Ontology-Driven development of Intelligent Documents*

Flavio Corradini, Francesco De Angelis, Romeo Pruno, and Alberto Polzonetti

Dipartimento di Matematica e Informatica,
Università di Camerino,
Via Madonna delle Carceri, 9,
62032, Camerino (MC), Italy
{name.surname}@unicam.it

Abstract. Interoperability and applicative cooperation are key concepts when information systems compete towards a proper collaborative Public Administration. We believe that a suitable solution to these concepts cannot prescind by a formal conceptualization of the application domain of interest.
In this paper we concentrate on the concept of Intelligent Document and propose a model-driven document management for Public Administrations that is based on ontologies to describe both the structure of the documents, type of data, information and process needed for their filling. The proposed infrastructure supports the entire document life-cycle.

1 Introduction

In e-Government, interoperability and applicative cooperation compete for the development of a proper:

- back-offices process integration for service provisioning, and
- distribution of services with suitable quality attributes.

In such a regime, Public Administrations (PAs) increase their shared knowledge [17], stimulating the realization of infrastructures that allow applications to suitably cooperate. Such cooperation would certainly benefit by a formal representation of the reality of interest.

In this paper we concentrate on document management and propose a model-driven document management for Public Administrations that is based on ontologies [6] to describe and analyze the relations between concepts [19].

The importance of ontologies to describe the structure of a document can be found in [10]. An ontological description of electronic documents has been explored in [4] for the so-called "Intelligent Document" that describes both the document structure and the procedural iter for its filling. Within an intelligent document we can identify different parts - different zones - the compilation of which is responsibility of different PAs by means of suitable workflows.

* This work has been supported by Halley Informatica and Regione Marche.

We use ontologies also to describe the type of data and information within the different zones. This provides a strong support for filling documents but also for their analysis and generation of suitable interfaces or documents form.

We also discuss how our ontology-based documents formalization supports the entire life cycle of the document: from the user interface creation to the management of the document compilation and its storage. From the ontology, indeed, we can retrieve an XForm that represents the user interface of the document. We can also manage the iter of compilation using the information contained in the process model of the document and document persistency is also managed by means of an ontology management system that maps the document ontology to a relational database.

Besides providing a semantic management of electronic documents, the general ontology-based framework we discuss in the domain of Public Administrations, provides a solid support to automatic administrative procedures and allows information retrieval and knowledge management according to the demands of a single public administration.

The rest of this paper is organized as follows. Section 2 discusses administrative and semantic cooperation among PAs. Section 3 introduces the concept of intelligent document and its main features while Section 4 shows an example of document ontology and its relation to intelligent document ontology. In Section 5 we present an architecture of an ontology-based information system for intelligent document delivery that use the ontologies previously introduced. We end with conclusion in Section 6.

2 Administrative and semantic cooperation

We use ontologies because it defines a set of the categories of certain entities that exist (or could exist) in an application domain. Indeed, one of the well-known definitions is by Tom Gruber [6]: "An ontology is an explicit specification of a conceptualization", but it has also been defined in [16] as "a catalog of the types of an entity of which its existence is assumed in a domain D in perspective of its access from an actor that uses a language L to enter or to communicate with D". The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. From a more applicative point of view what exists is that which can be represented. In fact, an ontology allows to define the meaning of the concepts that belong to a specific domain establishing the relationships between entities (classes, relations, etc.). To do this it is necessary a language (e.g. RDF) for ontology representation.

In particular, ontologies allow the formalization of a model of the world (or a part of it) that includes concepts and relations [2]. It is also worth of introducing the concepts of Informal Ontology and Formal Ontology. The former, by using a natural language description, simply describes the defined and indefinite entities of a given catalog of types. The latter, instead, arranges names in concepts and types of relationships, and is organized in classes and sub-classes of types.

The first step for ontology construction is the creation of a reference dictionary (informal ontology, a list of keywords that describes the content of the document). After the creation of this catalog we compose a standard representation of the information. In these years many standards have been proposed to describe ontologies and the most known languages are RDF (Resource Description Framework) [12] and OWL (Ontology Web language) [18]; both based on XML.

