NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

GRADUATE THESIS

Nomothesi@ API: Reengineering the Electronic Platform

Panagiotis Soursos

Supervisors: **Manolis Koubarakis**, Professor
Charalampos Nikolaou, PhD Candidate
Ilias Chalkidis, MSc Student

ATHENS

JUNE 2015

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Νομοθεσί@ API: Αναδιοργάνωση της Ηλεκτρονικής Πλατφόρμας

Παναγιώτης Σούρσος

Επιβλέποντες: **Μανώλης Κουμπαράκης**, Καθηγητής
Χαράλαμπος Νικολάου, Διδακτορικός Φοιτητής
Ηλίας Χαλκίδης, Μεταπτυχιακός Φοιτητής

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2015

GRADUATE THESIS

Nomothesi@ API: Reengineering the Electronic Platform

Panagiotis Soursos
A.M: 1115200800049

Supervisors: Manolis Koubarakis, Professor
Charalampos Nikolaou, PhD Candidate
Ilias Chalkidis, MSc Student

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Νομοθεσί@ API: Αναδιοργάνωση της Ηλεκτρονικής Πλατφόρμας

Παναγιώτης Σούρσος
Α.Μ: 1115200800049

Επιβλέποντες: Μανώλης Κουμπαράκης, Καθηγητής
Χαράλαμπος Νικολάου, Διδακτορικός Φοιτητής
Ηλίας Χαλκίδης, Μεταπτυχιακός Φοιτητής

Abstract

The objective of this thesis is to contribute in legal knowledge's representation and its integration in the area of Open Data in Greece, both from a technological perspective and in terms of transparency. Based on an earlier work in "Nomothesi@: Greek Legislation Platform" (2014) [1], we did not have to start from scratch but we concentrated on expanding, upgrading and developing the above mentioned work. Nomothesi@, is a platform to provide access to Greek Legislation, by means of a legal XML/RDF syntax and linked data. This new version of Nomethesi@ proposes the replacement of the previous XML standard for Greek legal documents to a new RDF one, a new Spring MVC architecture and many REST services such as a SPARQL Endpoint. It still adopts CEN Metalex OWL ontology's core, alongside with other RDF metadata, to describe the legislative relationships and events. Linking data is about interlinking and publishing openly Greek public data and legislative data across EU in order to enhance E-Government. On these fundamentals, we tried to expand Nomothesi@ with a unified RDF Schema, in order to create a RESTful API to serve all the precious semantic information Greek Legislation has to offer and to encourage further and more complex projects based on web services for searching and browsing legislation.

SUBJECT AREA: Semantic Web, AI, Software Engineering

KEYWORDS: Legislative Knowledge Representation, Open Data, E-Government, RDF/OWL Metadata, XML, Spring MVC, Universal Resource Identifiers, REST Services

Περίληψη

Ο στόχος αυτής της εργασίας, είναι να συμβάλει στον τομέα της αναπαράστασης νομικής γνώσης και στην ενσωμάτωση αυτής στην περιοχή των ανοιχτών δεδομένων στην Ελλάδα, τόσο από τεχνολογική σκοπιά, όσο και από άποψη διαφάνειας. Βασιζόμενοι σε μια προγενέστερη εργασία με τίτλο “Νομοθεσί@: Πλατφόρμα για την Ελληνική νομοθεσία” (2014) [1], δεν χρειάζεται να ξεκινήσουμε από μηδενική βάση, αλλά επικεντρωθήκαμε στην επέκταση, την αναβάθμιση και την ανάπτυξη του παραπάνω έργου. Η Νομοθεσί@, είναι μια πλατφόρμα που σκοπό έχει να δώσει πρόσβαση στην ελληνική νομοθεσία, με τη χρήση ενός νομικού XML/RDF προτύπου και με διασυνδεδεμένα δεδομένα (linked data). Αυτή η νέα έκδοση της Νομοθεσίας προτείνει την αντικατάσταση του προηγούμενου προτύπου XML για τα ελληνικά νομικά έγγραφα για ένα νέο RDF, μια νέα Spring MVC αρχιτεκτονική και την παροχή πολλών REST υπηρεσιών όπως αυτή ενός SPARQL Endpoint. Η νέα πλατφόρμα εξακολουθεί να υιοθετεί την OWL οντολογία CEN Metalex, παράλληλα με άλλες, με σκοπό να περιγράψει τις νομοθετικές σχέσεις και τα γεγονότα. Η σύνδεση δεδομένων αφορά τη διασύνδεση και την ανοιχτή δημοσίευση ελληνικών δημόσιων δεδομένων και των νομοθετικών δεδομένων κατά μήκος της Ευρωπαϊκής Ένωσης, με σκοπό την ενίσχυση της ηλεκτρονικής διακυβέρνησης. Πάνω σε αυτές τις αρχές, προσπαθήσαμε να επεκτείνουμε τη Νομοθεσί@ με ένα ενοποιημένο RDF Σχήμα δεδομένων, προκειμένου να δημιουργηθεί ένα RESTful API για να αξιοποιήσει ολόκληρη την πολύτιμη σημασιολογική πληροφορία που έχει να προσφέρει η ελληνική νομοθεσία και να ενθαρρύνει περαιτέρω και πιο πολύπλοκα έργα που βασίζονται στον τομέα του διαδικτύου για την αναζήτηση και την περιήγηση της νομοθεσίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Σημασιολογικός Ιστός, Τεχνητή Νοημοσύνη, Τεχνολογία Λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αναπαράσταση Γνώσης Νομοθεσίας, Ανοιχτά Δεδομένα, Ηλεκτρονική Διακυβέρνηση, RDF/OWL Μεταδεδομένα, XML, Spring MVC, Καθολικά Αναγνωριστικά Πηγής, REST Υπηρεσίες

Acknowledgements

First of all I would like to acknowledge my supervisor Prof. Manolis Koubarakis. His courses Artificial Intelligence I and Artificial Intelligence II had directly contributed in discovering my personal interest in AI and more specifically assisted me in selecting this topic. I would also like to acknowledge PhD candidate Charalampos Nikolaou, for his valuable assistance in some crucial technical points. Finally i would like to acknowledge MSc Student Ilias Chalkidis whose research and previous studies shaped the background and the goal of this project, he assisted and guided me as a second supervisor in the whole process of this thesis. This project is dedicated to all those people around the world, who rise up against fascism.

Contents

1 Introduction **11**
- 1.1 Related Work . . . 11
  - 1.1.1 CEN MetaLex . . . 12
  - 1.1.2 Legislation UK . . . 12
  - 1.1.3 Nomothesi@ Greek Legislation Platform . . . 13
- 1.2 Objectives of the thesis . . . 13
- 1.3 Organization of the thesis . . . 14

2 Background **15**
- 2.1 Greek Legislation Structure - Modifications, a great structural and temporal challenge . . . 15
  - 2.1.1 Greek Legislation structure . . . 15
  - 2.1.2 Modifications . . . 16
- 2.2 The dynamics of RDF Data Models, OWL Ontologies and SPARQL - Semantic Perspectives . . . 17
  - 2.2.1 The new Data Model with Resource Description Framework (RDF) . . . 17
  - 2.2.2 OWL Ontologies and SPARQL . . . 18
  - 2.2.3 Semantic perspectives and future Web . . . 19

3 Nomothesi@ API: Reengineering Nomothesi@ platform **20**
- 3.1 Ontology presentation . . . 20
  - 3.1.1 Representing legal documents . . . 20
  - 3.1.2 Publishing modifications . . . 23
- 3.2 URIs - Rest services . . . 24
- 3.3 Apply Nomothesi@ - Use case . . . 27

4 Nomothesi@ API architecture **32**
- 4.1 Spring MVC Framework . . . 32
- 4.2 Sesame RDF Store Server . . . 34
- 4.3 Strabon RDF Store . . . 34
- 4.4 Use cases and Rest Services . . . 34
  - 4.4.1 Use case 1 - Present enacted legislative version . . . 34
  - 4.4.2 Use Case 2 - Performing a search based on parameters . . . 37
  - 4.4.3 Use Case 3 - Geospatial Data . . . 39
  - 4.4.4 Use Case 4 - Rest Services . . . 40

5 Demonstration of Nomothesi@ Web Platform **42**

6 Conclusions **46**
- 6.1 Future Work . . . 46

A Nomothesi@ Web Application **49**

List of Figures

2.1 Life Cycle of legal documents 16

3.1 CEN Metalex OWL Ontology Core 21
3.2 OWL Ontology for Greek Legislation 22
3.3 P.D. 2011/54 Structure RDF Graph 30
3.4 P.D. 2011/54 - P.D. 2012/10 Interactions RDF Graph 31

4.1 Work-flow of the Spring Web MVC architecture 32
4.2 Spring MVC Framework architecture within Nomothesi@ 33
4.3 UML sequence diagram Use Case 1 36
4.4 Search implementation from *LegislationController* to *LegalDocumentDAOImpl* 38
4.5 *getKML* method in *LegalDocumentDAOImpl* 40

5.1 Home Page of Nomothesi@ 42
5.2 Page of Search Results 43
5.3 HTML Page of Latest version of PD 2011/54 44
5.4 HTML Page of Citations of PD 2011/54 44
5.5 HTML Page of Modifications of PD 2011/54 44
5.6 SPARQL query example in HTML format 45
5.7 SPARQL query example in SPARQL/XML format 45

6.1 Part of Future Greek Linked Data Cloud Diagram 47

List of Tables

3.1 Encoding Types of Legislation 25
3.2 Encoding Versions of Legislation 26
3.3 Encoding Types of File 26
3.4 Query results format extensions 26

4.1 Request driven text 41
4.2 Method calls on REST Services 41

6.1 LIST OF ABBREVIATIONS 48
6.2 TABLE OF TRANSLATIONS 48

Chapter 1

Introduction

In many countries around the world, most lawyers rely on pay-for commercial legal research services. But have you ever thought that law is not written just for lawyers but it is written to help and serve the public and be a useful tool in the hands of people. The people using the government's on-line legislation services are generally not lawyers, but are drawn from a much wider group of people who need to know, cite, or use legislation as part of their job. These can range from police officers, to head teachers, to citizens defending their rights. Our project aims to serve those people who need to know what a legal document states, how it has been modified over the years, or even ask for more complex questions and that encapsulates the role of Nomothesi@.

1.1 Related Work

Having in mind all of the above, many countries in the last decade have focused in transforming legislation and legislative documents into simple-accessed and better-understanding data. Luckily we do not have to invent the wheel each time, because research groups in many countries (Netherlands, United Kingdom, Finland, etc.) have been providing and developing frameworks and platforms, based on different data models, for on-line access to government files and services and especially to our concerns, access to legislation. Legislative documents ought to be produced and archived in information systems, in which one is capable of accessing and retrieving information about them in a large-scale manner with great respect to transparency.

These facts change the way a legal document should be displayed, for it to be applicable on web standards and be friendly to its transformation to an open-up piece of data. XML and RDF are core data models helping us to achieve our goals. The usability and quality of these systems is determined on the basis of opportunities related to the search of legal documents and their structure (e.g. articles, paragraphs etc.) based on criteria (thematic, verbal, chronological), on the representation of chronological versions (e.g. enacted, current, revised), on the publication of metadata and generally capabilities of reprocessing by third party CS engineers [2]. The quality of these capabilities is immediate consequence of the adoption of the new "state of the art" web and knowledge representation technologies and more specifically the semantic web ones.

Some relative noteworthy and on the edge work should be considered first, before we can describe and explain our own.

1.1.1 CEN MetaLex

MetaLex[1] is an Open XML Interchange Format for Legal and Legislative Resources which standardizes the way in which sources of law and references to sources of law are to be represented in XML, intended not to replace jurisdiction-specific standards and vendor-specific formats in the publications process but to impose a standardized view on legal documents for the purposes of information exchange and interoperability in the context of software development [3]. The standard provides a generic and easily extensible framework for the XML encoding by giving us a basic XML Schema adaptable to many languages and legislation formats across EU. It also prescribes the existence of a naming convention for minting URI-based identifiers for all structural elements of a legal document. MetaLex explicitly encourages the use of RDFa attributes on its elements, and provides special metadata-elements for serializing additional RDF triples that cannot be expressed on structural elements themselves. MetaLex includes an ontology, which defines an event model for legislative modifications. In that way MetaLex is a part of the general idea of an integrated semantic web where legal documents are enriched with metadata to enable smart applications such as (intelligent) retrieval and reasoning [4].

- MetaLex enables public administrations to link legal information from various levels of authority and different countries and languages.
- An open interchange protects customers of such companies from vendor lock in.
- Finally, the standard helps to improve transparency and accessibility of legal content for citizens and businesses.

1.1.2 Legislation UK

The legislation.gov.uk portal has partially adopted the MetaLex event model for representing modifications. On the surface, legislation.gov.uk is an attractive website, providing simple and direct access to legislation. People can view whole Acts, or a particular section, in either HTML or in a print version in PDF. An innovating and state of the arts feature of legislation.gov.uk is that they provide legislation in multiple different formats in a RESTful manner. All the data is held in XML, using a native XML database. The application logic is similarly constructed using open standards, in XSLTs and XQuery. The XML conforms to the Crown Legislation Markup Language (CLML) which we can think of as an enriched MetaLex XML Schema and for further metadata enrichment the API is using RDF based on Metalex OWL ontology. Let us here say that legislation.gov.uk was one of the first complete web interfaces across Europe, that used Linked Data and Semantic technologies to communicate legislative work with the public. It is not irrelevant to the fact that UK has continuously been developing and encouraging the use of Linked Data with all kinds of government files and national archives. The UK government is working to publish government data using Linked Data standards as part of work on data.gov.uk [5].

[1] See `ftp://ftp.cen.eu/CEN/Sectors/List/ICT/CWAs/CWA15710-2010-Metalex2.pdf`

1.1.3 Nomothesi@ Greek Legislation Platform

This platform is the very big step from Greece to catch up in the legislation representation field using Linked Data. It is still shaping and making progress as it has some pioneer elements. The first version was developed in terms of providing access to Greek Legislation, by means of a legal XML syntax and linked data. Nomethesi@ proposes a XML standard for Greek legal documents and adopts CEN Metalex OWL ontology's core, alongside with other RDF metadata, to describe the legislative relationships and events.

