patient)
     (cleo-requires 'robert-case-worker
       'jessica-info)
     (cleo-specifies 'jessica-patient
       'jessica-moving-date))))
```

The creation of instances is a useful mean for checking the consistency of the created descriptions. Inferences based on the axiom and rules of CLEO can help in the creation of the instances (e.g. we can infer the elements of a transition starting from the defined elements of related state of affair), as well as identify any lack in the model. This is the second and more accurate check point of the model (the first was the interaction analysis). In fact, we are able to test all the inference paths provided by the axiomatization of CLEO.

6.7 Create the SWS descriptions

The model created so far describes the context where the services are requested and provided and the involved concepts. As described in Section 5.3, the context provides useful knowledge for the SWS definitions. For instance, it may be the input for the SWS developers that can create goal and web service descriptions, starting from the ones defined in the conception descriptions (Figure 6.8). The SWS definitions that we create in this phase populate the Service Ontology, and complete the description of the scenario with the representation of the service delivery knowledge level.

The developers simply take into consideration the goals of the context, and create new WSMO-goal descriptions mapping to them. They can do the same with the service descriptions of the context for deriving WSMO-web-service descriptions. Finally, they describe the mediators that link the created elements, and provide solutions to existing data or process mismatches.

As example, we report the definitions of goal, web service, and mediator connected to the open assessment transaction between the case worker and housing department viewpoints (i.e. the examples we used in this chapter). The aim of this transaction is to open a new patient assessment after checking the care equipments the patient is eligible to. In particular, we refer to the event of listing care equipments.

The first step is to create the WSMO-goal description. The reference goal in the context is the list-services-goal, defined in the case worker viewpoint. Following the chain of uses relations, it is possible to access to the associated state of affair descriptions (New Patient Assessment).

\[^2\]we consider the case in which the viewpoint descriptions do not directly adopt WSMO concepts (Section 5.3.1)
6.7. Create the SWS descriptions

Moreover, the relation goal-executes links the goal to a specific transition description (list-equipments-resource). The axiomatization allows to obtain the possible inputs and outputs of the WSMO goal simply inferring the description entities defined in the states of affair (initial for inputs, final for outputs) whose counterparts in the Domain Ontology are also counterparts of state or resource elements of the associated transition. The following box shows the WSMO goal definition with the inputs and outputs inferred. The selected inputs are patient-weight and patient-impairment and the output is the list of eligibility-equipments.

Definition 6.9: The wsmo goal list-services-wsmo-goal

(defun list-services-wsmo-goal (wsmo-goal) ?goal
  ((has-input-role :value patient-weight
                  :value patient-impairment)
   (has-output-role :value eligibility-equipments)
   (patient-weight :type number)
   (patient-impairment :type impairment)
   (eligibility-equipments :type list)
   (has-input-soap-binding :value
                            (patient-weight "float")
                           :value
                            (patient-weight "sexpr"))
   (has-output-soap-binding :value
                            (eligibility-equipments "sexpr")
                           (has-non-functional-properties :value
                                                            list-services-wsmo-goal-q)))

In this case, the context does not provide any suggestion about specific capabilities for the goal. Neither the state of affair descriptions nor the transition state conditions defines specific constraints except the existence of the inputs and the outputs, that is implicit with the WSMO goal input and output role definitions.

The second step is to create the WSMO-web-service description. The reference service in the context is list-services-service, defined in the housing-department viewpoint. The same reasoning introduced for the WSMO goal is valid in this case. Thus, we can create the following WSMO web service definition. The inputs are the client-weigh and client-impairment, and the output is the list of eligibility-equipments.
Further the input and output definitions, we introduced choreography and orchestration descriptions (interfaces). Each transition the service executes may be mapped to a guarded transition. The set of all obtained guarded transitions is part of the choreography (other guarded transition can be added by developers for managing more detailed aspects; e.g. errors, acknowledge messages, etc.). In our example, the service description only links to the transition `list-equipments-event`. From such a transition, we can (i) derive the conditions of the guarded transition, referring to the transition condition `list-equipments-state-condition`, and (ii) define the call to a function that retrieves the transition resource element `list-of-equipments`.

The considered service is a complex service, and hence defines a functional decomposition. The plans `list-equipments-service-plan` and `retrieve-list-equipments-need-plan` describe such a decomposition. The former defines a delegation by means of a need description; the latter decomposes the need description in terms of goals. The orchestration is defined in the format: (Sequence G1 G2 G3 M1 M2), where G1, G2 and G3 represent goals and M1 and M2 the GG-mediators connecting them (Figure 6.9). In our example the goals are: `finds-items-matching-weight-goal`; `finds-items-matching-impairment-goal` and `list-intersection-goal`.

The last step is to create WSMO mediator descriptions. The existence of WG-mediators and OO-mediators is proved by means of axiomatization. The mediator descriptions used in this example (Figure 6.9) are explained in the following. Note that all links coming from mediators connect source to target components (labels were omitted to avoid cluttering the diagram).

- **WG-mediator**: connects `list-services-wsmo-web-service` to `list-services-wsmo-goal` allowing it to be selected for solving the goal. This mediator defines a mediation service for converting the value of input weight from pounds (in the goal) to kilos (in the web-service).