Using an infrastructure based on ontologies, namely an ontology-driven information system, we are able to overcome various PAs problems such as:

- semantic management of documents;
- scalability and, in particular, dynamic specialization according to the real needs of the Public Administration;
- knowledge access and knowledge sharing;
- integration of the existing technology.

3 Intelligent document definition

More and more organizations use electronic documents for process management. Marketing analysis have shown a cost-effective use of such electronic documents in place of standard document management. In several cases, costs can even be eliminated in favor of a more efficient data retrieval from the actors of the automated process.

PAs can take advantage of the use of electronic documents because of a (i) consistent decreasing of the costs of press, elaboration, distribution, delivery, etc., (ii) consistent decreasing of compilation errors, (iii) possibility to share and/or to access the shared information. Citizens and companies, as end-users, can take advantage of the use of electronic documents because of a (i) consistent decreasing of the time needed for manual compilation, (ii) electronic dispatch of the documents, (iii) decrease of the delivery costs (postal, fax, etc). This short and certainly incomplete analysis shows the necessity to create a process of workflow management that contemplates the fruition of the electronic document inside the application domain. The approach that has been used till now is not enough to answer our questions, we don't have to think to an Intelligent Document as a simple form but as the result of an orchestration of services offered by the community of PA's that participates in its filling.

With this idea in mind we describe the main characteristics of an Intelligent Document (we refer the reader to [4] for more details), namely an "e-Document for the Public Administration". An intelligent document must be:

- Multi Composed: it contains information on both its state and the application domain. Different processes and users may compete to its creation;
- Multi Version: the tracking of data is guaranteed by the persistence of the various documents versions. Every office of the involved Public Administrations compiles the part (zone) of the document of its competence;

– Multi Dependence: it consists in the creation of the electronic document from a collection of other documents via a modular application-domain dependent architecture;
– Multi Data Type: different data types can be considered within the document;
– Multi Platform: the compilation and use of a document have to assure the interoperability at the application level;

The attribute Multi Dependence is particular significant for our discussion. It automatizes the standard procedure of a document creation in the Public Administration where different office may compete for a document filling. For example, a "Birth Certificate" needs information by both the registry office of the municipality where one person was born and by the residence office. Of course, the municipality where one person was born is different from where he/she lives. The compilation of this typology of documents may require, therefore, external actors to a specific Public Administration [1].

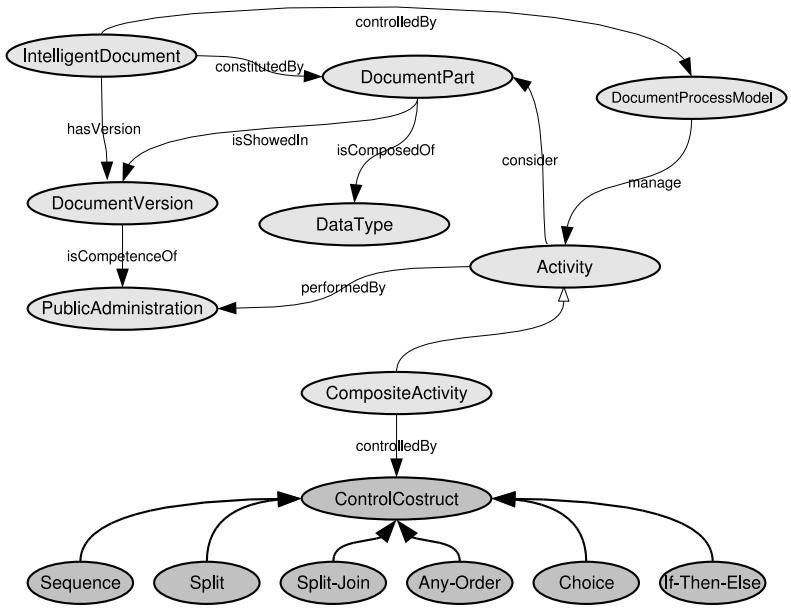

Fig. 1. Intelligent document ontology

These concepts are shown in Figure 1. We describe a simple ontology that represents the concept Intelligent Document. This ontology can be merged with the ontology describing the structure of a document - a document-specific ontology - in order to detect the forming parts of the document itself. In fact, an Intelligent Document can be divided in several parts shown in different versions/views.