1.2 Objectives of the thesis

This project forms a part of the general goal of legislative knowledge representation in terms of Linked Data. Especially its main goal is to update and upgrade the technologies used in the previous version of Nomothesi@ by expanding them and establishing a fresh new redesigned framework corresponding to the new era of Semantic Web by dropping the old XML Schema and adopting an extended RDF one based on CEN Metalex Core OWL ontology. Its minor goal is to redesign the web platform to be more efficient, more user friendly and more accurate in terms of searching, querying, accessing the law and mostly publishing legislation in a RESTful manner. We chose to let go of the XML Schema because XML is concerned with serialization which is a way to encode information that when it is passed between machines it can be parsed. We do not want XML fans holding a grudge on us, we still believe that XML, XSLT, XQuery and XPath technologies describe and identify information accurately and unambiguously. This does not mean that XML by itself will revolutionize the web, or that the web is the only field in which XML is going to be useful. XML may have become a common thread uniting a wide variety of applications, smoothly managing data across distributed applications, but our needs and the needs of the web are forcing us to take a calm look and find a way to work with the fruitful knowledge that relates, expands and describes a marked document. Especially when we talk about legal documents whose structure is heavy and complex with lots of metadata shaded between the lines, we cannot lower our needs for serving linked and semantic information over the web.

RDF, in contrast, is a data model, which is liberal set of rules for representing information and it is used to define objects and concepts and relationships between them. The Semantic Web comes in layers. RDF (Resource Description Framework) is the keystone which defines the structure of data. The most important things RDF defines in our project are the several sub-types of a legal document shaped in our demands. RDF, is a graph database. RDFS (RDF Schema), on the other hand, is object oriented in its nature. That is, RDFS is fundamentally about describing classes of objects. This means we can start making statements about classes of a legal document, and types of relationship or describing them giving us a very flexible data model for storing and recording data relations. It also allows us to describe in human readable text the meaning of a relationship or a class. Furthermore it is used to indicate that a class or property is a sub-type of a more general type. We are also using an OWL ontology which adds semantics to the schema. It allows us to specify far more about the properties and classes. Another useful thing OWL adds is the ability to infer additional knowledge based on the rules we imply, we therefor chose OWL to take advantage of infer implicit facts and of its logical – semantic infrastructure. RDF is very useful when we deal with legislation where our main concern is the most efficient modeling and at the same time without losing any of the semantic information that is vital to us (structural, time, modifications). In that way we end up with a faster, more workable and more expansion friendly framework.

We are also serving a SPARQL Endpoint. This service will give the opportunity for everyone to query our data sets and draw legislative knowledge in their own special interests.

An extra goal, and vision at the same time, from this project aims for it to be considered as a RESTful (Representational State Transfer) web platform and to provide keystone ingredients for other and maybe more specialized applications. We have developed and hoping to continue developing services that allow our API users to retrieve Greek Legislation in many forms (PDF, RDF, XML and JSON) from accessing via HTTP GET requests specific URIs (Uniform Resource Identifiers). Linking data has always been the greatest challenge in Semantic Web community, therefore as a first step of this vision we linked our semantic legislative documents to the semantic spatiotemporal RDF store Strabon, designed from our research group in University of Athens. Strabon is built by extending the well-known RDF store Sesame and extends Sesame's components to manage thematic, spatial and temporal data that is stored in the back-end RDBMS and supports spatial data-types enabling the serialization of geometric objects in OGC standards WKT and GML. A legal document may have spatiotemporal information. An example could be a legal act that refers to a specific city of a place. We are invoking this service as part of the semantic information a legal document can include and we hope as time passes to be capable of serving more aspects of these inexhaustible information that stay hemmed in legal documents. It is also important to be able to serve query endpoints to access that data more conveniently. This effort forms the very beginning of making the Web of Data a reality, for new datasets such as the connection between Greek Legislation and special/temporal information. It is important to have the huge amount of data on the web available in a standard format, reachable and manageable by Semantic Web tools. Furthermore, not only does the Semantic Web need access to data, but relationships among data should be made available, too, to create a Web of Data (as opposed to a sheer collection of datasets) [6]. To achieve and create Linked Data, technologies should be available for a common format (RDF), to make either conversion or on-the-fly access to existing databases (relational, XML, HTML, etc.).

1.3 Organization of the thesis

The organization of the thesis is separated in five Chapters. In Chapter 2 we deal with two main themes. The first discusses the background of Greek Legislation and helps us to understand its structure, and the second one concerns the dynamics of Semantic Web Technologies regarding Greek Legislation. Based on this research, Chapter 3 analyzes the new RDF re-designed data model, the challenge of representing legal documents and modifications in RDF/OWL, URI encoding and REST Services and finally a brief presentation of the use cases for our design proposals. Furthermore, in Chapter 4 we present the architecture of our platform on all of its levels and its different parts among with some use cases. Chapter 5 is about the demonstration of this platform and serves the purpose to navigate the reader of this thesis through the available web services we are serving through this work. In summary (Chapter 6), there are possible matters that can be dealt with in future work.

Chapter 2

Background

In this Chapter we are going to discuss the background of Greek Legislation structure, its challenges and our proposals for solutions. We will be proposing our way of dealing with modifications in legal documents and temporal knowledge. The structure itself brings us to the main point of this Chapter which is, why we decided to re-design our framework and replace the old one, with Semantic Web technologies. We are also presenting the benefits from this transition along with all advantages in semantic data designs, their dynamics and the future of Semantic Web.

2.1 Greek Legislation Structure - Modifications, a great structural and temporal challenge

2.1.1 Greek Legislation structure

As we already have stated, legislation for a non expert becomes difficult to understand let alone transforming it into a semantic data model. Especially in the case of Greek Legislation we are facing challenges like the mediocrity that Hellenic Administrations have been dealing with issues like publishing government files and national archives for many years. Legislation in Greece has been accessible through the Government Gazette which is a fully bureaucratic process and many times comes with printing errors. It had to be until 2010 for Greece to revitalize the way legislation is handing out. According to *Article 7 of Law 3861/2010* for the Di@vgeia[1] program about transparency through mandatory disclosure of government's decisinstitutions and acts, the Government Gazette is now available to citizens free of charge from the website of the National Printing House for reading, storing and printing. The Article 3 of the same Law refers to the publication on web, for all public documents as separate entities from individual institutions, something that currently happens via Di@vgeia website. Di@vgeia website publishes any kind of public document, including legislation, totally inconsistently either in text or in PDF format. Due to the lack of machine-readable format, there can be no advanced search functionality and any kind of advanced services [1]. This fact stand alone does not solve the issue.

A modern web platform like the one we are developing aims to go higher than just serving the text of a legal document, we want to elicit all useful information that comes with one. Mainly we want to publish legislation, both in terms of legal documents in machine-readable format and also legislative knowledge (legislative interactions - events) as linked data, which can offer crucial semantic information that currently is "ruined", in order to deliver advanced legislative services open publicly. We are also aiming in serving legislation dynamically across a time-line and defining the changes, editions or enrichments that may have occurred over the years. Our understanding for legislation so far, tells us that we are dealing with documents with many versions and references which are capable of changing

[1] See `https://diavgeia.gov.gr`

from day to day and we have to keep a record of this behavior. To achieve this we first had to understand the structure of Greek Legislation. At this point we shall make a few commends on legislation structure. The encoding of Greek Legislation in our project, follows the rules set out in “Manual directives for encoding of legislation” [7], and the final adoption reflects as it was listed in a comprehensive way in “Modeling and Querying Greek Legislation using Semantic Web Technologies” (2015) [8] and has been the guide on which we are following to encode Greek Legislation. The structure itself is defined as nested, because it follows a very strict way of encapsulating many different structural divisions of legislation, as they are reflecting Greek Legislation text.

2.1.2 Modifications

Now, it is common international practice, the amendment of divisions of legal documents by later legal documents. This change is not fundamental, but merely modifying and/or removing divisions (such as article, paragraph or sentence). In Greek Legislation, there are often modifications through replacements, additions or extensions (add-on phrases) in passages and paragraphs’ cases, but generally in any kind of text’s subdivision. Therefore a legislative modification has independent significance but is also expressed in a certain way (replacement or extension) on the legislative text. It is necessary to mention that sometimes modifications are called in the level of words and phrases. Unfortunately in Greek Legislation, modifications as one could put it, consist of idioms. Over the years there has not been a certain institutionalized way of drafting modifications or even computerizing them. This fact leaves us suspended, in interpreting and understanding each modification in a different way. This has become for us a very big challenge to deal with. We had to make an automatic and stable way of modeling modifications, understanding and transforming them into meaningful linked information for our purposes and for the purposes of Greek Legislation’s modernization. In this project we are also trying to serve temporal knowledge, as legal documents have enacted versions or active periods or even a trail of their modifications in partial structure elements which forms the structural knowledge (e.g. paragraphs added/extended, cases deleted/edited etc.). We had to choose based on which facts a certain modification applies into which rule. But to come to a right conclusion, we had to keep in mind the life cycle of legal documents. The “date-publication” of a legal document is the time the document is officially published or announced. The “date-enacted”, the time the content becomes applicable in decision making, is always later than or the same as date-publication, but before “date-repealed”, the time the content becomes inapplicable in decision making [4]. As long as we are talking for time periods between date-enacted and date-repealed the element and its content is “active” and outside this interval it is “inactive”. So a piece of legislation can have different states and different structural elements as we navigate through time. It becomes immediately obvious that modifications are the cornerstone of legislation. They might have effect on the meaning of a legal document, its use or even its own lifetime. A time-line of the life cycle of legal documents can be seen in Figure 2.1.

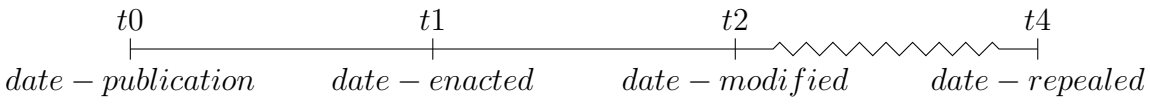

Figure 2.1: Life Cycle of legal documents

To sort the life cycle problem out, we established a way of dividing modification based on (a) which structural division it refers to and (b) in which way it changes it (deletion, expansion, addition, editing). These types refer to situations from Greek Legislation amendments based in our observance and experience thus it its very likely for them to change, evolve of even disappear in the future of the project. Modifications in legislation are still an open challenge and have drawn a lot of attention from our time of research. Concluding we would like to present these types of modifications that are being supported in our platform so far. We can find the below mentioned types:

- Adding a whole/part structural division in existing legal documents.
- Removing a whole/part structural division.
- Editing/Replacing a whole/part structural division.

2.2 The dynamics of RDF Data Models, OWL Ontologies and SPARQL - Semantic Perspectives

World Wide Web has been designed in order to offer information understood by human beings. It consists by billions of documents, mainly HTML pages, interlinked via “lousy” hyperlinks. Current architecture offers very limited opportunities for querying and retrieving information tasks by machines. Semantic Web provides a state of the art framework that allows data to be shared and reused across application, enterprise, and community boundaries with great respect in machine automated processing. This framework has three crucial elements: the RDF data model, the OWL ontology and SPARQL, the query language for RDF data sets.

2.2.1 The new Data Model with Resource Description Framework (RDF)

A data model is defined as “the data items of a certain part of the perceived reality (business domain) relevant for a specific application or a specific user in a structured way. This model includes the data relationships” [9]. Several data models have been implemented for management of changes in legislative document systems. But how our data model enables us to observe the versions or the metadata of a legal document and solve the issues mentioned in previous Sections. The Resource Description Framework (RDF) is a framework for representing information in the web. The core structure of the abstract syntax is a set of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph. An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. RDF data comply on RDF Schema, a semantic extension which mainly support classification and describe the interactions between resources. Conceptualizing legislation in a machine readable format, RDF obviously seems to be a perfect fit. But before we can start talking about our new RDF data model, let us see why our old XML schema does not suits the purposes of our goals from now on and in the future. While the first problems addressed with semantic information like, the need to distinguish between information content and presentation in a legal document, have been dealt with XML (XML tags used to express the “semantics” of various pieces of information),

different legal documents may represent in different ways semantically related pieces of information. That brings to the surface a new problem that XML schemas and data models could not solve. So, such different XML documents might not share such common semantics. RDF in the contrary, is a data model that can express the “meaning” of such a piece of information in a shared way. With dropping the XML technology from our data model we immediately dropped problems like querying the same semantics represented in different XML trees or the need of converting of all possible representations of a fact into one statement. RDF gives us a standard way of writing statements. Finally, we end up with a unifying model not only to itself but open to other data models as well, immediate expandable by adding simple statements without re-designing the whole model and fast in terms of querying and mining information. So in this project unlike the old version of Nomothesi@ we are trying a new approach. Instead of using RDF to store metadata for Greek Legislation, we are also using it to shape the whole model for two main reasons. First of all to query them and infer the appropriate knowledge needed for advanced services. Secondly to publish our data sets and link those to third-party data sets across web.

2.2.2 OWL Ontologies and SPARQL

Web Ontology Language (OWL) designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to infer implicit knowledge explicit. Legislation is all about consistency and implicit knowledge, semantic information spelled in words, ruined in text documents. OWL documents, known as ontologies, can be published in the World Wide Web and may refer to or be referred from other OWL ontologies. The data described by an ontology in the OWL is interpreted as a set of “individuals” and a set of “property assertions” which relate these individuals to each other. An ontology consists of a set of axioms which place constraints on sets of individuals (called “classes”) and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. We are certain to believe that we have established a strong and efficient data model given all the advantage above using RDF, RDFS and OWL. Now that we have added semantics to our RDF data, we must consider how is it published and queried. SPARQL is an RDF query language, that is, an SQL-like semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middle-ware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions, it also supports extensible value testing and constraining queries by source RDF graph. For the very first time in Greek Legislation and worldwide, our project innovates with the idea of serving a SPARQL Endpoint in a RESTful manner. Users can now take advantage of legal documents not only by reading them, but also querying all the semantic information. This forms a new path in the way legislation is being served and represented by any organization or official institution and we hope that many other will follow our way until we have unified government archives of all types. At this point we would like to present some technical information regarding the semantic information available from our Endpoint. The main query operations in this system are:

(a) Obtaining the legislative document valid at a given date or a time space.

(b) Historical evolution of a legislative document.

(c) Full text search in the text of the legislative document.

(d) Laws repealed by a law.

(e) Obtaining all kinds of Metadata (the model allows new Metadata to be easily included).

(f) Description queries on the RDF graph.

2.2.3 Semantic perspectives and future Web

The Semantic Web is a vision for the future of the web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the web. As Sir Tim Berners-Lee puts it:

> "The concept of machine-understandable documents does not imply artificial intelligence which allows machines to comprehend human mumbling. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operation on existing well-defined data."