- **OO-mediator**: Defines mapping rules for aligning `housing department` domain ontology (used by the Web Service) with `community care` ontology; for instance, it aligns the concept `impairment-HD` to the concept `impairment` that are used as input roles in the firs and the second ontology, respectively.

- **GG-mediator1**: Allows the output of `find-items-matching-weight-wsmo-goal` to be used as input by `list-intersection-wsmo-goal`.

- **GG-mediator2**: Allows the output of `find-items-matching-impairment-wsmo-goal` to be used as input by `list-intersection-wsmo-goal`.

---

**Definition 6.10: The wsmo goal `list-services-wsmo-web-service`**

```
(def-class list-services-wsmo-web-service (wsmo-web-service) ?web-service
  ((has-input-role :value client-weight :value client-impairment)
   (has-output-role :value eligibility-equipments)
   (client-weight :type number)
   (client-impairment :type impairment-HD)
   (eligibility-equipments :type list)
   (has-input-soap-binding :value (client-weight ''float'')
      :value (client-weight ''sexpr''))
   (has-output-soap-binding :value
      (eligibility-equipments ''sexpr''))
   (has-non-functional-properties :value list-services-wsmo-service-q)
   (has-capability :value list-services-wsmo-web-service-capability)
   (has-interface :value
      list-services-wsmo-web-service-interface)))
```
6.8 Summary

In this chapter, we have described the methodology introduced by the modules of CLEO and explained their capabilities of eliciting and representing knowledge. We have clarified the use of the context (CLEO) and the lexical layer (Domain Ontology), stressing their aims.

We applied such a methodology to a change of circumstance case study for evaluating both the ontological framework and the methodology itself. Such a case study presents several facets that allow to test the main feature of the ontological framework. In this chapter, we just reported some examples. In the following, we summaries the concrete results of our work.

The first result has been the creation of four ontologies describing the considered change of circumstance context. We defined an ontology containing all the life events involved in the considered scenario (patient moving house, Passes Away) and three ontologies describing the three actor’s viewpoints on that life events: community care case-worker, community care department, and housing department (Figure 6.10).

The second result has been the elicitation of concepts that populated the Domain Ontology. The latter is composed by the E-government-upper-level-ontology representing its generic level, and two ontologies representing its specific level: Change-of-circumstances-citizen-ontology for the community care department domain, and Change-of-circumstances-equipment-ontology for the housing department domain. Note that case worker and community care department share the same domain. SWIFT-services-ontology and ELMS-services-ontology are further domain ontologies describing the provider viewpoints and strictly connected to the respectively legacy systems.

Finally, we created the Service Ontology that contains the SWS descriptions. Goal, Web Service, and Mediator descriptions have been obtained starting from the knowledge represented in
context and using the concept of the Domain Ontology as building blocks.

Figure 6.11 depicts the Domain and Service Ontologies and illustrates the “inheritance” dependencies of the models [15]. The light-colored rectangles on the top-half of the diagram represent domain ontologies. The dark-colored rectangles on the bottom-half of the diagram represent service ontologies available from Community Care (boxes on the left) and Housing Department (boxes on the right). It is important to note the absence of dependencies crossing the two different domains.

![Figure 6.11: Change of circumstance case study: the Domain and Service ontologies.](image-url)
In this chapter, we introduce the aspects connected to the use and transfer of knowledge. The aim is the adoption of the ontological framework defined in Chapter 4 as the core of a knowledge-based service-oriented system, by designing a middleware that enables the automated integration and paves a common ground for services.

Currently, the middleware is at an early stage of development and its components are not defined and realized yet. Thereby, we simply introduce its high level architecture, and describe its main components and functionalities. The proposed architecture represents the first stage of our next future work. The final result will be a sound framework for supporting e-Government applications. Its architecture will involve the interaction among heterogeneous agents to perform operations in an open and interoperable environment, where operations can be performed regardless of details of implementation for e-Government legacy systems. In particular, the middleware will interface e-Government Web applications with Web services, where the adopted semantic will enable (i) inference paths for driving the user in the discovery and use the services matching his/her needs, (ii) the composition, mediation, and execution of Web services for accomplishing user’s goals, and (iii) the exploration of the scenario’s collective views of captured knowledge by deriving additional links and descriptions that are not explicitly given.

Since the preliminary proposal, the actual purpose of this chapter is to show the advantages and provide an evaluation of the reasoning capabilities of our ontological framework, when adopting the functionalities of the middleware software components defined in the proposed architecture. For this reason, we employ a use case-based narrative description based on a simple but very common life event example: moving house.

The rest of the chapter is organized as follow: Section 7.1 outlines the structure of a Knowledge Management System based on the semantically-enhanced middleware detailed in Section 7.2. Finally, Section 7.3 describes the inference paths and the knowledge that drive the functionalities of the system.

7.1 The proposed Service-Oriented System Architecture

In this section, we define the basic structure of a generic e-Government service-oriented system based on a SWS infrastructure. This architecture defines a knowledge-based system, which is a program based on inference mechanisms to solve a problem by employing the relevant knowledge. The primary goals of the knowledge-based system are selecting a life event applicable to the user’s requirements, identifying the services needed to solve a given situation and matching the user request, and executing the invoked services. The architecture is composed of the loosely-coupled modules outlined in Figure 7.1.