Each view has a version identifier and each PA involved in the document filling process can modify only the specific part that is of its own competence. The filling process is itself part of the intelligent document definition. It performs activities (represented by the *Activity* concept) that are a couples composed by a *DocumentPart* and a *PublicAdministration*.

4 Document-specific ontology

In order to provide an ontology-based information system able to manage documents of PAs, we need an ontology that describes the concepts and the relations defining the documents themselves.

In this paper we consider, as a case study, a specific document the "Birth Act"; though similar considerations hold for other kinds of documents within Public Administrations. A birth act legalizes a birth [3]. The iter in the italian legislation starts with a "birth declaration"(also called "birth denunciation") carried out by a parent or a delegated. This document must be presented by ten days to the "Stato Civile" office in the municipality in which the birth was happened. However, the hospital can start this iter, but the birth declaration carried out by a parent or a delegated must be performed by three days after the birth. The birth declaration is mandatory for the "birth act" that is also the bases for other documents such as the "birth certificate". The birth certificate must be delivered to the registry office and to the "imposte dirette"[1] office in order to allow the compilation of the "codice fiscale"[2] that is demanding for the personal "tessera sanitaria"[3].

The needed ontology must contain concepts describing a birth that refer to a child. Moreover, the document is composed by an header and several clauses that represent *DocumentPart* of the intelligent document. This kind of pointers integrate the intelligent document ontology with the document-specific ontology as shown in Figure 2 regarding relation *represent* that binds concepts from document ontology to concept *DocumentPart* of the intelligent document ontology. The reader interested in the whole ontology description of the birth certificate can consult [3].

Though we have presented the key work on a specific document, this is the procedure one has to consider to develop a new document in our framework. We have to describe the reality with an ontology and link concepts to the intelligent document ontology. Of course, we also have to state the activities and how these are handled by PAs. This can be done through the definition of *Activity* concept in the intelligent document ontology.

These ontologies are the models used by our information system in order to build the infrastructure used to perform the procedural iter of a document compilation.

[1] *Imposte dirette office* is an office that manages taxes related documents
[2] *Codice fiscale* is an italian personal code for taxes related documents
[3] *Tessera sanitaria* is an italian document used to sanitary identification of people

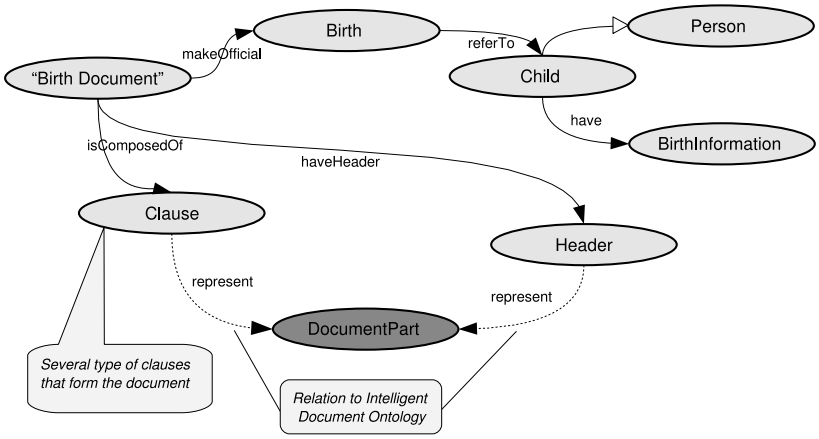

Fig. 2. Document-specific ontology and relations to intelligent document ontology

5 Ontology-driven Intelligent Document Compilation

All forms of engineering rely on models to better understand complex, real-world systems. Ontology-based models provide abstractions from physical world allowing developers to reason about systems at different abstraction levels. We now discuss how ontologies can be used to guide the process of building on-line intelligent documents and to drive the storage of relevant information inherent the document itself.