We have to be then, as more efficient as it gets when we are about to represent complex documents on the web, and especially more careful when we talk about legislation. The legal systems are usually developed in several databases and formats that require integration. RDF offers the possibility of building a simple semantic data model with several advantages over traditional data models for legal systems. So, RDF helps to merge data from multiple sources and does it in such a way that data about the same things gets collated. This merge comes with the benefit of enabling cross-data source querying using SPARQL as we saw earlier, allowing content or relationships between content from several datasets to be discovered. Properties and values can be described with a shared schema or ontology, taking advantage of RDF in legal systems. But most importantly, this new era of modeling legislation semantically helps us develop a Restful API to serve different versions of a legislative document in different formats at the same URI, to give developers full and open access to the data, which concentrates the vision of this project.

Chapter 3

Nomothesi@ API: Reengineering Nomothesi@ platform

Adapting a new RDF data model helps us for the first time to unify all of our resources because it allows us to describe them with RDF expressions, always named by URIs. In this Chapter we are going to present how are RDF and OWL making possible the representation of all the structural elements of Greek Legislation. How are we using OWL properties to expand temporal and semantic knowledge and how we bind this technology with URIs to use it directly onto the web.

3.1 Ontology presentation

3.1.1 Representing legal documents

Unlike XML, when simple RDF vocabulary reaches its limits we use RDFS (RDF Schema) and OWL to extend it. We can imagine RDF as an inexhaustible source of vocabulary, this way we can express without limitations every single aspect of legislation. We follow the architecture of different entities. We have the main entity which is the legal document itself. It contains all metadata and each piece of it implements a different instance of a *Fragment*. Fragments can be articles, paragraphs or other structural entities. To distinguish versioning and metadata, our data model follows in terms of storing the model of legislative versions, in independent documents, with links to related version or metadata about modifications. As we discussed in Chapter 2, this project adopts CEN MetaLex standard OWL ontology and for the first time drops the XML schema for an RDF one, so that we can extend and upgrade this ontology with more classes and properties, to represent Greek Legislation in the most effective and efficient way. The benefits coming from using OWL ontologies representing information on legislative events (publication, modification, repeal) concern the validity, the content and in general the behavior of the legal document (work) and also the interactions between legal documents arising from specific events.

The main entities used for describing interactions in this model are:

Bibliographic Work is any legal (bibliographic) document (e.g. law, presidential decree, etc.), created and published by legislative procedures at a given time. For our project's uses we added under Bibliographic Work the subclass Signer which represents the ministers signing any type of legal document of those being implemented by *Bibliographic Work*.

Bibliographic Expression is every expression (version), which is expressed by modifying the original content of a specific legal (bibliographic) work.

Fragment is identified as a fragment (structural element) of a legal document. Trying to be more accurate and improving legal document representation, we expanded this class by adding subclasses such as: *Article*, *Paragraph*, *Case*, *Passage* etc. In this way we managed to model the Greek legal documents structural elements into different implementations of class Fragment saving their unique semantic information but at the same time sharing all common attributes and properties.

Legislative Modification defines any modification in content (e.g. article, passage, case etc.) on a legal document.

Legislative Competence Ground defines legal documents, which contains modifications referring to other legal documents.

Legal documents (*BibliographicWork*) are organized in fragments (*Fragment*). Each legal document has (*realizedbBy*) subsequent versions (*BibliographicExpression*), which are differentiated by the previous ones based on (*matterOf*) some subsequent modifications (*LegislativeModification*) that are referred in subsequent legal documents, known as their *LegislativeCompetenceGround*. They also include references (*BibliographicCitation*), which cite other legal documents or their fragments (*CitableBibliographicObject*) and are defined around a textual context with respect on the encoding. Most of these concepts come straight from the adoption of the MetaLex ontology (Figure 3.1). MetaLex ontology mainly describes the event-driven process of legislation, we extend this basis to describe also the structure, the content and the metadata of legal documents. Greek legal documents are organized in specific types of *Fragment* (e.g. *Article*, *Paragraph*, *Passage*, etc.). Cases and passages are the bottom elements that define text (*text*). Legal documents come along with some metadata concepts such as signers (*signer*), the government members who signed (*signer*) this document, the Gazette issue in which has been published and possibly a Place which is referred by the document. Title (*title*), publication date (*created*) are also crucial metadata pieces of information recorded.

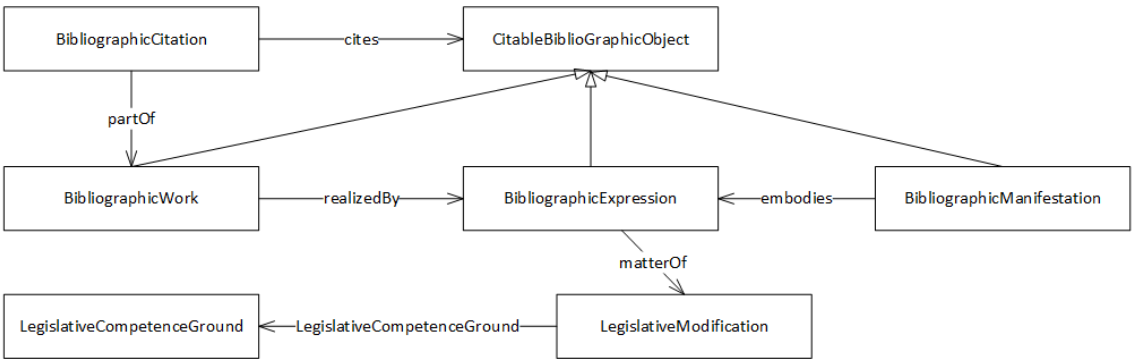

Figure 3.1: CEN Metalex OWL Ontology Core

Another thing working with ontologies, is storing and producing crucial RDF metadata for each entity. As we already know CEN MetaLex has been designed to store legacy metadata from RDF triples with an expression URI as subject, the literal attribute value aw object, and an RDF property with the source attribute's name as predicate [10]. Now, for structural and identity metadata we need to combine our resources according to our ontology to store chronological data (dates), thematic (title, tags, organizations involved etc.) and many other government information. A complete picture of our OWL ontology is being displayed in Figure 3.2 below.

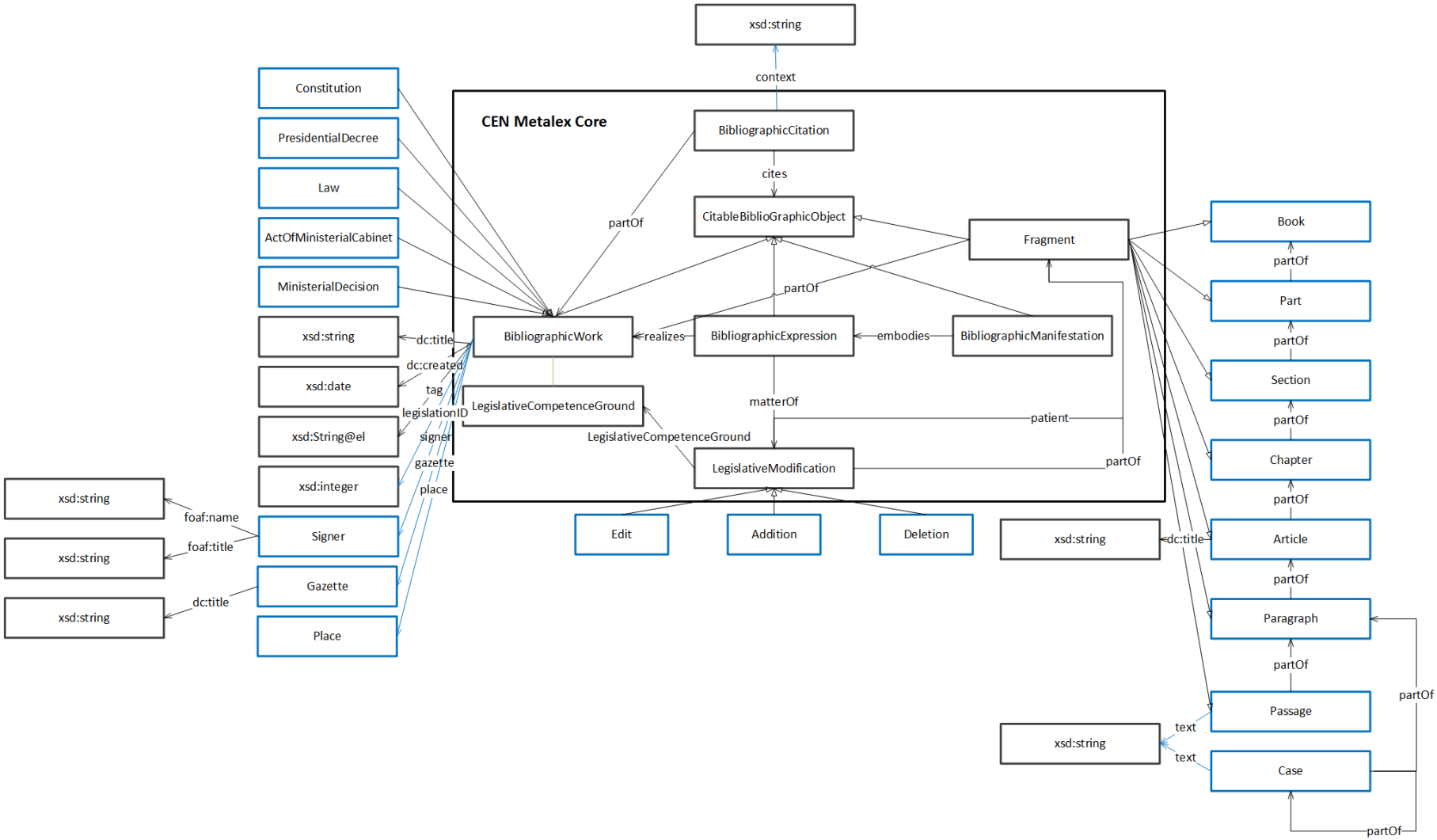

Figure 3.2: OWL Ontology for Greek Legislation

In Figure 3.2, we are presenting not only the core elements of Metalex OWL ontology, which are inside the frame, but also all the other very basic elements (DC Terms, add-ons) and metadata.

3.1.2 Publishing modifications

In Chapter 2 we explained the importance of modifications to a legal document. We presented our perception on them, and how we settled our minds in terms of modification-representation in this project. Now we would like to explain in an RDF based analysis the use of modifications and how they interact with the publication of a legal document or its versions through time. We have already seen that under *LegislativeModification* we have four more subclasses, *Deletion*, *Edit*, *Creation* and *ChangeWords*. *LegislativeModification* defines any modification in content and based on its type. We have settled the problem regarding what forms a modification in Greek Legislation as we extensively described in Section 2.1.2.

e.g.

<!– http://legislation.di.uoa.gr/ontology/Deletion –>
<owl:Class rdf:about="&ontology;Deletion">
<rdfs:subClassOf rdf:resource="&metalex;LegislativeModification"/>
</owl:Class>

Next, we have to state in RDF terms, that a modification might appear, and might be stating that a past legal document must change based on a certain type. For that we are using the *matterOf* property to link the exact *LegislativeModification* to a *BibliographicExpression* so we can store the information regarding when this modification modified the original version of the legal document that has to be modified.

e.g.

<http://legislation.di.uoa.gr/pd/2011/54/2012-02-02>
<http://www.metalex.eu/metalex/2008-05-02#matterOf>
<http://legislation.di.uoa.gr/pd/2012/10/article/1/paragraph/1/modification/1>.

With the property *LegislativeCompetenceGround* we can define that a certain legal document is containing modifications, referring to other legal documents.

e.g.

<http://legislation.di.uoa.gr/pd/2014/65/article/7/paragraph/1/modification/1>
<http://www.metalex.eu/metalex/2008-05-02#legislativeCompetenceGround>
<http://legislation.di.uoa.gr/pd/2014/65>.

In this way we are storing information about our modifications, how many we have and where they are referring to. Finally we must link these knowledge in terms of versioning, so we serve on demand the original version (enacted) or a current one (up to date) or in all its forms between the original publication date and the current form. We can accomplish that with another data within our ontology *realizedBy*.

e.g.

<http://legislation.di.uoa.gr/pd/1985/373>
<http://www.metalex.eu/metalex/2008-05-02#realizedBy>
<http://legislation.di.uoa.gr/pd/1985/373/2014-04-28>.

OWL offers many data properties and classes and there for many paths to follow on how to represent and publish a legal document to its final form. We have to state that our approach has developed having in mind the peculiarities of Greek Legislation.

3.2 URIs - Rest services

Uniform Resource Identifiers (URIs) are short strings that identify resources in the Web: documents, images, downloadable files, services, electronic mailboxes, and other resources. Such identification enables interaction with representations of the web resource over a network, typically the World Wide Web, using specific protocols. In addition to utilizing the HTTP requests appropriately, resource naming is arguably the most debated and most important concept to grasp when creating an understandable, easily leveraged web service API. When resources are named well, an API is intuitive and easy to use. Done poorly, that same API can feel klutzy and be difficult to use and understand. Essentially, a RESTful API ends up being simply a collection of URIs. In our platform, each resource has its own address or URI- every interesting piece of information the platform can provide is exposed as a resource. In other words, the RESTful principal of addressability is covered by the URIs. We have chosen that each resource in a service suite will have at least one URI identifying it. And it is for our benefit when that URI makes sense and adequately describes the resource. URIs should follow a predictable, hierarchical structure to enhance understandability and, therefore, usability, they have to be predictable in the sense that they are consistent and hierarchical in the sense that data has structure-relationships.

Fixed URIs to divisions of legislation are very important, as they are on legal documents in general. Various initiatives are trying to upgrade reliable classification for the legislation to existing bibliographic scheme. Their aim is to facilitate the process of creating URIs for legal sources, regardless of the availability of a document on the web, location of a document, and the way to access it. Initiatives such as PRESTO[1] in the UK describe a system for legislation and public information in which *"All documents, views and metadata at all significant levels of granularity and composition should be available in the best formats practical from their own permanent hierarchical URIs."* [11]. Based on international practice and the particularities of Greek legislation, we proposed a schema of URIs, which is very similar with the UK.

In our platform every single legal document, its subparts, its versions or services are resources, which need to be addressed by a specific URIs system. URIs system must be persistent and build so as URI for any kind of resource to be highly guessable. There are a number of different ways one might assign an unequivocal identifier to a legislative document. We have decided to use HTTP URIs. These URIs have been designed following the guidelines of our conception on the matter, but we are hoping that our work will form a new way of describing Greek Legislation over the web. At this point we would like to present the Schema of URIs one can get in contact with when using our REST services.

