Requirements Engineering manuscript No.
(will be inserted by the editor)

GaiusT: Supporting the Extraction of Rights and Obligations for Regulatory Compliance

Nicola Zeni · Nadzeya Kiyavitskaya · Luisa Mich ·
James R. Cordy · John Mylopoulos

Received: / Accepted:

Abstract Ensuring compliance of software systems with government regulations, policies and laws is a complex problem. Generally speaking, solutions to the problem first identify rights and obligations defined in the law and then treat these as requirements for the system under design. This work examines the challenge of developing tool support for extracting such requirements from legal documents. To address this challenge, we have developed a tool called GaiusT[1]. The tool is founded on a framework for textual semantic annotation. It semi-automatically generates elements of requirements models, including actors, rights and obligations. We present the complexities of annotating prescriptive text, the architecture of GaiusT, and the process by which annotation is accomplished. We also present experimental results from two case studies to illustrate the application of the tool and its effectiveness relative to manual efforts. The first case study is based on the US Health Insurance Portability and Accountability Act (HIPAA), while the second analyzes the Italian accessibility law for information technology instruments.

Keywords semantic annotation · legal documents · requirements engineering · regulation compliance problem · legal requirements · multilingual annotation.

1 Introduction

Globalization has amplified the problem of internationalizing and localizing software systems to ensure that they comply with both international and local regulations. Different international policies, laws and regulations, written in a wide range of languages even within one jurisdictional entity – such as the European Union – together with privacy and security requirements, pose serious challenges for software system developers world-wide. In particular, IT professionals have to face the so-called regulation compliance problem, whereby companies and developers are required to ensure that their software systems comply with relevant regulations, either by design or through re-engineering [3].

Legal documents are written in a jargon colloquially referred to as *legalese*, a specialized language that makes the elicitation of requirements from regulations a difficult task for

Address(es) of author(s) should be given

[1] Named after GaiusTerentilius Harsa, a plebeian tribune who played an instrumental role in establishing for the first time in ancient Rome a formal code of laws through the Twelve Tablets (462BC).

developers who lack proper legal training. Misinterpretation – for example by overlooking an exception in a regulatory rule – can have legal consequences, as it may lead to incorrect assignment of rights or obligations to stakeholders. Elicitation of requirements is generally achieved in a manual and often haphazard way. Breaux, Vail and Antón [13] propose a systematic manual process for extracting rights and obligations from legal text. Our work seeks to augment this manual process with tool support, thereby improving productivity, as well as quality and consistency of the output.

To accomplish this complex task, we use *semantic annotation* (SA) techniques, where legal text is annotated on the basis of concepts that have been defined in terms of a conceptual schema (aka ontology). Such techniques are used widely in the field of the Semantic Web to generate semi-structured data amenable to automated processing from unstructured web data. Such techniques have also been fruitfully applied in Requirements Engineering to support partial automation of specific steps in the elicitation process [29].

This paper presents the details of an SA-based tool for requirements extraction from legal documents. We discuss related issues and propose an engineering approach founded on an existing framework for SA called *Cerno* [32]. This framework has been adapted and extended to deal with many of the complexities of legal documents and the result is a new tool named *GaiusT*. The main objective of this paper is to present GaiusT along with its evaluation on regulatory documents written in two different languages: the U.S. Health Insurance Portability and Accountability Act [56] (HIPAA, in English) and the Italian accessibility law for information technology instruments (known as the Stanca Act, in Italian) [26]. The novelty of our contributions rests in identifying a combination of techniques adopted from research on semantic annotations, legal document processing and software reverse engineering which together demonstrably improve productivity on extracting requirements from legal text.

These contributions expand on an earlier paper [30], where we first outlined our preliminary research plan, along with a first application of GaiusT. In this work we focus on the architecture and implementation of GaiusT and expand on the experimental results reported in [30] based on two case studies.

The rest of the paper is organized as follows. Section 2 describes the challenges of analyzing prescriptive documents. In Section 3 we present the research baseline of this work, while Section 4 highlights requirements for analyzing legal documents. In Section 5 we propose a design for GaiusT to address these requirements, and elaborate on its architecture and application from a user-centered perspective in Section 6. In Section 7 we evaluate our proposal by applying the tool to two case studies and report the results of experiments comparing the tool's analysis of these laws with that of human experts. Finally, we provide an overview and comparison with other methods and projects in the area in Section 8, and draw conclusions in Section 9.

2 Complexities of Prescriptive Documents

Analysis of legal documents poses a number of challenges different from those found in other documents, largely because of the special traditions and conventions of law. In this section we report on some of the most distinctive characteristics of legal documents, focusing in particular on laws and regulations. The handling of these characteristics forms a part of the requirements for the GaiusT system.