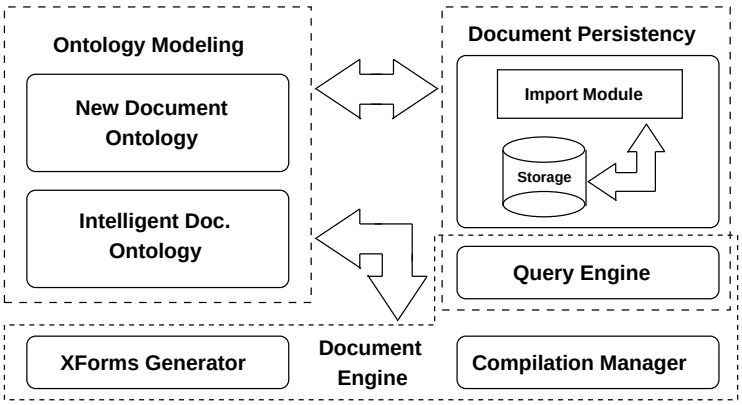

Fig. 3. Architecture of the ontology-based information system for intelligent document delivery

Our approach is based on the architecture shown in Figure 3. We model the document using the intelligent document ontology and the document-specific ontology (describing document related concepts) just like the birth certificate discussed in 4.

This model is used in the *Document Persistency* and in *Document Engine* modules of our architecture. Document persistency module is constituted by an ontology management system that is formed by an import module, a back-end storage and a query engine. The import module is responsible of the mapping from ontology to relational database. The main advantage of this approach instead of classic database is the use in conjunction with a query engine. Querying ontologies can benefit of inference mechanisms and it is possible for the system to answer complex queries using deduction.

Moreover, document engine has two tasks performed by *XForms Generator* and *Compilation Manager*. The first generates forms - user interface of on-line intelligent document - based on ontologies. These forms can be viewed and utilized with a browser. Using a formal representation of concepts and related data types we can assembly forms considering various constraint on controls inside it. The *Compilation Manager* module is responsible for the control of the compilation process by taking into account the workflow that has to be performed to fill the document. The workflow is, of course, document-dependent and contains all the information about the procedural iter of the document filling.

For example, assume there is a document with two parts: part 1 and part 2. The compilation of part 1 is responsibility of the Public Administration "A" while the compilation of part 2 is responsibility of the Public Administration "B". Moreover, assume that the filling of part 2 must follow the filling of part 1 because PA "B" wants to check part 1 before its task. This also means that PA "A" starts the document compilation and only when it has finished sends the document version "B". By its side, "B" completes part 2 and then sends the overall document to "A" because "A" has started the whole process.

Of course, this is a very simple kind of cooperation that involves only two PAs. In reality we can find more complex scenarios and in these cases a proper process specification that describe the document compilation iter can be of great benefit.

Summing up, in order to deliver a new kind of document we must specify the ontology that describes the document itself together with its iter of compilation. This ontology is used by the information systems of the involved PAs (those that are involved in the delivering of the document). Each PA generates the forms needed for the document filling and is able to detect the tasks to be performed on the basis of the process model describing the activities. The compilation process is managed by the "Document Engine" of the PA that starts the iter of compilation as orchestration of services of the involved PAs, querying both the local knowledge base and external ones and interacting with users by means of the generated forms.

5.1 A tool prototype

We now briefly discuss the current solutions we are adopting or we would like to adopt for each single component of the architecture. We plan to use IBM Minerva [8] for the *Document Persistency* module. Minerva is an ontology repository built on a relational database that follows OWL and SPARQL Query Language [20] to query the knowledge base. Moreover, Minerva embeds a description logic reasoner in order to provide inference capabilities. The DBMS used for the backend is the HSQLDB database [7] (a small and fast database engine written in java with both in-memory and disk-based tables). Also, for inference we use the Pellet reasoner [13] that is an open-source Java based OWL DL reasoner which provides functionalities like species validation, ontology consistency checking, ontology classification, etc.

On the other side, the *Ontology Modeling* module does not impose a specific solution because ontologies are based on a standard language (OWL). For development purposes we use Protégé Ontology Editor [11].

The crucial component of this system is the *Document Engine*. It deals with forms generation and workflows execution. The form generation, generates XForms [21]. These are significant in several web applications because provide a rich, secure and device/platform independent way of handling web input separating data and logic from presentation. We are studying an automated solution like the one used in the IBM XForms Generator plugin for Eclipse [9] that is able to generate XForms starting from an XML Schema definition. Regarding the compilation manager we plan to use an adaptation of OWL-S API[4][14] from *mindswap* [15] that provides an API for access to read, execute and write OWL-S documents. After a little adaptation this opensource library can be used to read ad write the process model of intelligent documents allowing the construction of an efficient document engine. Alternatively, we can exploit Hermes [5], a workflow engine that we have already developed to support the execution of activity based applications.