The modules are organized in three layers:

- **User Interaction** supports the user to identify a service and collects information for its execution. The framework defined in the underlying layer, allows the definition of multi-entry points to represented knowledge and services e.g. life events, actor’s viewpoint, state of affairs, etc.
7. A Semantically-Enhanced Middleware

Figure 7.1: The semantically-enhanced infrastructure of a service-oriented system.

- **Middleware** allows the description, publishing, updating, and using of knowledge in order to provide users with an up-to-date and personalized list of available services, and allows the description, identification, instantiation, and invocation of services. This level addresses the semantic interoperability between heterogeneous domains, and the integration of services.

- **Web Services** is responsible for the execution of services. Each PA supplies services through the WS technology. Each one is connected to the back-office and semantically described via the middleware layer. This level addresses the interoperability between different legacy systems.

7.2 The semantically-enhanced Middleware

The core of the architecture is the Middleware: the semantically-enhanced layer responsible for the interoperability and service integration. It is based on the **ontological framework** defined in Section 4.4, which supplies the necessary knowledge to drive the other modules in the accomplishment of their functionalities.

This is one of the most important advantages of knowledge-based approaches. The capabilities of the system are embedded in the knowledge and not in the architecture or code. This introduces more system scalability and flexibility than the traditional approaches. For instance, future evolutions (e.g. adding new web services or changing existing ones) simply involves the evolution of the ontologies, without affecting the other modules.

In the following, we briefly introduce the other modules of our framework:

- The **Application Module** provides the API’s (Application Programming Interfaces) of the middleware. It collects the requests from the front end, and interacts with the other modules in order to provide the requested functionalities. In particular, it interacts with the reasoner to query and access knowledge of the ontological framework, and the workflow management system to process a service.

- The **Reasoner** module is used during all of the activities, and provides the reasoning support for interpreting the semantic descriptions and queries. The application module and workflow management system interact with the ontological framework through the reasoner. Since our
ontologies are implemented in OCML (Section 1.9.1), we will refer to the OCML reasoner [70].

- The **Workflow Management System (WfMS)** is able to load and interpret (through the reasoner) the process definitions of the ontological framework (CLEO Ontology), interact with workflow participants (through the application module) for gathering data and providing results, and invoke the use of SWS's.

- **SWS environment** provides the environment that enables SWS's to interact each other in order to achieve a requested goal. Based on the semantic descriptions defined within the ontological framework (Service Ontology), it allows the invocation, mediation, and execution of the web services of the layer below. Since it is an implemented WSMO-based framework, in our future work, we will refer to IRS-III [29] (Section 2.5) as SWS environment of the architecture.

7.3 Inferences on the ontological framework

The middleware described above allows the definition of systems that provide several functionalities, such as:

- exploiting the ontological framework by offering navigation structure;
- exploring the scenario’s collective views of captured knowledge by deriving additional links and descriptions that are not explicitly given;
- invoking services that match specific user’s situations and needs.

The multi-viewpoint structure provided by CLEO – introduced in Section 5.1 – may represent the navigation structure of the system. The **life event** and its multiple **viewpoint descriptions**, are useful access points to the information and services provided by the system. The relations among the several descriptions, and the rich axiomatization that characterize CLEO enable the exploration of the scenario and inference of new knowledge. Finally, the descriptions of the Service Ontology – properly identified through the context description – allow the invocation, mediation, composition, and execution of Web services.

In the rest of this section, we consider the **moving house** life event and two use cases connected to it for highlighting the advantages of using our ontological framework.

In the first use case, the user seeks and then instantiates the description of his/her situation in order to receive the adequate services; we explain the inferences provided by the ontological framework and the main operations of the involved modules. This use case involves the **configuration and service delivery** knowledge levels (user and provider descriptions of the life event) of a service-supply scenario.

In the second one, a PA explores the scenario description in order to make decision about the supply of a new service or change an existing one. This description involves the **guidelines** knowledge level (politician and manager descriptions of the life event) of a service-supply scenario.

7.3.1 Moving House Life Event

A citizen has to deal with a number of tasks when moves house from an Italian municipality to another one. In addition to the arrangement of the physical re-location of belongings, the citizen has to interact with various agencies to arrange for the daily necessities such as water and electricity. Different institutions and companies need to be informed about change of address. Personal documents (ID card, passport, driving licence, etc.) must be changed or even newly obtained.

Currently, the citizen has to go to the registry office of the destination municipality and notify the change of address. The destination municipality informs the registry office of the provenience
municipality. Separately, the citizen has to notify the same event to the other agencies, often providing the same information.

The description of the scenario by means of our ontological framework allows the identification of all of the involved actors, relationships, and exchanged resources. Moreover, each actor (e.g. the public administrations) can describe the processes and actions that should perform in order to accomplish its tasks. These knowledge is used and transferred by the system in order to accomplish its functionalities and satisfies user’s requests. For instance, if we consider the exchange of information between the citizen and the municipalities, the citizen may simply declare his/her data, and the provenience and destination municipalities. Based on the represented knowledge, a service automatically interacts with the destination municipality to create a new citizen record, and delivers the same information to the provenience municipality, where another service will update the existing citizen record. Note that this procedure tightly follows the rules provided by the Italian legislation. Also these knowledge is represented, and future changes in the laws may influence the evolution (change) of the procedure.