[1] See `http://archive.oreilly.com/pub/post/presto_a_www_information_archi.html`

Main pattern:

`http://legislation.di.uoa.gr/{typeoflegislation}/{year}/{id}`

Any field between curly brackets needs to take specific value. As a type of legislation, we mean all different types of Greek Legislation, we are using the encoding of Table 3.1. Year is actually the year of publication (e.g., 2012) and ID is the Number (ID) of the specific legal document. So for example if we want to address in Law 12 of 2014, the corresponding URI is:

`http://legislation.di.uoa.gr/law/20014/12`

Table 3.1: Encoding Types of Legislation

Type of Legislation	Code
Constitution	con
Law	law
Presidential Decree	pd
Act of Ministerial Cabinet	amc
Ministerial Decision	md

Correspondingly, we can follow a structural search:

e.g. `http://legislation.di.uoa.gr/search?keywords={titleoflegaldocument}&type={typeoflegislation}&year={year}&id={id}&date={YYYY-MM-DD}`[1]

Keywords are possible words of the title, separated with comma (,). Type of legislation, year and ID follow the same rules, as we already mentioned. Date of publication must be in a specific format, which contains 4-digit year, 2-digit month and 2-digit day separated with dashes (-). Redesigning our platform gave us the opportunity to redesign the service of advanced search under the search location. In this way we can now search for legal documents in a time distance. Any of these criteria are optional.

e.g. `http://legislation.di.uoa.gr/search?keywords={titleoflegaldocument}&type={typeoflegislation}&year={year}&id={id}&date=datefrom={YYYY-MM-DD}&dateto={YYYY-MM-DD}`

We can also refer to specific conceptual unit of the legal document, by collocate the branch of the calling division after the ID. So for example if we want to address in the third paragraph of the fourth Article in Law 3/2003, the corresponding URI is:

`http://legislation.di.uoa.gr/law/2003/3/article/4/paragraph/3`

[1] See `http://en.wikipedia.org/wiki/ISO_8601`

As we have mentioned already, there are three type of versions for legal documents. In order to refer to them, we follow the URI's extensions, in Table 3.2.

e.g. `http://legislation.di.uoa.gr/law/2003/3/2007-12-23`

Table 3.2: Encoding Versions of Legislation

Version	URI extention
current	-
enacted	enacted
date-version	YYYY-MM-DD

Any legal source has multiple file formats available, both for internal use of the system and also to share open with third party CS engineers. The available file formats are: XML file (.xml), RDF file (.rdf), PDF file (.pdf) and JSON file (.json). For the disposal of these files we use the following extensions, mentioned on Table 3.3.

Table 3.3: Encoding Types of File

File - Manifestation type	URI extension
XML	data.xml
RDF	data.rdf
PDF	data.pdf
JSON	data.json

Also, for the first time with this project we added the ability for a SPARQL Endpoint service. Under this service users can perform SPARQL queries and collect a vast amount of information based on their needs. The main pattern of an endpoint URI is:

`http://legislation.di.uoa.gr/legislation/endpoint?{query}&format={format}`

Where the word query between curly brackets represents the query to be executed through the Sesame database, and the word format between curly brackets has the values mentioned on Table 3.4, our platforms offers the result in these formats in a RESTful way to encourage further use of information.

Table 3.4: Query results format extensions

Query result type	URI extension
HTML	format=HTML
SPARQL/XML	format=XML

3.3 Apply Nomothesi@ - Use case

At this point we have presented some of the main classes and properties of our ontology. We disgusted, how a legal document is being arranged to have semantic information from plain text, its organic structure and the key element of this arrangement which is the conception of modifications in semantic web standards. Now, it is time to move up a step and explain how all these classes and properties which are being transfigured in RDF graphs, interact with each other, bind with each other and finally produce the final linked, semantic and solid representation of Greek Legislation.

To achieve this we will be studying a specific use case, on how we apply Nomothesi@ on Greek Legislation. For our example we will examine Presidential Decree (PD) 2011/54 and Presidential Decree (PD) 2012/10. Let us take a look below.

Example of Presidential Decree[1]

(Type) *(Number)*
PRESIDENTIAL DECREE **No. 54**

(Title)
Flood Protection Special Service Public Works of the valley of the river Evros and its tributaries (EYDE EVROY)

THE PRESIDENT OF THE HELLENIC REPUBLIC

Having regard to:

(Citation)

1. Paragraph 1 of Article 5 of Law 679/1977, "On increase increase staff posts for the Public Works Ministry and regulation of related matters" (A' 245) as the last two passages of this paragraph amended and supplemented by par. 1 of the Article 23 of Law 1418/1984 (A 23).

[...]

Article 1
Establishment, Title , Responsibilities

1. A Special Service entitled "Flood Protection Special Service Public Works of the valley of the river Evros and its tributaries" (EYDE EVROY) , with responsibility for all matters relating to the special and important work of flood protection of the valley of the river Evros, Erythropotamos, Arda, tributaries and Orestiadas' Regional Moat.
2. This Service shall exercise all responsibilities of the Management Service in accordance with the provisions applicable to the design and construction of Public Works.
3. Registered office of EYDE EVROY is set Soufli and its running period is specified at eight years after the entry into force of this Decree.

[...]

(Publication Date)
Athens, 24 May 2011

(Signatures of the competent Ministers)
PRESIDENT OF REPUBLIC
KAROLOS G. PAPOULIAS

DEPUTY MINISTERS

INFRASTRUCTURES, TRANSPORT AND NETWORKS
IOANNIS MAGKRIOTIS

[1] As the Greek legislation does not formally be translated, the translation of this presidential decree is unofficial for the purposes of this thesis.

Studying legal documents that have been published in the Government Gazette, we noticed that most modifications are related with passages or cases (e.g. “At the end of paragraph X add passage as follows”, “The X passage of paragraph Y of Article Z is replaced as follows”). This directly affects the design of our data models. Looking the legislative document as a tree structure from top to bottom, we understand that our “children” nodes should be at the level of passages and cases as structural entities in our data models.

Example of Legislative Modifications

Article 1 of PD 54/2011 has the following structure:

Article 1
Establishment, Title , Responsibilities

1. A Special Service entitled “Flood Protection Special Service Public Works of the valley of the river Evros and its tributaries” (EYDE EVROY) , with responsibility for all matters relating to the special and important work of flood protection of the valley of the river Evros, Erythropotamos, Arda, tributaries and Orestiadas’ Regional Moat.
2. This Service shall exercise all responsibilities of the Management Service in accordance with the provisions applicable to the design and construction of Public Works.
3. Registered office of EYDE EVROY is set Soufli and its running period is specified at eight years after the entry into force of this Decree.

There are couple of legislative modifications, an addition and a replacement, in the previous article by Article 1 of P.D. 10/2012.

Article 1

1. After the end of paragraph 1 of Article 1 of Presidential Decree 54/2011 is added paragraph as follows: “The maintenance of these projects remain within the remit of the region of Eastern Macedonia and Thrace in accordance with the provisions of Law 3852/2010 (Government Gazette A 87).”
2. Paragraph 3 of Article 1 of Presidential Decree 54/2011 is replaced as follows: “Registered office of EYDE EVROY is set Alexandroupoli and its running period is specified at eight years after the entry into force of this Decree.”

Bearing in mind the legislative modification, the updated version of Article 1 of PD 54/2011 follows.

Article 1
Establishment, Title , Responsibilities

1. A Special Service entitled “Flood Protection Special Service Public Works of the valley of the river Evros and its tributaries” (EYDE EVROY) , with responsibility for all matters relating to the special and important work of flood protection of the valley of the river Evros, Erythropotamos, Arda, tributaries and Orestiadas’ Regional Moat. The maintenance of these projects remain within the remit of the region of Eastern Macedonia and Thrace in accordance with the provisions of Law 3852/2010 (Government Gazette A 87).
2. This Service shall exercise all responsibilities of the Management Service in accordance with the provisions applicable to the design and construction of Public Works.
3. Registered office of EYDE EVROY is set Alexandroupoli and its running period is specified at eight years after the entry into force of this Decree.

P.D. 2011/54 is a Presidential Decree, which comes as a generalization of *BibliographicWork*. It consists of 6 articles, each has its own nested structural elements (paragraphs, passages, cases) which are different instances of *Fragment*. There are also the appropriate metadata (title, publication date, signer, etc.) (Figure 3.3).

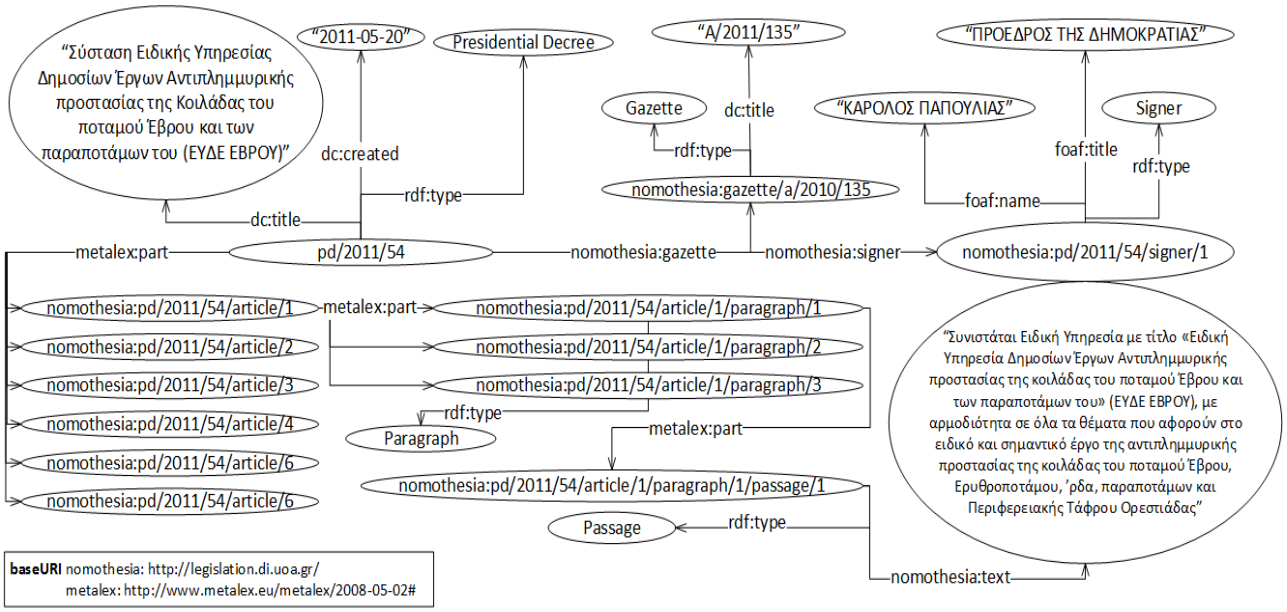

Figure 3.3: P.D. 2011/54 Structure RDF Graph

No matter how many structural elements (articles, paragraphs, cases, cases of cases) a legal document might have, we have managed under this graph representation with the use of *Fragment* to encapsulate this structural complexity. That is why, all structural elements are types of fragments (Article, Paragraph, Passage, etc.), and each fragment is encapsulated (*partOf*) by a greater fragment level. We would like to state that we have efficiently found a way of representing each structural element without degrading semantic information. But this is not enough, as we have already experienced, Greek Legislation might become a little vague and as a result the majority of all legal documents, have modifications referring on them. We discussed the issues raising from the existence of modifications in Section 2.1.2, and our solution was presented in technical terms in Section 3.1.2. *LegislativeModifications* are also encapsulated by the *Paragraph* level nodes and they encapsulate a nested *Fragment* hierarchy. So below at Figure 3.4 we can see the interaction of classes and properties that sort out the modification problem.

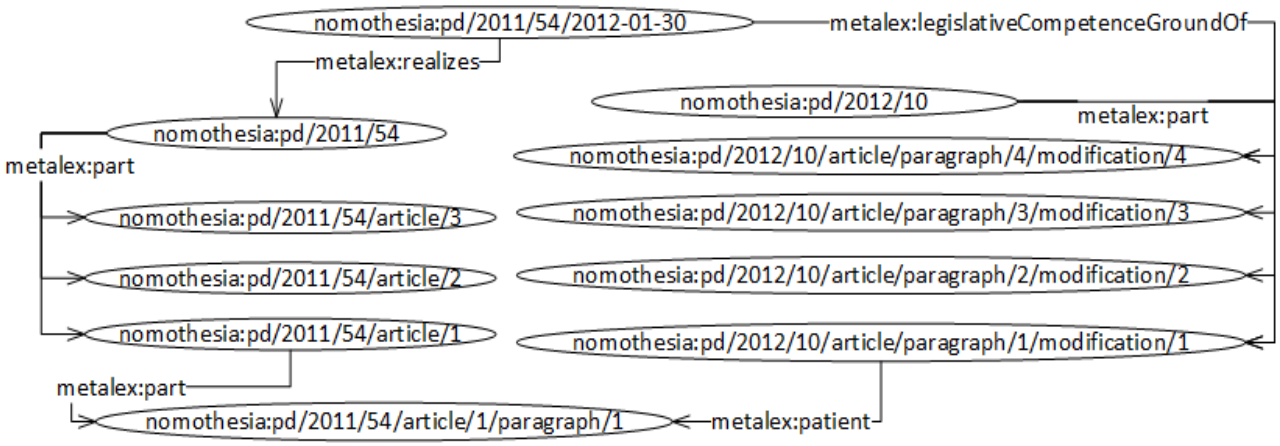

Figure 3.4: P.D. 2011/54 - P.D. 2012/10 Interactions RDF Graph

P.D. 2012/10 acts as a Legislative Competence Ground for Modifications applied on P.D. 2011/54 and so on produce a version that realizes this legal document. What we just saw in the figure above, was the application of all modifications that the amendment legal document states (PD 2012/10), based on their type (*Edit*, *Delete*, *ChangeWords* etc.), to the modified legal document (PD 2011/54). We have developed this architecture that helps us not only to have knowledge over the applied modifications (*matterOf*) but also to have a fully updated and linked (*realizedBy*) legal document with the one that states the amends, being able to present the modified divisions though out its structure or follow these changes through time (versioning), making in this way modifications to look like structural elements (fragment types) of the original legal document.