6 Conclusion

In this paper we have proposed an ontology-based methodology and infrastructure for the delivery of intelligent documents within Public Administrations. PAs can access a common repository to find ontologies describing a document and set up their information systems to manage the entire life cycle of the document: user interface creation, data storage and management of the iter of compilation.

The models and the architecture that we have discussed in this paper derive from a careful analysis of the E-Government application domain. We deal with the concept of Intelligent Document, a document that is able to describe its structure and to specify the procedural iter for its filling. This can be seen as a result of a cooperation between PAs involved in the delivery. This kind of

[4] The Intelligent Document Process Model is similar to OWL-S Process Model and we can reuse some functionalities of existing libraries

cooperation certainly improves efficiency among PAs but also satisfaction by final users as citizens and companies.

So far, we have only considered a conceptual architecture of the system and started the implementation of a prototype. In the future we plan to face with other important attributes like security of documents and digital signatures that are key concepts for an effective delivery of public services.

References

1. Ahonen, H., Heikken, B., Heinonen, O., Jaakkola, J., Kilpelainen, P., Lindén, G., Mannila, H.: Intelligence Assembly of Structure Document. Technical report, Deparment of Computer Science, University of Helsinky (1996)
2. Boughettaya, A., Elmagarmid, A., Medjahed, B., Ouzzani, M.: Ontology based support for Digital Government. In: Int. Conf. on Very Large Data Bases, London (2001)
3. Corradini, F., De Angelis, F., Polzonetti, A., Brugnoni, E.: Sviluppo di un'ontologia per l'e-Government: un caso di studio per un documento italiano, Cesena, Tecnologie digitali e competitivitá: quale Ricerca, quali Professioni, AICA (2006)
4. Corradini, F., Polzonetti, A., Pruno, R.: E-Government administrative and semantic cooperation: the role of "Intelligent Documents". In: Electronic Government: 4th International Conference, Copenhagen, Denmark, EGOV (Workshops and Posters) 2005 (2005) 150–157
5. Corradini, F., Merelli, E.: Hermes: An agent-based middleware for mobile computing. In: Formal Methods for the Design of Real-Time Systems: Mobile Computing, Springer, 2005.
6. Gruber, T.: A translation approach to portable ontologies. Knowledge Acquisition **5(2)** (1993) 199–220
7. HSQLDB: HSQL Database engine. (*http://www.hsqldb.org/*)
8. IBM: IBM Integrated Ontology Development Toolkit. (*http://www.alphaworks.ibm.com/tech/semanticstk*)
9. IBM: IBM XML Forms Generator. (*http://www.alphaworks.ibm.com/tech/xfg*)
10. Klischewski, R.: Towards an ontology for e-document management in public administration, the case of schleswig-holstein, Hawaii, (36th Hawaii Int. Conf. on System Sciences)
11. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: The Semantic Web - ISWC 2004, Hiroshima, Japan, Third International Semantic Web Conference (2004)
12. Manola, F., Miller, E., McBride, B.: RDF Primer. http://www.w3.org/TR/rdf-primer/ (2004) W3C Recommendation.
13. Mindswap Group: Pellet OWL Reasoner. (*http://www.mindswap.org/2003/pellet/*)
14. Mindswap Group: OWL-S API. (*http://www.mindswap.org/2004/owl-s/api/*)
15. Mindswap Group. (*http://www.mindswap.org/*)
16. Oppermann, H., Schnurr, H.P., Studer, R.: Die bedeutung von ontologien für dasWissens mangement. wissens management **6** (2001) 33–66
17. Pollock, J.T., Hodgson, R.: Adaptive Information. Wiley (2004)
18. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. http://www.w3.org/TR/owl-guide/ (2004) W3C Recommendation.

19. Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA (1999)
20. W3C: SPARQL Query Language for RDF. (*http://www.w3.org/TR/rdf-sparql-query/*)
21. W3C: XForms - The Next Generation of Web Forms. (*http://www.w3.org/MarkUp/Forms/*)