Moreover, the citizen usually does not know what he/she should do after moving in the new house, and which agencies notifies. The system may drive the citizen to accomplish all the necessary tasks, and eventually manage complex interaction between the citizen and a service provider. For instance, drawing up a new contract with an electric power company involves the research of the best company matching some user parameters, and then an active exchange of information between the two parties following a specific protocol (usually defined by the company). These tasks cannot be accomplished in a single step (one shot service), because they require the citizen participation (non-deterministic choices); however, they can be managed by representing the composition of several one shot services.

7.3.2 Use Case 1: Invoking a service

In the first phase, there is a set of stages, which serves as a means for the user to identify, explore, instantiate, and orchestrate a service. This planning phase represents the informative part of the service before its actual execution that starts with its invocation. This aspect is well mapped by the clean separation between context and service delivery description introduced by our ontological framework (Section 4.4).

In this way, the user becomes able to drastically cut down the overall time for getting all the relevant information and to manage complicated services without any external help. More specifically, in this informative phase, the first step is to support him/her in finding and identifying the correct service, and then explore and instantiate the service to his/her specific needs without needing professional help.

Figure 7.2 shows a generic path within our ontological framework that the user may follow in this phase. We use this path as a guide of the present narrative description.

In the moving house life event, the user of the system is a citizen that should notify his/her change of residence. The front end of the system – through the application and reasoner modules of the middleware – offers him/her the possibility to navigate within a taxonomy of life events represented by the class life event of CLEO. In this case, the user follows the following path: Citizen-Related-life-event, Having-a-House-Related-life-event, Moving-House-Life-Event.

The system provides the descriptions, documents, useful links, and generally information objects associated with such a life event. In particular, it allows the observation of the life event from three distinct viewpoint: citizen (user), municipality (provider), and electric power company (provider). As we are discussing the user perspective, the first option is the default.

The user explores the knowledge associate with the citizen viewpoint.

- **Initial state of affairs.** The user identifies the main actors and resources involved in the life event: the applicant, provenience and destination municipalities, old and new address, and the moving date. A second state of affairs identifies the same entities, but adds the family group entity. These two states of affairs maps two possible situations associated with the life event: the applicant moves alone or with his/her family. Relations link the involved entities.
in order to depict the two situations. In the following box, we report only the part of the second state of affairs definition that depicts the situation. Note that in the initial state of affairs, the old address specializes the citizen applicant’s information.

**Definition 7.1: Relations of a initial state of affairs definition**

\[
\text{(and } \text{ (moves-with-family ?citizen-applicant } \\
\text{ ?citizen-family-group)} \text{ )} \\
\text{ (cleo-specializes ?citizen-old-address } \\
\text{ ?citizen-applicant))} \\
\text{ (cleo-supplies ?citizen-applicant } \\
\text{ ?from-municipality))} \\
\text{ (cleo-supplies ?citizen-applicant } \\
\text{ ?to-municipality))} \\
\text{ (cleo-supplies ?citizen-applicant } \\
\text{ ?citizen-new-address))} \\
\text{ (cleo-specifies ?citizen-applicant } \\
\text{ ?moving-date))}
\]

Finally, another initial state of affair describes the situation before asking a new electric power contract. The user identifies the applicant, the locality, the type of house, the type of contract, and the cost for hour.

- **Final state of affairs.** The user identifies the results of the change of residence: the applicant – and eventually also his/her family members – has a new address and receives the list of agencies that have been notified by the system. Moreover, the final state of affair description contains the results of the subscription of a new electric power contract: the confirmation of the activation. In the following box, we report only the part of the second state of affairs definition that depicts the situation. Note that in the final state of affairs, the new address specializes the family members’ information.
Definition 7.2: Relations of a final state of affairs definition

\[
\text{(and (moves-with-family ?citizen-applicant ?citizen-family-group)}
\text{(forall (?citizen-family-member)
  \text{(and (member-of ?citizen-family-member ?citizen-family-group)
    (cleo-specializes ?citizen-family-member ?citizen-new-address)))}
\text{(received ?citizen-applicant ?electric-power-company ?contract-activation)}
\text{(received ?citizen-applicant ?to-municipality ?citizen-list-notifications))}
\]

- **Needs.** The user identifies which of his/her needs the system can satisfy: notifying the change of residence to the two involved municipalities, and drawing up a new contract with an electric power company.

- **Plans.** The user identifies the main steps to follow for achieving the depicted final state of affair. A plan describes the sequence of the needs introduced above: (i) notifying the change of residence; (ii) drawing up the electric power contract.

This exploration phase helps the user to first recognise and then match his/her particular situation. Note that the constraints of the interaction description module (Section 5.1.3, Interaction Description Module) make sure that all of the involved knowledge is represented in the state of affair description.

In our case, the citizen considers the initial state of affairs where the applicant moves with his/her family. At this point, the systems offers two possibilities:

- (a) carrying on the exploration of the ontological framework in order to select a specific need, choose the adequate offer seeking among the existing interaction descriptions, and directly invoke the associated goals by supplying the requested inputs. The core of this approach is the choice of the interaction description that represents the transition events between the two parties, and thus the goals associated with each transition (Section 5.1.3, Modeling interactions between viewpoints). This is a useful option for expert users that know a priori which need/goals they want to invoke.