Chapter 4

Nomothesi@ API architecture

Nomothesi@ API is a RESTful Web Application as we have already explained. Taking the time to look into its architecture, one can see that we have chosen Spring MVC as a Web Framework, Sesame as an RDFStore Server and we make requests to Strabon as a client. So every HTTP request passes from Spring, then if needed we settle a connection with Sesame and/or Strabon to retrieve appropriate data and turn back to Spring to form the HTTP response for the user. In this Chapter we are discussing the main modules constituting our API.

4.1 Spring MVC Framework

A new thing about reengineering Nomothesi@ platform was that while we dropped the XML schema, we had no further use for a XML pipeline, so we faced the challenge of investigating the existence of a new and more suitable application framework as well. While the previous version was built on Apache Cocoon Framework[1], for this project we finally settled for Spring MVC Framework[2]. MVC stands for Model-View-Controller which makes Spring Framework popular amongst web applications. The *Model* encapsulates the application data and in general they consist of POJO (Plain Old Java Objects). The *View* is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret using JSP files in our case, and finally the *Controller* is responsible for processing user requests and building appropriate model and passes it to the view for rendering. Spring's Web MVC Framework is, like many other Web MVC Frameworks, request-driven, designed around a central Servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications. Spring's *DispatcherServlet* however, does more than just that. *DispatcherServlet* handles all the HTTP requests and responses. The request processing work-flow of the Spring Web MVC *DispatcherServlet* as we can see in Figure 4.1.

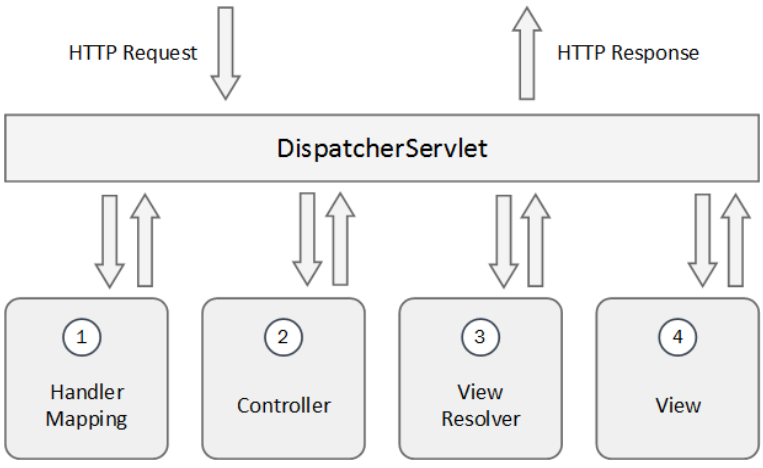

Figure 4.1: Work-flow of the Spring Web MVC architecture

Following is the sequence of events corresponding to an incoming HTTP request to *DispatcherServlet*:

1. After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate *Controller*.
2. The *Controller* takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.
3. The *DispatcherServlet* will take help from *ViewResolver* to pickup the defined view for the request.
4. Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser.

Also, Spring is known for providing hooks for extension and customization in RESTful APIs. In a RESTful API like Nomothesi@ we are receiving and serving many different requests. It is a very common use case to have Controllers serving only JSON, XML, RDF or custom media type content. For this reason Spring's view resolution is extremely flexible. We can directly generate XML (.xml), JSON (.json), RDF (.rdf), and many other types of content at an instance saving time and complexity. At the same time Spring enables custom JSP tags and it is meant for front-end frameworks like Bootstrap[3] which is the one we are using. However, to fully understand how Spring enables its flexibility and clean roles we can take a look at Figure 4.2.

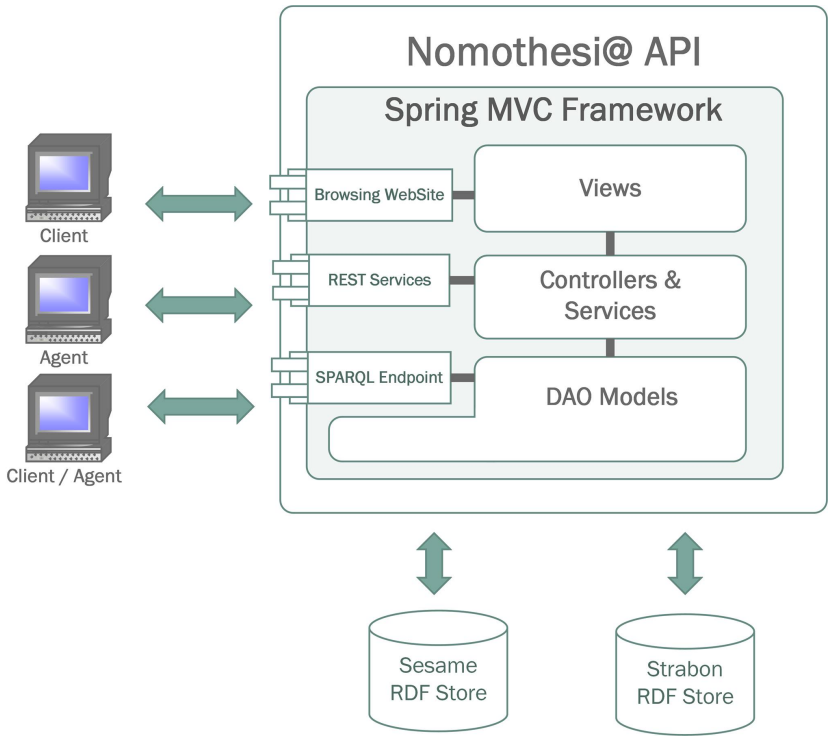

Figure 4.2: Spring MVC Framework architecture within Nomothesi@

[1]See `http://cocoon.apache.org`
[2]See `https://spring.io`
[3]See `http://getbootstrap.com`

4.2 Sesame RDF Store Server

Our project needs a RDF Store to organize RDF triples. OpenRDF Sesame[1] is a de-facto standard framework for processing RDF data. This includes parsers, storage solutions (RDF databases a.k.a. RDFStores), reasoning and querying, using the SPARQL query language, that we introduced in Section 2.2.2. It offers a flexible and easy way to use Java API that can be connected to all leading RDF storage solutions. Using Sesame remotely as a Server in a Client/Server architecture, whose client through Java API is Spring, we can handle CEN Metalex OWL ontology data.

4.3 Strabon RDF Store

The reengineering of Nomothesi@ enabled us to link semantic legislative documents to the semantic spatiotemporal RDF store Strabon. Strabon[2] follows the modular architecture of Sesame. It has a module called Storage Manager which handles data storage and extend Sesame's components in order to be able to store linked geospatial data that changes over time. Unlike Sesame we are using Strabon remotely as a Client in a Client/Server architecture. Our request for geospatial semantic information has the form of a query which is applied to Stabon's endpoint. The returned data have the form of KML and then, we are able to connect our references of administrative districts in a legislative document, in a semantic way over the web, with administrative divisions as they have been formed with Kallikratis Plan[3].

4.4 Use cases and Rest Services

Based on Figure 4.2, in this Section we are about to display the two most important and complex use cases of our API, as well as, all the "state of art" REST services we deliver through our platform. The use cases will be referring to the background of the platform in a form of steps beginning from a request and ending with the response of each case. In that way we shall make the point that Spring MVC is the best choice for RESTful APIs, to serve multiple type responses from URI requests.

4.4.1 Use case 1 - Present enacted legislative version

Suppose we take the following request: `http//legislation.di.uoa.gr/pd/2011/54/enacted`, which means to return Presidential Decree (PD) 2011/54 on its enacted version (original). We have the following steps (Figure 4.3):

[1] See `http://rdf4j.org`
[2] See `http://www.strabon.di.uoa.gr`
[3] See `http://www.kallikratis.eu`

1. Spring will map the request URI into the Controller (*LegislationController*).
2. The Controller will handle the request through the main Service (*LegislationService*) which is responsible for handling the functionality of the request.
3. The service decides which function has to be executed and begins building the response package by calling the DAO (*LegalDocumentDAOImpl*) to implement the request's functionality.
   (a) DAO calls method *getMetadataById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's metadata.
   (b) DAO calls method *getCitationsById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's citations.
   (c) DAO calls method *getById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's modular elements their structure, text, etc.
4. After having retrieved the requested legal document, the Controller calls the service once again, this time for fetching the possible modifications of the enacted version.
   (a) DAO calls method *getAllModifications* connects with Sesame Server to apply SPARQ query for retrieving the requested legal document's modifications. It accepts a list of modifications with two main elements: modification's URI and modification's source file. For each modification in the list, query the modification's source file and get as a result modification's content.
5. *LegislationService* returns the request package in POJO form and the Controller finally depending on the request's nature responds via the suitable View JSP to serve the HTTP request in the desirable media type form (HTML,XML,PDF etc.).

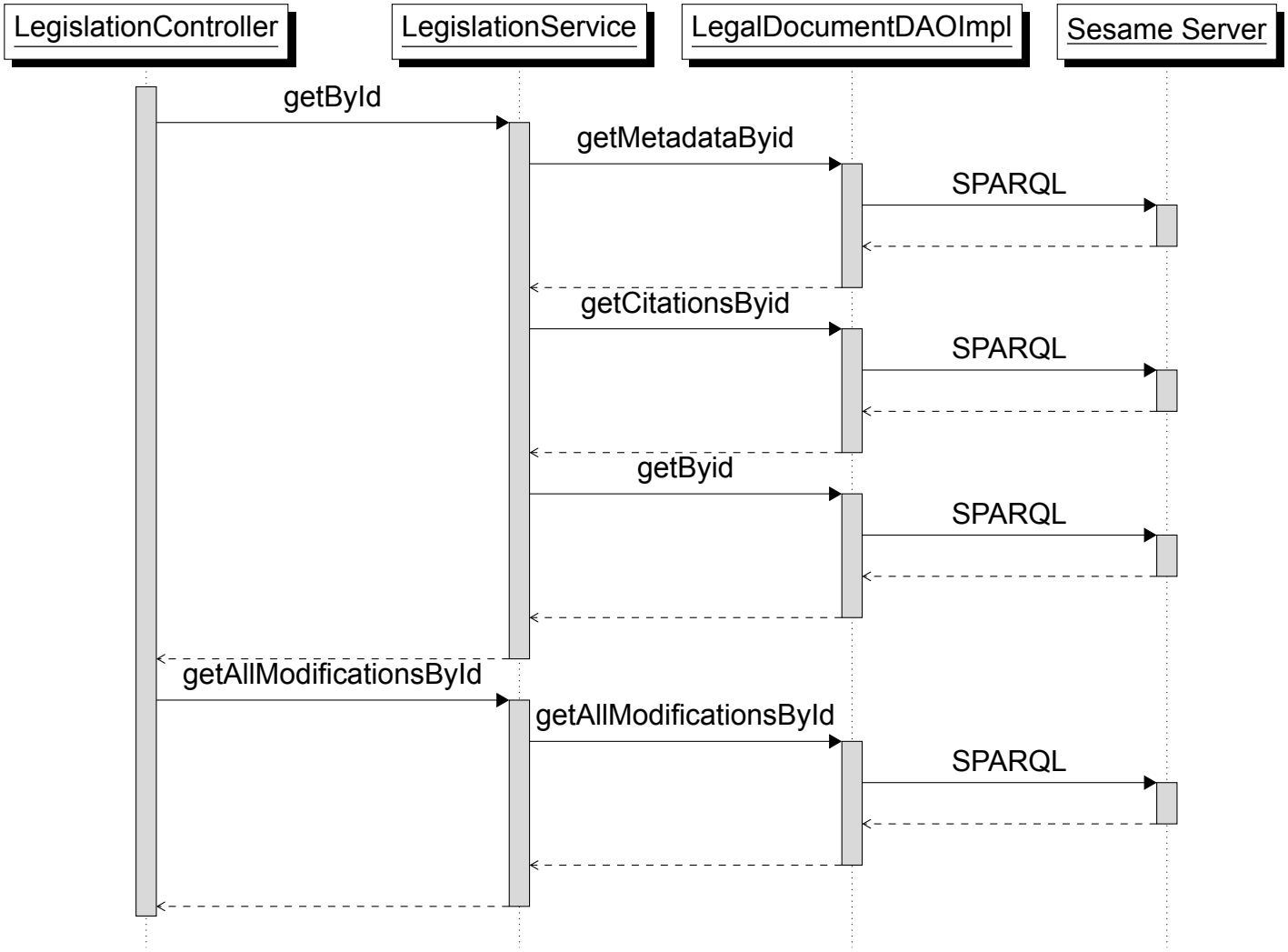

Figure 4.3: UML sequence diagram Use Case 1

4.4.2 Use Case 2 - Performing a search based on parameters

Now we are ready to perform o search scenario. We will be searching for Presidential Decrees (PDs) that their publication date ranges from 01-01-2010 to 01-01-2015. The request URI will be shaped like this: `http://legislation.di.uoa.gr/search?keywords=&type=pd&year=&id=&fek_issue=A&fek_year=&fek_id=&date=&datefrom=2010-01-01&dateto=2015-01-01`. We have the following steps (Figure 4.4):

1. Spring will map the request URI into the Controller (*LegislationController*). Note that all search parameters are communicated from the HTTP request through a Map<String,String> variable at the Controller.
2. The Controller will handle the request through the main Service (*LegislationService*) which is responsible for handling the functionality of the request.
3. The service decides which function has to be executed (*searchLegislation* in our case) and begins building the response package by calling the DAO (*LegalDocumentDAOImpl*) to implement the request's functionality.
   (a) DAO calls method search passing the search parameters. Then the method connects with Sesame Server, after a thorough casing, the method filters all the parameters validating their values and overlapping issues. The final search query that will be running over Sesame Server, is being built step by step. We are only fetching the title, type, and of the legal document. Finally the method, constructs a list of legal document POJOs based on Sesame's respond.
4. The Controller now through method *getTags* requests from DAO the tags metadata associated with the specific legal document.
   (a) DAO calls method *getTags*, connects with Sesame Server, and fetches all the tagging metadata, packages them in a list and finally it responds to the request.
5. *LegislationService* returns the request package in POJO form and the Controller finally serves the HTTP request in the appropriate View JSP.

```
@RequestMapping(value = "/search", method = RequestMethod.GET)
public String search(@RequestParam Map<String,String> params, Model model, Locale locale) {

    if(params != null){
        LegislationService lds = new LegislationService();
        List<LegalDocument> LDs = lds.searchLegislation(params);
        List<String> tags = lds.getTags();
        model.addAttribute("legalDocuments", LDs);
        model.addAttribute("tags",tags);
        model.addAttribute("locale",locale);

        if((params.get("keywords")!=null) && !params.get("keywords").equals("")) {
            model.addAttribute("keywords",params.get("keywords"));
        }

        if((params.get("date")!=null) && !params.get("date").equals("")) {
            model.addAttribute("date",params.get("date"));
        }
```

⇓ Call searchLegislation() ⇓

```
public List<LegalDocument> searchLegislation(Map<String, String> params) {
    //Get the Spring Context
    ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");

    //Get the ProductDAO Bean
    LegalDocumentDAO legalDocumentDAO = ctx.getBean("legalDocumentDAO", LegalDocumentDAO.class);

    return legalDocumentDAO.search(params);
}
```

⇓ Call search() ⇓

```
@Override
public List<LegalDocument> search(Map<String, String> params) {

    /*for (Map.Entry entry : params.entrySet()) {
        System.out.println(entry.getKey() + "" + entry.getValue());
    }*/

    List<LegalDocument> LDs = new ArrayList<LegalDocument>();
    String sesameServer ="";
    String repositoryID ="";

    Properties props = new Properties();
    InputStream fis = null;
      [...]
            //search between two dates
            if ((params.get("datefrom")==null) || (params.get("datefrom").equals(""))) {

            }
            else if((params.get("year")==null) || (params.get("year").equals(""))) {
                queryString += "FILTER ( ?date >= \"" + params.get("datefrom") + "\"^^xsd:date)\n";
            }

            if ((params.get("dateto")==null) || (params.get("dateto").equals(""))) {  [...]

            }
            else if ((params.get("year")==null) || (params.get("year").equals(""))){
                queryString += "FILTER ( ?date <= \"" + params.get("dateto") + "\"^^xsd:date)\n";
            }

            //search by year
            if ((params.get("year")==null) || (params.get("year").equals(""))) {

            }
            else if (((params.get("date")==null) || (params.get("date").equals(""))) && ((params.get("datefrom")==null) || (params.get("datefrom").equals(""))) && ((params.get(
                queryString += "FILTER ( ?date >= \"" + params.get("year") + "-01-01\"^^xsd:date && ?date <= \"" + params.get("year") + "-12-30\"^^xsd:date)\n";
            }

            //search by fek year, issue and id
            if (((params.get("fek_year")==null) || (params.get("fek_year").equals(""))) && ((params.get("fek_id")==null) || (params.get("fek_id").equals("")))) {

            }
            else {
                queryString += "?legaldocument leg:gazette ?gazette.\n" +
                "?gazette dc:title \"" + params.get("fek_issue") + "/" + params.get("fek_year") + "/" + params.get("fek_id") + "\".\n";
            }
```

Figure 4.4: Search implementation from *LegislationController* to *LegalDocumentDAOImpl*

4.4.3 Use Case 3 - Geospatial Data

Geospatial data are being requested when the requested legal document has reference to one or more administrative districts based on Greek law 3852/2010.

So when our request URI is `http//legislation.di.uoa.gr/pd/2011/54`, we are requesting a Presidential Decree that refers to the administrative district of Evros. The construction of the request will be exactly the same as every request on a legal document. Below we are presenting the use case we saw in Section 4.3 with a more analytical glance on the method *getMetadataById* assuming that the requested legal document has reference to an administrative district.

1. Spring will map the request URI into the Controller (*LegislationController*).
2. The Controller will handle the request through the main Service (*LegislationService*) which is responsible for handling the functionality of the request.
3. The service decides which function has to be executed and begins building the response package by calling the DAO (*LegalDocumentDAOImpl*) to implement the request's functionality.
   (a) DAO calls method *getMetadataById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's metadata.
      i. **If legal document has administrative district reference, the *getKML* method is being called. The method constructs a query that will run over Stabon's endpoint and will fetch the geospatial information in KML form.** (Figure 4.5)
   (b) DAO calls method *getCitationsById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's citations.
   (c) DAO calls method *getById* connects with Sesame Server to apply SPARQL query for retrieving the requested legal document's modular elements their structure, text, etc.
   (d) DAO calls method *getAllModifications* connects with Sesame Server to apply SPARQ query for retrieving the requested legal document's modifications. It accepts a list of modifications with 2 main elements: modification's URI and modification's Source File. For each modification in the list, query the modification's Source File and get as a result modification's content.
4. Model LegalDocument's method *applyModifications* is being called to apply the newly fetched list of modifications to our requests legal document's divisions, which have to be modified (replaced, enriched, deleted or added).
5. *LegislationService* returns the request package in POJO form and the Controller finally depending on the request's nature responds via the suitable View JSP to serve the HTTP request in the desirable media type form (HTML,XML,PDF etc.).

```
// HTTP GET request
private String getKML(String place) {

        EndpointResult result = null;
        Properties props = new Properties();
        InputStream fis = null;
        String host = "";
        int port = 0;
        String appName = "";
        try {

            fis = getClass().getResourceAsStream("/properties.properties");
            props.load(fis);

            // get the properties values
            host = props.getProperty("StrabonHost");
            port = Integer.parseInt(props.getProperty("StrabonPort"));
            appName = props.getProperty("StrabonAppName");
        } catch (IOException e) {
            e.printStackTrace();
        }
        String KML = "";
        String query = "SELECT ?geo WHERE {<"+place+"> ?hasgeometry ?geo. ?hasgeometry <http://www.w3.org/2000/01/rdf-schema#label> \"has_geometry\"@en.}";

        SPARQLEndpoint endpoint = new SPARQLEndpoint(host, port, appName);

        try {

            result = endpoint.query(query, stSPARQLQueryResultFormat.KML);

            System.out.println("Status code: " + result.getStatusCode());
            System.out.println("Query: " + query);
            System.out.println("Status text: " + result.getStatusText());
            KML = result.getResponse().replace("<?xml version='1.0' encoding='UTF-8'?>", "");
            KML = KML.replaceAll("\n", "");
            KML = KML.replaceAll("\r", "");
            System.out.println("<----- Result ----->");
            System.out.println(KML);
            System.out.println("<----- Result ----->");

        } catch (IOException e) {
            e.printStackTrace();
        }

        return KML;

    }
```

Figure 4.5: *getKML* method in *LegalDocumentDAOImpl*

4.4.4 Use Case 4 - Rest Services

We keep on stating all over this project, that Nomothesi@ is a RESTful API[1]. One of its goals is to serve a series of Web Services over Greek Legislation to encourage further and more specialized projects. In this Section we will be presenting these services, so that we can underline the benefits of working with one single RDF data model through Sesame Server, hoping to inspire other projects to adopt this method, shaping a simpler Semantic Web. In our API one can request any legal document in PDF, XML, RDF, HTML and JSON format, as well as its enacted version or its updated one or even a specific version based on a date (see Section 3.2). In terms of technical engineering the RDF data model contributed in a very determinant way. We eliminated the unnecessary calls to the database and at the same time with RDF text annotations we managed to separate the text of a legal document based on the request. For example when the requested legal document has to be in HTML form, we are fetching the “@html” type text from Sesame so our text has its references in other legal documents etc. When the request demands a media format for reading purposes or just plain text representation without semantic information in it (PDF, XML etc.), we are fetching plain text type from Sesame “@el”, we can take a look at Table 4.1. In this way we do not only keep the complexity with requests low (one single query to fetch everything and then decide what to use), but also we presenting an expandable model which can very easily adopt more languages in the future.

[1] See `http://en.wikipedia.org/wiki/Representational_state_transfer`

Table 4.1: Request driven text

URI type	Media format	Fetched text
.../pd/2011/54/data.pdf	PDF	”Τις διατάξεις του Π.Δ. 189/2009 ”Καθορισμός και ανακατανομή αρμοδιοτήτων των Υπουργείων” (Α΄ 221).”@el.
.../pd/2011/54	HTML	”Τις διατάξεις του ¨ Π.Δ. 189/2009 ”Καθορισμός και ανακατανομή αρμοδιοτήτων των Υπουργείων” (Α΄ 221).”@html.

So no matter what the requested URI is, the method that will be called follows the same procedure. The only differences between the requests and media types are the final methods that shape the model's returned data type. We can see the differences in Table 4.2. The below mentioned methods are those we examined in the use case of Section 4.4.1.

Table 4.2: Method calls on REST Services

URI type	*getMetaDataByid*	*getCitationsByid*	*getByid*	*getAllModifications*	*applyModifications*	*Final construct method*
PDF	●	●	●	●	●	PDF-Builder
XML	●	●	●	●	●	XML-Builder
JSON	●	●	●	●	●	-
HTML	●	●	●	●	●	-
RDF	❍	❍	❍	❍	❍	getRDF-ById

Some conclusions reading Figure 4.2 are, that our architecture is type free which is the key to REST Services. We have one unified request-respond architecture, which differentiates from each type only with the final respond contracting method. That proves once again that RDF data models and Spring MVC are the most suitable tools for a RESTful API like Nomothesi@. We avoided creating long and messy methods for each media type and we ended with an expandable, easy and intelligent sequence of request building methods.

Chapter 5

Demonstration of Nomothesi@ Web Platform

Nomothesi@ comes with an intuitive interface for presenting legal documents, their content, and also the time-line of their evolution in terms of modifications. It also offers three ways for retrieving legal documents: 1) a search form for searching legal documents based on keywords and other co adjutant elements (e.g. date, id, and government gazette sheet), 2) a SPARQL endpoint which can be used by more advanced users for posing SPARQL queries and retrieving legal documents in RDF, and 3) a RESTful API. The second and third ways of retrieving legal documents gives the opportunity to a programmer for direct open access to the knowledge base of Nomothesi@, which can be employed to consume the legislative work of a government into applications and combining it with other resources on the web so as to increase its value. Therefore, third party developers can build applications around legal or generally public interest [8]. Finally in this Chapter we are going to present images of the interface of our REST API Nomothesi@[1].

First we present the Home Page of Nomothesi@ (Figure 5.1). Home Page provides a search form for legal documents, two tables, one for most requested acts and one for new legislation and an introduction text about our web application. The user can also download our OWL Ontology and Dataset.

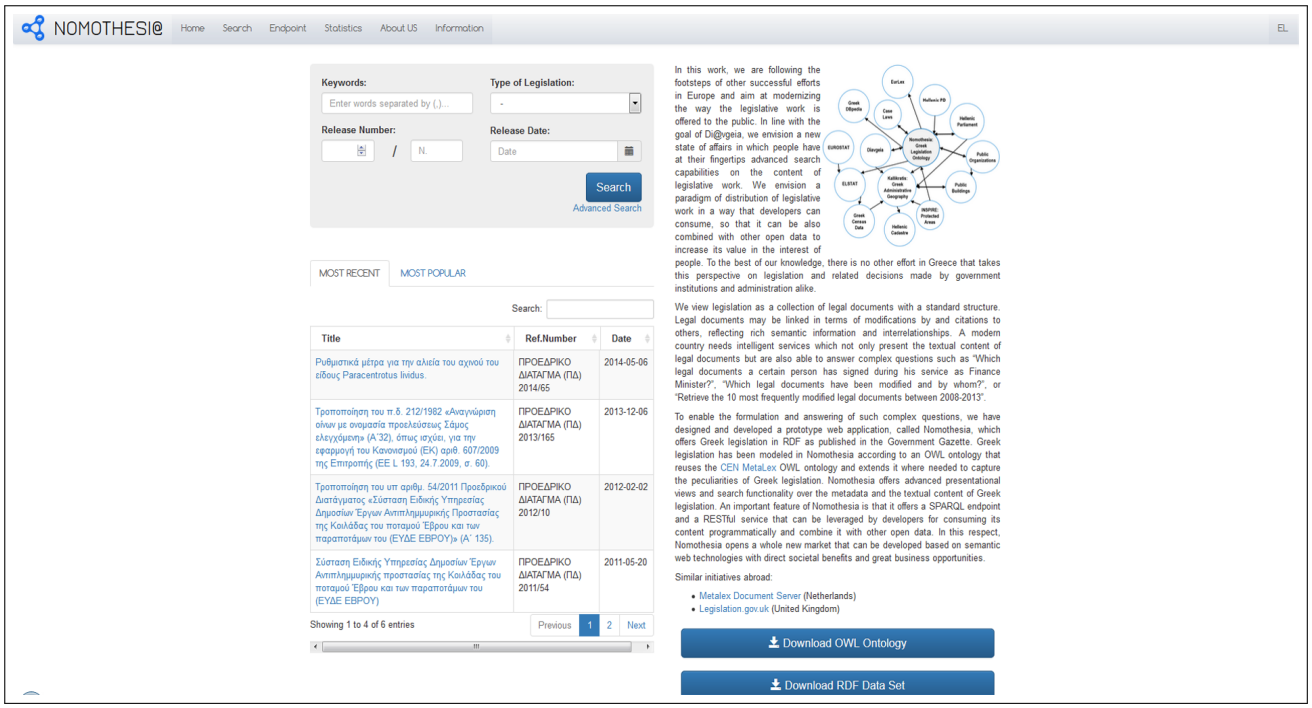

Figure 5.1: Home Page of Nomothesi@

[1]Nomothesi@ platform is demonstrated with a small data set at `http://legislation.di.uoa.gr`.

The API offers an advanced search page, for more complex searches. As we can see one can search a legal document based on keywords, type of legislation, release number, FEK number (publication number of Government Gazette), release date or finally a time space between two different dates (from-to). Below (Figure 5.2) we can see the results after searching for all Presidential Decrees (PDs) from 01-01-2010 to 01-01-2015.