- (b) instantiating the chosen initial state of affair description with the user data for representing the user specific situation, and thus setting the system to automatically provide the best services that match the user’s needs. Starting from the defined instances, the system manages the interaction between the user and the service providers. Differently from the case (a), here the user does not know a priori goals, interactions, offers, and services. The system will guide the user through a (usually) complicated set of alternatives on the basis of the existing knowledge representations, instance of initial state of affair, and eventually further required inputs (e.g. in complex interactions, the system may require user’s choices on the basis of service results).

It is important to note that in both the cases described above, the system hides the heterogeneity and complexity of the requested services. In fact, user deals with needs and goals (we do not speak of services), and the used vocabulary (terms, concepts, language, etc.) is always the one adopted for the viewpoint description.

For the rest of the use case description, we focus on the option (b). Thus, the citizen creates an instance of the initial state of affair and its descriptive entities by selecting elements of the Domain Ontology. He/she specifies the applicant, family group, provenience and destination municipalities,
old and new address, and moving date. Note that some of the information can be automatically retrieved by the system; for instance the family group and the old address instances can be inferred when the applicant one is defined. Moreover, generally not all of the descriptive entities of a state of affair can be specified by the user at the initial stage; some of them may be provided by the system between the executions of two services. For instance, the type of contract and cost information, in the initial state of affairs associated with the drawing up a new electric power contract, can be chosen by the user only after receiving the list of possible options provided by the electric power company.

From now on, the citizen interacts – through the front end and the application module – with the WfMS of the Middleware that orchestrates the interactions between him/her and the SWS environment (IRS-III). At this point, we pass from the informative part to the service delivery part. The WfMS “knows” the internal logic and the details of the execution of a workflow, because refers to the ontological framework through the reasoner module. It “moves” the execution of all the plans defined within a viewpoint description (Section 5.1.3, Plan Module). It is important to mention that the WfMS does not manage the orchestration of the web service descriptions of the Service Ontology.

Figure 7.3 shows the first step of the service delivery phase. The WfMS receives in input the instance of the state of affair description created by the citizen, and queries the ontological framework in order to load the plan of needs associated with such a state of affair description.

Once the plan is loaded, it can be instantiated and then executed. The plan instance is enriched with further information after each step of the plan; i.e. the instance of the just processed task is linked to the instance of the plan. In this way, the plan can be suspended – and all of the instances connected to it are stored –, and then reactivated starting from the last task instance created.

In our case study, the WfMS loads the sequence of two needs introduced above. The output of this phase will be the instance of the associated final state of affair.

Each need description defines a decomposition into one or more goals. For instance, notifying the change of residence involves only one goal that invokes the communication of the new address, while drawing up the electric power contract involves two goals that drive a transaction: the first goal supplies a list of contracts that matches the user’s requirements (e.g. locality, type of house, etc.), the second one notifies the user choice. Moreover, for the second need, more than one interaction can be considered, because distinct electric power companies provide offers.

These decompositions are represented by means of plans that the WfMS loads from the ontological framework, when a need description is processed. Figure 7.4 shows a generic plan of goals.
loaded by the WfMS\(^1\).

![Diagram of WfMS and IRS-III interactions](image)

Figure 7.4: The WfMS loads the goals plan description and interacts with IRS-III.

At this point, the WfMS processes goal invocations. Before invoking the goal, the WfMS gathers the goal inputs from either the current instance of the initial state of affair or the user.

The goal invocation and the subsequent discovery, composition, mediation, and execution of web services are managed by the IRS-III module. All the information needed by IRS-III are defined in the Service and Domain ontologies. IRS-III discovers existing web services that match the invoked goal on the basis of the WG-mediator descriptions associated with the goal. Among all of the discovered web services, IRS-III checks the defined capability expressions in order to select the most useful service.

We first consider the notification of the change of residence. The plan is composed by a unique goal. The inputs of the goal are the citizen applicant, the family group, the new address, the provenience and destination municipalities, and the moving date. Thus, they can be automatically derived from the instance of the initial state of affair description. The web service selection is based on the destination municipality. The correspondent WSMO service is a complex service, and it is decomposed into two sub-goals (delegation). The first one requires the creation of a new record for the citizen applicant and each member of his/her family in the legacy system of the destination municipality; the second one communicates the change of residence to the provenience municipality. The associated web services are sequentially executed and their results composed to create the list of notifications excepted by the user. In the transition of the change of residence information from the destination to the provenience municipalities, an OO-mediator is defined for solving a mismatch in the description of the moving date format. It maps the format dd/mm/yyyy used in the destination to the format dd/mm/yy, used in the provenience municipality. Note that the orchestration of the Web services is managed by IRS-III and does not require user interaction.

After the Web service executions, IRS-III answers to the WfMS with the adequate output instance that updates the current instance of the initial or final state of affair descriptions. The control is returned to the WfMS that shows the results to the citizen, and eventually waits his/her acknowledgment or a specific choice for carrying on the goal workflow process.