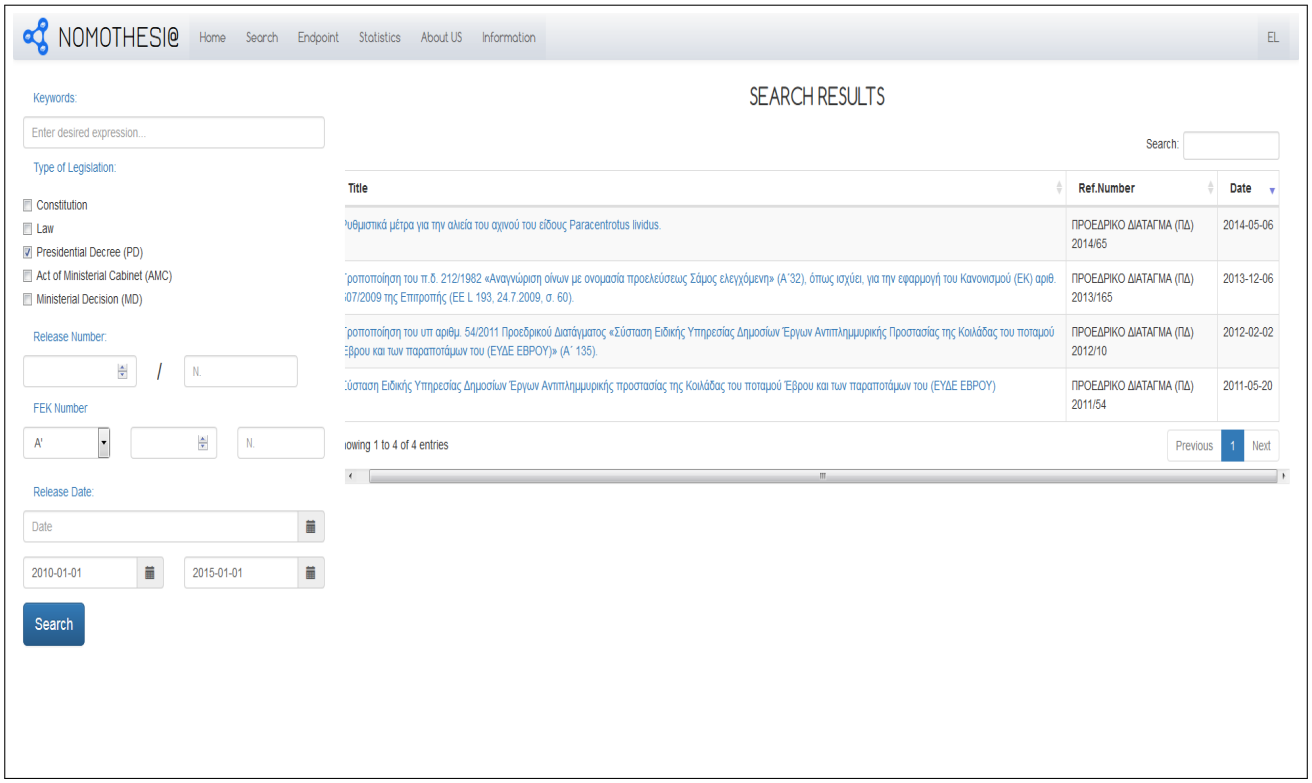

Figure 5.2: Page of Search Results

In Figure 5.3, we can see the updated and last available version of PD 2011/54. For the first time in Greek Legislation the reader can examine a legal document and recognize the modifications that have been applied on it (green color), but at the same time the reader has the ability to see the original text that has been modified (red color). This is very important in terms of reader-friendly legislation. On the left side we are serving general information on the specific legal document. Another innovating element is the existence of the map which is the visual outcome of the geospatial information that we serve in a semantic way with Strabon (see Section 4.3). The legal document is also available in its enacted version and five different formats (XML, PDF, RDF, JSON). The citations of the legal document are being displayed in a separate tab (Figure 5.4) and finally under the tab ”TIMELINE” (Figure 5.5) we are serving the time evolution of modifications and more specific information on them (type, target, text).

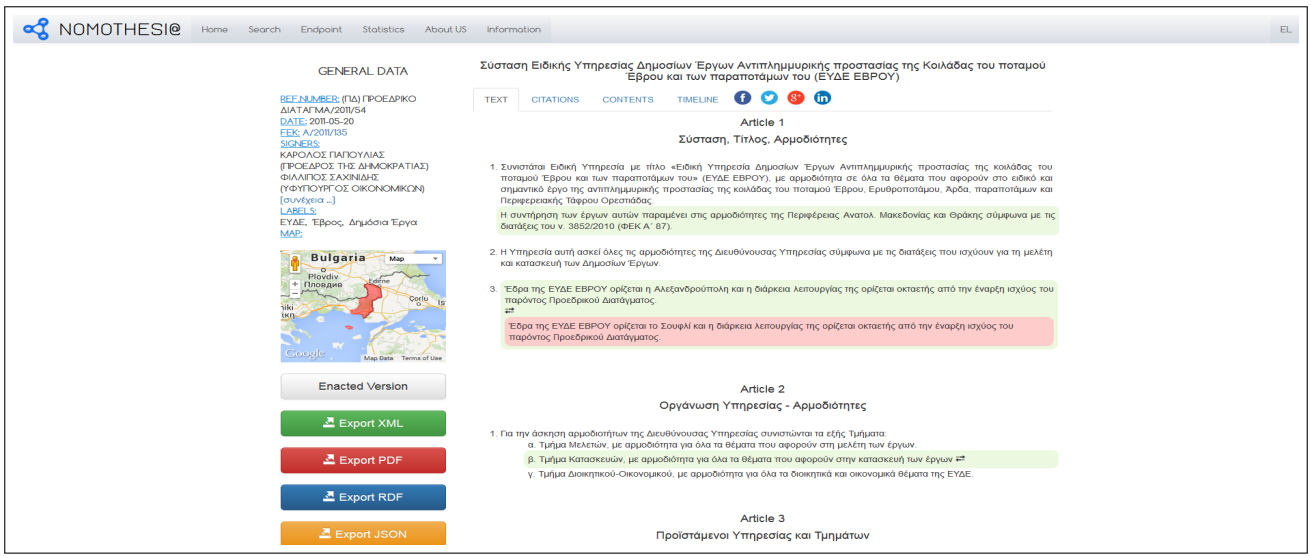

Figure 5.3: HTML Page of Latest version of PD 2011/54

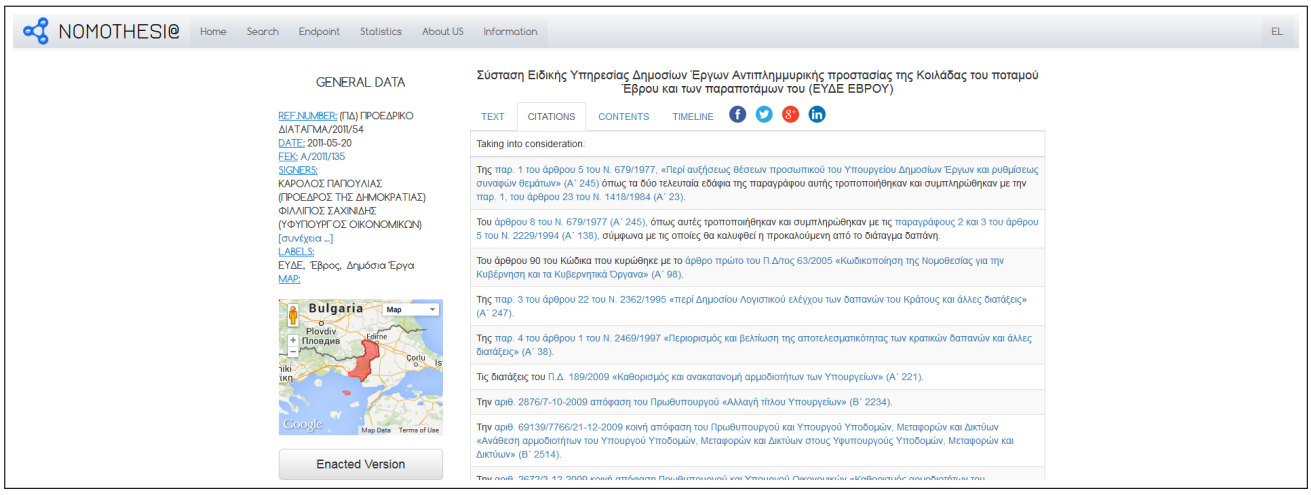

Figure 5.4: HTML Page of Citations of PD 2011/54

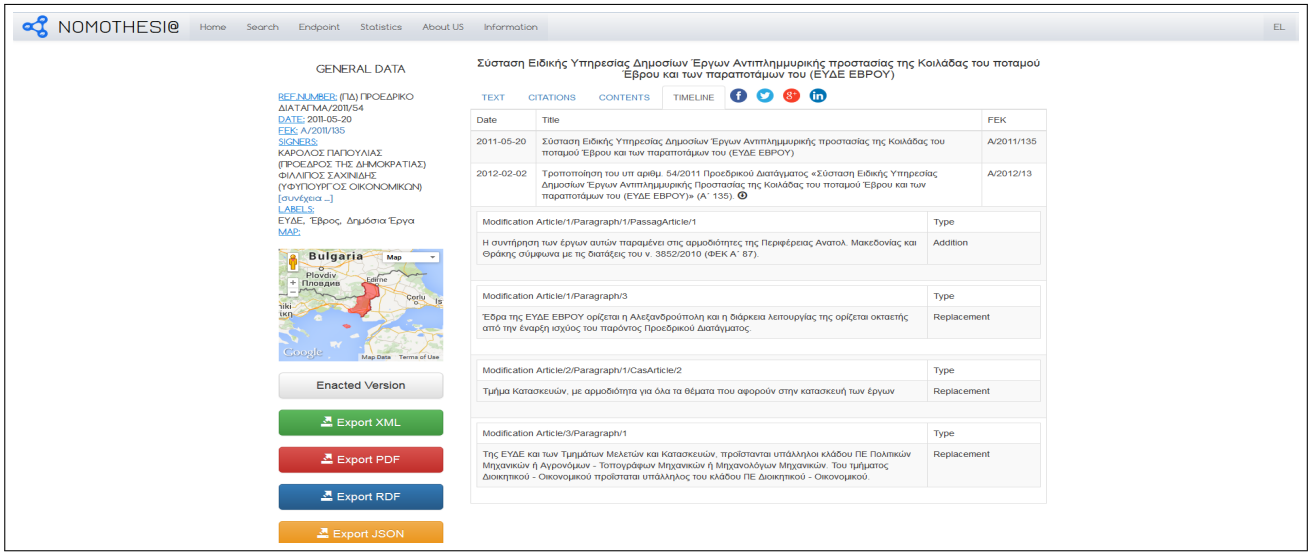

Figure 5.5: HTML Page of Modifications of PD 2011/54

Finally for this sort presentation, we shall finish with an example of our main REST Service which is the SPARQL Endpoint (see Section 2.2.2). The user can absorb endless information about a legal document in two formats (HTM and SPARQL/XML) and reuse this information for more complex projects. Below (Figure 5.6) we can see the example of querying our dataset with the query ”Show all modifications relating to PD 2011/54” and the outcome in its two formats.

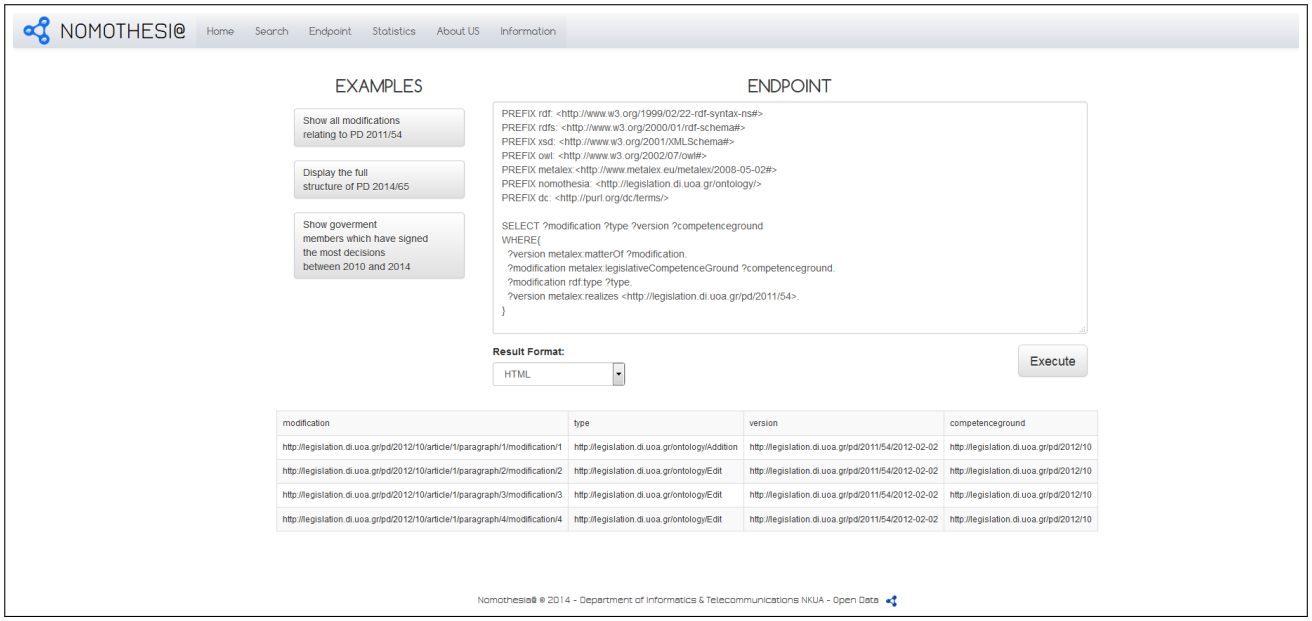

Figure 5.6: SPARQL query example in HTML format

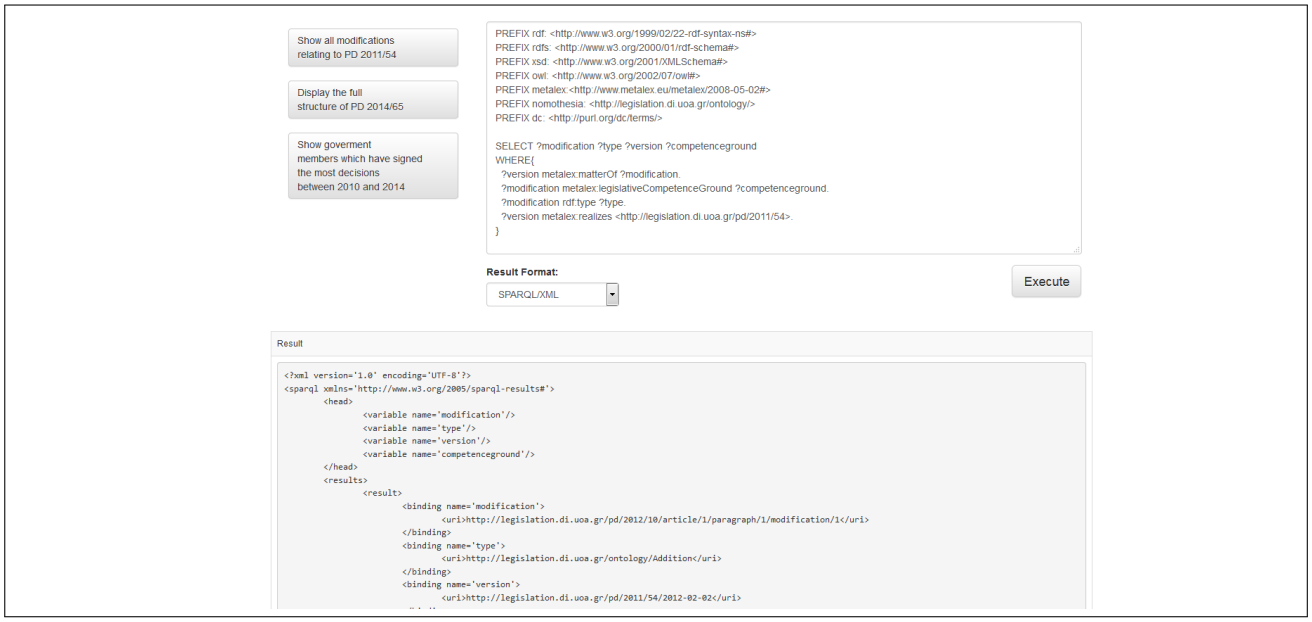

Figure 5.7: SPARQL query example in SPARQL/XML format

Chapter 6

Conclusions

In this work, we demonstrated how semantic web technologies can be employed to model Greek Legislation. Having as a basis the CEN MetaLex ontology, we developed an OWL ontology that models the structural characteristics of Greek legal documents as well as the processes that are used for modifying them. Taking as examples the Presidential Decrees 2011/54 and 2012/10, we expressed them in RDF following the developed ontology and showed how we can query the resulting RDF data using the query language SPARQL. These technologies have been implemented in our web platform Nomothesi@ that offers advanced search capabilities for retrieving legal documents in a number of data formats, as well as a SPARQL endpoint and a RESTful API that can be used by developers to consume and combine this data with other open data. The greatest challenge of our project is to apply our modeling to the whole legislative work of Hellenic Parliament. Another direction that we will pursue is the linking of Greek legislation with other government data that has recently been made available as linked data [12]. We believe that there is a whole new market that can be developed based on the technologies that Nomothesi@ brings into play in this area of open and public data, which offers great business opportunities. Despite the research presented in this work and the fact that we have demonstrated the effectiveness of such a platform for Greek Legislation, we believe that it could be further developed in a number of ways, that are being described below.

6.1 Future Work

Legal Documents Converter

As we mentioned, Greek legal documents are stored as text files (.doc) and published as PDF files (.pdf). In order to apply our platform in the real world, we need to enrich our data with as many legal documents as possible. Of course, it cannot happen manually, so we need a tool (converter) to accomplish the hard and lengthy job. In order to transform legacy XML to Metalex and RDF, Metalex research group implemented the MetaLex converter, an open source Python script [13]. Based in the same direction, we have already began developing a converter of our own, based on Greek Legislation's characteristics. We are currently trying to fully understand and produce a grammar that describes the behavior of legal documents' writing, in order to be able to parse the excising official legal documents as they are being served from the Government Gazette in their electronic form (PDF). Then we shall be able to extract all the information we need and integrate it into our RDF datasets.

Populate Data

Currently, Nomothesi@ platform serves only a dozen of legal documents. If we want to capture the whole legislative process coming through the Hellenic Parliament, we have to populate our datasets with as much legal documents as we can. To do so, we have to take advantage of the above mentioned converter. Populating our datasets will also be a great

push towards the modernization of Greek e-Government and add even more transparency and consistency in the law making process.

Core Vocabularies

The last three years, the Core Vocabularies Working Group of the Interoperability Solutions for European Public Administrations (ISA) Programme has the objective to promote the specification of the Core Person and develop the specification of the Core Business and Core Location vocabularies. These vocabularies will promote interoperability through different data sets, which are published by many organizations across EU. With cautious circumspection in the great issue of interoperability, we will use these vocabularies to redefine and enrich our data sets.

Greek e-Government Linked Data

A key objective of our project is to link our data sets with any other e-Government data sets, such as Public Services and Organizations and Government staff (See Figure 6.1). The adoption of Core Vocabularies will accelerate this effort [14]. Linking Data has to be both in RDF/OWL metadata level and also in content's markup. Marking and linking all available knowledge in legal documents' content widen the whole concept of our services' quality and usability. This specific aspect is one of the most important that undergoes in UK legislative services and it has great interest for the working group of The National Archives (UK).

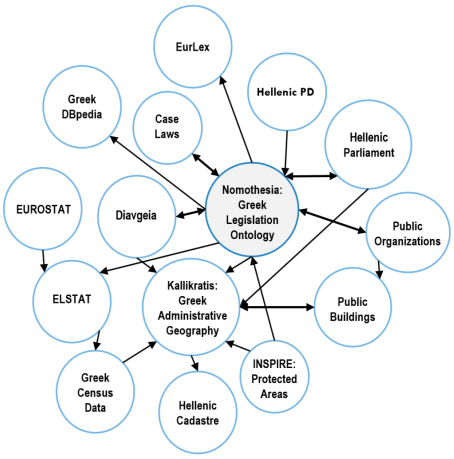

Figure 6.1: Part of Future Greek Linked Data Cloud Diagram

Local Government Decisions

In our project, we worked with the core of Greek Legislation (Laws, Presidential Decrees, Acts of Ministerial Cabinet, Legislative Acts and Ministerial Decrees). Other secondary legal documents have to be handled in the same way. Something really innovative is to include Local Government Decisions. Greek Local Government consists of 7 decentralized administrations, 13 regions and 325 municipalities. Each of them publish Local decisions in Government Gazette and there is a great interest in all these paralegal documents. The first step was made with this work, as we managed to enrich our metadata with geospatial information from Strabon. Research groups in our department have published an ontology of Kallikratis Plan, the administrative system of Greece [15].

Table 6.1: LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CEN	European Committee for Standardization
CS	Computer Science
ELI	European Legislative Identifier
EU	European Union
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
OWL	Web Ontology Language
REST	REpresentational State Transfer
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
URI	Universal Resource Identifier
XML	Extensible Markup Language

Table 6.2: TABLE OF TRANSLATIONS

Artificial Intelligence (AI)	Τεχνητή Νοημοσύνη
E-Government	Ηλεκτρονική Διακυβέρνηση
Legislative Knowledge Representation	Αναπαράσταση Γνώσης Νομοθεσίας
Metadata	Μεταδεδομένα
Open Data	Ανοιχτά Δεδομένα
Semantic Web	Σημασιολογικός Ιστός
URIs	Καθολικά Αναγνωριστικά Πηγής

Appendix A

Nomothesi@ Web Application

Nomothesi@ is implemented as a Spring web MVC Project. The Spring web MVC framework provides Model-View-Controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

Software / Library dependencies

Java Version: Java™ SE 7 (or greater)
Spring Framework 4.x
Build Tool: Apache Maven 3.1.1
Web Server: Apache Tomcat 7 (or greater)
RDFStore: Sesame Server 2.7.9 (on Apache Tomcat 7)
Strabon: Strabon-Endpoint-Client 3.2.10
iTextPDF: iTextPDF 5.5.3

Installation instructions

Nomothesi@ platform is compressed in nomothesia-1.0.0-BUILD-SNAPSHOT.war file. The .war file in folder /webapps needs to be deployed in a Web Application Server (Apache Tomcat) and also collaborates/communicates with two external web applications Sesame and Strabon, Sesame is a server sided so it also needs to be deployed. We will give instructions of how-to complete installation. After completing installation, you can browse Nomothesi@ platform at `http://localhost:8084/legislation/` locally or you can visit a deployed version at `http://legislation.di.uoa.gr/`.

1. Apache Tomcat (`http://tomcat.apache.org/`)

Apache Tomcat can be downloaded here (`http://tomcat.apache.org/download-70.cgi`), depends to the operating system and its distribution. You can find installation instructions here (`http://tomcat.apache.org/tomcat-7.0-doc/setup.html`). After installation, you can point your browser at this location (http://localhost:8080/) to verify that the deployment succeeded. You can find information of how-to deploy an application here:
(`http://tomcat.apache.org/tomcat-7.0-doc/deployer-howto.html`)

2. Sesame Server (`http://rdf4j.org/`)

Sesame Server can be downloaded here (`http://sourceforge.net/projects/sesame/files/Sesame%202/2.7.11/openrdf-sesame-2.7.11-sdk.zip/download`). You can also find the .war files in /webapps folder. The .war files needs to be deployed in Apache Tomcat. The easiest way to deploy this application is to place the .war files in webapps folder of Apache

Tomcat and restart it. After you have deployed the Sesame Server webapp, you should be able to access it, by default, at path /openrdf-sesame. You can point your browser at this location to verify that the deployment succeeded. You can also manage repositories from the OpenRDF Workbench, which should be available at path /openrdf-workbench.

For the purposes of our project, you need to visit /openrdf-workbench create a new repository called “Legislation” with type Memory store / persist or Native store and then load our data set from files legislation.owl and legislation.n3. You can complete any appropriate action from OpenRDF Workbench’s menu.

Any additional information of downloading, installing and interacting with Sesame Server can be found in the website’s user’s documentation.

3. Strabon Endpoint Client (`http://www.strabon.di.uoa.gr/`)

Strabon spatiotemporal RDFStore can be downloaded here (`http://www.strabon.di.uoa.gr/download/`). The easiest way to connect with Strabon Endpoint as a client is to place the dependency .jar in the classpath of your application. Further installation information can be found here (`http://strabon.di.uoa.gr/installation-windows`).

Any additional information for downloading, installing and interacting with Strabon can be found in the website’s user’s documentation. There is also some useful information at (`http://www.strabon.di.uoa.gr/files/stSPARQL_tutorial.pdf`), which really helps to understand how it works.

4. iTextPDF (`http://itextpdf.com/`)

iText is an open source library that allows us to create and manipulate PDF documents. It enables us to enhance our web application with dynamic PDF document generation and/or manipulation. iText is used as any other Java library. There is no installation, just put the jar in the classpath of your application application. You can download iTextPDF .jar here (`http://sourceforge.net/projects/itext/files/latest/download`)

Any additional information for interacting with iText can be found in the website’s free books and a list full of examples can be found here (`http://itextpdf.com/book/examples.php`).

Configuration instructions

There is a possibility that you need to configure Nomothesi@. For this purpose there is a configuration file called properties.properties, which specify the following information:

SesameServer = (Sesame Server path)
SesameRepositoryID = (Sesame Server repository name)
StrabonHost = (Name of Strabon Host)
StrabonPort = (Sesame Server port)
StrabonAppName = (Sesame Server application name)

An example of the .properties file would be the following:

SesameServer = http://localhost:8080/openrdf-sesame
SesameRepositoryID = legislation
StrabonHost = geo.linkedopendata.gr
StrabonPort = 80
StrabonAppName = gag-endpoint/Query

Bibliography

[1] Ilias Chalkidis. Nomothesi@: Greek Legislation Platform. Technical report, National and Kapodistrian University of Athens, 2014.

[2] María Hallo Carrasco, M. Mercedes Martínez-González, and Pablo De La Fuente Redondo. Data models for version management of legislative documents. *J. Inf. Sci.*, 39(4):557–572, 2013. http://jis.sagepub.com/content/39/4/557.abstract.

[3] Rinke Hoekstra, Leibniz Center for Law, Universiteit van Amsterdam and VU University Amsterdam. The MetaLex Document Server, 2014. http://doc.metalex.eu.

[4] A. Boer, R. Hoekstra, R. Winkels, T. van Engers, and F. Willaert. ^{META}lex: Legislation in XML. In T. Bench-Capon, Aspassia Daskalopulu, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems. Jurix 2002: The Fifteenth Annual Conference*, Frontiers in Artificial Intelligence and Applications, pages 1–10, Amsterdam, 2002. IOS Press. http://www.jurix.nl/pdf/j02-01.pdf.

[5] John Sheridan. LEGISLATION.GOV.UK, 2010. http://blog.law.cornell.edu/voxpop/2010/08/15/legislationgovuk/.

[6] W3C. Linked Data, 2015. http://www.w3.org/standards/semanticweb/data.

[7] Central Codification Committee of Goverment's SG. *Manual directives for encoding of legislation*, 2003. http://www.ggk.gov.gr/wp-content/uploads/2010/02/teliko_egxeiridio_odigion_gia_tin_kodikopoiisi_tis_nomothesias.pdf.

[8] Ilias Chalkidis, Charalampos Nikolaou, Panagiotis Soursos, and Manolis Koubarakis. Modeling and Querying Greek Legislation using Semantic Web Technologies. Technical report, National and Kapodistrian University of Athens, 2015.

[9] J. Eder, I. Rozman, and T. Welzer. *Advances in Databases and Information Systems: Third East European Conference, ADBIS'99, Maribor, Slovenia, September 13-16, 1999, Proceedings*. Advances in Databases and Information Systems: Third East European Conference, ADBIS'99, Maribor, Slovenia, September 13-16, 1999 Proceedings. Springer, 1999. https://static.aminer.org/pdf/PDF/000/026/087/evaluation_of_data_modeling.pdf.

[10] Rinke Hoekstra. The metalex document server - legal documents as versioned linked data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (2)*, volume 7032 of *Lecture Notes in Computer Science*, pages 128–143. Springer, 2011. http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/In-Use/70320129.pdf.

[11] Rich Jelliffe. PRESTO - A WWW information architecture for legislation and public information systems, 2008. http://www.oreillynet.com/xml/blog/images/PRESTO.pdf.

[12] Evangelos Kalampokis, Efthimios Tambouris, and Konstantinos Tarabanis. On publishing linked open government data. In *Proceedings of the 17th Panhellenic Conference on Informatics*, PCI '13, pages 25–32, New York, NY, USA, 2013. ACM.

[13] Alexander Boer, Radboud Winkels, Tom van Engers, and Emile de Maat. A content management system based on an event-based model of version management information in legislation. In T. Gordon, editor, *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference.*, Frontiers in Artificial Intelligence and Applications, pages 19–28, Amsterdam, 2004. IOS Press. http://jurix.nl/pdf/j04-04.pdf.

[14] ISA programme. eGovernment Core Vocabularies, 2012. https://joinup.ec.europa.eu/sites/default/files/a7/3d/18/ISA_eGovernment-Core-Vocabularies_February2012.pdf.

[15] Greek Linked Open Data: Greek Administrative Geography. http://www.linkedopendata.gr/dataset/greek-administrative-geography.