The second step regards the transaction for activating a new electric power contract. The WfMS loads the correspondent plan, which is composed by a sequence of two goals. The inputs for the first goal are already available in the instance of the initial state of affair description, while the inputs for the second goal are provided by the user only after the execution of the Web service associate with the first goal. Here, the WfMS interacts with the user between the two goals for

---

\(^1\)In this use case, we assume that all of the goals of the plan are – or map to – WSMO-goal descriptions of the Service Ontology (Section 5.3.1).
showing the results of the first one, and requiring a choice that is communicated to the electric power company by means of the second one. In this case, the orchestration is managed by the WfMS, while IRS-III manages simple services.

Summarizing, in this use case, the knowledge retained by means of the ontological framework have been used for identifying a user situation, representing complex interactions, and driving the execution of multiple services that transfer knowledge among user and distinct organizations. Beside this, it is important to note that all of the operations connected to the IRS-III module, and thus the SWS environment, are executed without user interaction, and involve mediation and composition among heterogeneous domains. Contrariwise the operation connected to the WfMS module may require the user interaction and use always the same vocabulary (the one defined by the user). Finally, all of the instances created along this process may be stored and reused for future user’s accesses or for specific instance analysis (e.g. logging, data mining based on instance reasoning, etc.).

7.3.3 Use Case 2: exploring the scenario description

The first phase of this use case is composed by the identify and explore stages defined in the previous case. Thus, we do not detail such a description, but simply introduce the differences between the two cases.

Figure 7.5 shows the generic path within the ontological framework that will guide our description.

![Figure 7.5: A generic inference path in CLEO from the provider viewpoint.](image)

The user of the system is a municipality (i.e. municipality worker or domain expert) that accesses the system in order to investigate the possibility of introducing a new service or simply modifying an existing one. In this case, the represented knowledge is used as the base for a decision making process. Before designing the process, all of the aspects associated with a particular offer are valued: the reference legislations and policies, citizens’ needs, and eventually offers of other organizations.

The municipality selects the moving house life event and obtains the associated description provided by its viewpoint.

- **Initial state of affairs.** The municipality identifies the necessary resources for providing the change of residence service: the citizen information, the address, the provenience municipality, and the moving date.

- **Final state of affairs.** The municipality identifies what should be provided to the citizen that applies for the change of residence: the list of which agencies have been notified. Currently such a list contains only the provenience and destination municipalities.

- **Offers.** The municipality identifies which are its offers and services within the considered life event: there is a unique offer that concerns the change of residence, and it is composed by two services; the first one is the complex service – introduced in the previous use case – that
manages a request of change of residence by composing two web service; the second one is the service that updates the legacy systems about citizen record modifications.

- **Plans.** The municipality identifies the structure of the offer, but overall the decomposition of the complex service that manages the change of residence.

However, more interesting knowledge for the municipality may be provided by the context description that comprises its viewpoint. The offer description is the starting point for several inference paths based on the following relations: *influences* and *influenced by*. These relations link the conception descriptions of the different viewpoints (Section 5.1). In particular, it is possible the inference of:

- the *user’s needs* that describe the goals requested by a user, and hence the inputs, outputs, functional and non-functional requirements. In our use case, the user has a unique need connected to the change of residence: notifying all of the involved agencies at the same time.

- the *legislation* that regulates the scenario; these knowledge bound the range of the possible services by stating what can be realized. In our use case, the legislation defines that the moving citizen has to notify the destination municipalities within 30 days, and latter will notify the provenience municipality. Moreover, it defines the list of other agencies that should be notified.

- the *policies and strategies* provided by the managers; these knowledge specify particular aspect that the service should take into consideration, such as policies of cooperation among distinct organizations that lead to the creation of common services.

- the *offers* supplied by other organizations for satisfying the same user need; these knowledge may suggest the structure of the business process of the service.

As a result, starting from its viewpoint, a municipality can obtain a clear description of the context by considering several aspects.

Moreover, further knowledge can be inferred by investigating changes in the situation described above. For instance, the definition of a new norm that requires to keep alignment between the legacy systems of the registry office and cadastre may influence the policies that regulates the cooperation between the two agencies. Thus, the system notifies to the managers the exact policies that should be modified, on the basis of the existing relations. Thereby, the managers change the policy of cooperation between the registry office and cadastre by introducing a strategy of renovation of all of the services based on the legacy systems that should be aligned. The strategy clearly specifies the offers – and their specific services – that are influenced by the change. The system may notify to the agencies worker the service that should be updated, referring to the directives imposed by the strategy. In our use case, on the basis of the changes described above, a municipality can modify the service that manages the change of residence notification by composing a new web service. The latter notifies to the cadastre the citizen change of residence. In this way, when the service is executed, both the registry office and cadastre legacy systems are updated, matching the alignment request of the norm.
In this thesis, we have proposed the application of Knowledge Management (KM) techniques within service-oriented systems in order to supply add-value e-Government services. In particular, we provide a semantic-based framework with which most Public Administrations (PA’s) can identify, from which they can work when designing and delivering e-Government services. This general framework can be adapted and applied as appropriate.

The directions investigated in this thesis involve all of the phases that describe the flow of knowledge within an organization [7] [95]: creation, retention, use, and transfer.

We start from the definition of the structure for the knowledge retention; i.e. knowledge representation. We focus on the knowledge modelling techniques and their application for the semantic representation of services. In particular, we deepen the concepts associated with the development of ontologies (Chapter 1) and the emerging technology of Semantic Web Services (SWS’s) (Chapter 2). These two reviews allow us to become familiar with the most central concepts for building a semantic-based framework.

Then, we compare the existing approaches that adopt such technologies in the e-Government field, in order to identify their relevant aspects and shortcomings (Chapter 3). They face more or less the same problems: there is no generic domain analysis for the overall PA system at any level of granularity; there are no generic PA models for processes and objects; there are no ontologies for modelling PA objects and relationships; there are no standard vocabularies for describing concepts. Moreover, they usually address specific service-oriented models, where the provider’s point of view plays the central role. However, the e-Government scenario is composed by several actors. Each of them deals with different kind of knowledge, conceptions, processes; in other words they have different viewpoints. Such viewpoints differently influence and relate to the service.

This analysis is the base for the definition of the objectives (Section 3.4) and requirements (Section 4.1) of our approach. This is the first step of our conceptual modelling process. We follow a double stages process that first creates a conceptualization of the reality in terms of conceptual models, and then uses ontologies for representing the semantic structure of involved knowledge (Chapter 4).

The e-Government domain is complex. It cannot be described by a unique conceptual model (Section 4.2). We refer to several conceptual models that (i) outline the elements of a government service-supply scenario, (ii) specify the actors and roles of an e-Government application, (iii) introduce the life event metaphor for enabling a multi-viewpoint approach, and (iv) define two distinct aspects of e-Government business processes: planning and interacting.

In our work, we encapsulate all of the above features in distinct independent modules that can be combined and configured as appropriate. These modules are part of the meta-ontologies of our framework.

The abstract views captured by conceptual models above mentioned are organized in three distinct levels:

- **Guidelines**, describing the context that bounds the creation and evolution of services: legislations, policies, and strategies influencing the development and management of an e-Government service-supply scenario.

- **Configuration**, describing the context in which services are supplied: requirements, resources, actor’s role, business processes, and transactions of an e-Government service-supply scenario.

- **Service delivery**, describing the discovery, composition, mediation, and execution of (Web) services.
This novel partition reflects three distinct moments of real-life service-supply e-Government scenarios. They influence each other, but can be described independently. This enables us to segment the description of the scenario on the basis of the actor’s competences: service user and provider describe the configuration of the scenario; manager and politician describe the guidelines, and the application developer describes the service delivery.

Moreover, we introduce two other important separations based on the kind of knowledge: Context vs Services and Context vs Vocabulary. Context represents domain-independent knowledge that describe situations, topics, plans, beliefs, etc. Vocabulary represents domain-dependent knowledge that is linked to a specific description of the context. Services represents technical knowledge associates with the service description. Note that the context description is the key of our approach; for instance (i) two terms of distinct vocabularies may refer to the same description of the context as well as multiple vocabularies may be involved in the description of the same scenario; (ii) the context represents the environment where the services are used, and thus may provide the requirements for the description of services.

The introduced epistemology of the e-Government service-supply scenario has driven the design and development of the proposed ontological framework. To map such a semantic layer, we adopt a meta-modelling approach (Section 4.3). In this way, the resulted specifications are reusable meta-ontologies that can be applied to several e-Government applications. In fact, they describe the global, uniform view of the scenario by using commonly accepted and standardized concepts and properties. This approach is the base for the cooperative and distributed development of specific applications. It allows involved actors to keep their autonomy in the description of their domains: they simply follow the described schema for creating ontologies extending and adapting the meta-ontologies.

As a result, we define an architecture composed by the following three meta-ontologies:

- **Core Life Event Ontology (CLEO)** is the heart of our framework by allowing the description of the configuration and guidelines knowledge of an e-Government service-supply scenario by four classes actors: end-user, domain expert (provider), manager, and politician. It represents the contextualization of the scenario.

- The **Service Ontology** allows the description of the service delivery knowledge. Application developers provide SWS descriptions by addressing integration and interoperability issues. This ontology plays a double role: (i) completing the framework with the description of e-Government processes and services that are implemented by Web services; (ii) solving mismatch problems by introducing mapping mechanisms between two heterogeneous viewpoints.

- The **Domain Ontology** defines the vocabularies used by different actors in the description of their viewpoints. We designed a structure that resides on two levels of abstraction: generic and specific level. *Generic level* represents commonly accepted and standardized concepts and properties of the e-Government domain; *specific level* extends the generic level within the specific domain of an involved actor or PA, ending and adapting the existing concepts and relations. Actually, each instance ontology describes the vocabulary of the respective actor or organization. Every actor or organization is responsible of its specific level ontology.

We use OCML as knowledge modelling tool (Section 4.3.2), and its associated Base Ontology as foundation of our ontology development. We adopt DOLCE – extended by its module Descriptions & Situations – as upper ontology, and WSMO as meta-ontology for SWS’s (Section 4.3.1). Starting from such a semantic layer, we formalized (Chapter 5) the ontological framework introduced above. As main result of our work, we describe the Core Life Event Ontology (CLEO) that provides the minimal concepts and relations required to describe aspects connected to e-Government service-supply scenarios. Its descriptions represent independent views by the various actors involved and may significantly differ in the notions that are used and the granularity of the description (high-level tasks vs. detailed processes). Similarities among such views are to be found on the level of constructs used to describe these views: both of them discuss roles and attributes, parameters, and course of events that can be respectively played by, valued by, and sequences a number of
objects of the Domain Ontology. The objective of this ontology is to enable e-Government actors to represent the knowledge they want to describe with the scenario in their own “language” and based on concepts which are familiar to them, excluding the use of technical concepts. The knowledge may include generic service processes as well as situation descriptions that involve human and not-human agents. CLEO has been designed to be modular: it contains the modules that enclose the features of the conceptual models considered in our work. Every module can be readily extended and freely reused.

One of the most important modules of CLEO allows the description of the interactions between e-Government actors (Chapter 5, Modeling interactions between viewpoints). The latter is a too often neglected component of business process. In our work, we extend an existing framework for designing multi-agent scenario based on economical transactions by adapting it to describe generic interactions between heterogeneous viewpoints. The interaction modules is tightly connected to the conception description module. The latter introduces a mechanism based on the need/goal and offer/service descriptions for modelling knowledge at different level of granularity, fitting project-specific requirements, and representing complex interactions that cannot be represented by the current one-shot SWS approaches. This mechanism does not take the place, but takes advantage of SWS technology. In particular, the goal and service descriptions are the two gates that link the context to the Service Ontology (Chapter 5.3). As a result, non-software agents (e.g. citizen) may interact with the multi-agent environment provide by SWS technology in order to achieve their tasks.

After the definition of the structure for the knowledge retention, we describe two applications of the obtained ontological framework. Such applications provide the evaluation of the contributes of our proposal.

The first one involves the use and transfer of the described knowledge (Chapter 7). We define a high-level architecture of a semantically-enhanced middleware that interfaces e-Government Web applications with Web services. The core of the middleware is our ontological framework (Chapter 7). Without providing technical details, we describe two use cases of the outlined system, in order to evaluate the reasoning capabilities of our ontological framework. The descriptions are based on a simple but very common life event example: moving house. The first use case involves the configuration and service delivery knowledge levels and highlights the knowledge structures and inference paths that are the basis of user-provider, and provider-provider interactions. The second one involves the guidelines and configuration knowledge levels and shows the inference paths within the CLEO description. In particular, it investigates which knowledge can be inferred after some changes in the situation description.

The second application takes advantage of the meta-models that compose our ontological framework for defining a knowledge elicitation methodology (Chapter 6). To describe the methodology and the involved meta-models, we worked out a case study within the change of circumstances scenario. Such a case study provides an evaluation of the applicability to real-life scenarios of our ontological framework. It highlights (i) the cooperative and distributed work of PA’s; (ii) the structure of the proposed modules; (iii) how domain experts can create a full description of a specific e-Government context by using models close to their experience and specific languages of the multiple involved domains, and then application developers can implement SWS descriptions inferring requirements from the context description; (iv) the rigour check imposed by the axiomatization defined in CLEO, ensuring that fundamental concepts are captured, described, and understood.

Finally, our work contributes to KM research in the e-Government field both from an analytical and an engineering perspective. Analytically, it provides both a novel integration of the various conceptual models that have been developed for describing the several e-Government aspects and an insight into the various knowledge-based operations that are performed in Knowledge Management Systems. From an engineering perspective, our framework offers comprehensive support for the rapid construction of e-Government applications in different domains.
Future Work

The future research directions mainly regard the following aspects:

- **Development of the infrastructure for the semantic interoperability.** Software modules are used to implement the functionalities of the semantically-enhanced middleware system that enables the automated interpretation and paves a common ground for services. In Chapter 7, we simply describe the architecture and main functionalities of such software modules. Future work may regard the development of the middleware system and a knowledge management system that adopts it.

- **Further development of the ontological framework for mapping the guidelines knowledge level.** This knowledge level is composed by the politician and manager viewpoints, and drives the evolution of the scenario description. A change in an element of the guidelines knowledge level may produce changes in the other elements of the scenario. An example is provided in the second use case of Section 7.3. Changes can be propagated following the chain created by the influences relations that links the CLEO elements. In this thesis, we do not detail this aspect, but we create the basis for its development. Future work may regard extending the knowledge structure of the Politician and Manager viewpoints, and creating the appropriate infrastructure for tracing the changes and suggesting the evolution of the configuration knowledge.

- **Social and organizational aspects connected to the adoption of the proposed framework.** For instance, the ontologies offer a clear advantage in the (cooperative) description of a scenario, how can we envisage some ways to help domain experts with dealing with ontologies? Future work may regard the development of appropriate tools that allow the development of application-specific conceptual models starting from our ontological framework. Such tools will respect the structure of the framework, and the elicitation methodology defined in this thesis.
Bibliography


[19] DAML-S Coalition. Damls 0.9 draft release. DAML-S Coalition. Website: http://www.daml.org/services/damls/0.9/, 2003.


che per la valutazione di un sistema per la gestione della conoscenza. In \textit{LIUC}. Cattaneo University, 2003.