The most distinctive characteristic of legal documents is that they are *prescriptive* [37], and are written accordingly. This means that they express the permission of desirable and/or the prohibition of undesirable behaviors and situations. The norms that they convey are requirements that control behaviors, or possible states of affairs in the world. Consequently, each norm has a core part called the *norm-kernel* which carries the essential information conveyed by the norm. This information includes: a *legal modality*, a *subject*, an *object*, *conditions of applicability* and *exceptions*. The *legal modality* determines the function of a norm, which can be either an *obligation* or a *right* [60]. A *right* is an action that a stakeholder is conditionally permitted to perform, an *obligation* is an action that a stakeholder is conditionally required to perform. In contrast, an *anti-right*/*anti-obligation* states that a right/obligation does not exist. The *subject* of a norm is the person or institution to whom the norm is directed. The *object* of a norm is the act regulated by the norm, considering the modality and other properties of the action. The *conditions of applicability* establish the circumstances under which a norm is applicable. These are clauses that suspend, or in some way modify, the principal right or obligation. In addition to these, each law contains general rules which prescribe a norm for a large group of individuals, actions or objects, followed by a rule that excludes specific members or subsets of these groups. Often, conditions of applicability are modified by *exceptions* both for subjects and objects [44]. *Exceptions* can be introduced explicitly through words such as *except* or *contrary to*, or implicitly when they have to be inferred. For example, in “A medical savings account is exempt from taxation under this subtitle unless such account has ceased to be a medical savings account.”, the use of the term *unless* indicates an exception. One or several exceptions related to a right can be formulated directly in a section mentioning this right, or indirectly in another part of the law, or even in another law using cross-references. To support the markup of exceptions, we have identified a set of terms that are commonly used to express exceptions (see Table 1).

Cross-references create links between parts of the same document (internal) or with parts of other documents (external), and are essential to the interpretation of laws. To unambiguously identify the parts of a document and to allow for explicit references to related parts, legal documents are organized using a deep *hierarchical structure*.

To facilitate the understanding of a law, every legal document contains a declarative part – the *definition of legal terms* – that defines legal concepts used in the document. This constitutes the glossary section of a legal document, and plays a fundamental role in its interpretation. The lack of such a declarative part can result in ambiguity and misinterpretations.

All of these features pose serious challenges for semantic analysis. In fact, although the vocabulary and grammar used in such documents is often restricted, inconsistent use of terms is pervasive, mainly due to the length and complexity of the lifetime of legal documents. Moreover, it is often the case that different parts of a legal document are written by different people, either because of the sheer size of the document, or as a result of subsequent modifications.

Another problem dimension arises from the *multilinguality* of laws applicable in a given jurisdiction (such as the European Union). Analysis of regulations written in different languages entails handling the different structural, syntactic and semantic issues of each particular language.

Furthermore, the requirements obtained from several regulations must be integrated, aligned and ranked according to the priority of the regulations they were extracted from.

Finally, inaccuracy of the analysis of legal documents must be minimal because in the eyes of the law, there is no such thing as “near-compliance”. For example, by missing an exception or a condition, one could bestow or deny an important right or obligation to the wrong party.

3 Research baseline: Cerno

In light of the complexity of legal documents outlined above, we propose a tool-assisted process for their annotation, which includes the following main activities: (i) *Structural analysis* to handle identification of structural elements of a document, basic text constructs and cross-references; (ii) *Semantic analysis* to identify normative elements of a document such as, rights, obligations, anti-rights, anti-obligations.

In order to realize this process, we have adopted and extended the semantic annotation framework Cerno [31], and used it as a foundation for our work. Cerno is based on a lightweight text pattern-matching approach that takes advantage of the structural transformation system TXL [16]. The types of problems that Cerno can address are similar to those faced in the analysis of legal documents, such as the need to understand of the document's structure and the need to identify concept instances present in the text. Moreover, experimentation with Cerno has shown it to be readily adaptable to different types of documents [31,33,63].

In Cerno, textual annotations for concepts of interest are inferred based on a conceptual model of the application domain. Annotation rules can be automatically or manually constructed from this model. Such an approach factors out domain-dependent knowledge, thus allowing for the development of a wide range of semantic annotation tools to support different domains, document types, and annotation tasks.

Cerno's annotation process has three phases: (1) *Parse*, where the structure of an input document is processed, (2) *Markup*, where annotations are generated based on annotation rules associated with concepts of the conceptual model, and (3) *Mapping*, which results in a relational database containing instances of concepts, as identified by annotations.

The goal of the *Parse* step is to break down an input document into its structural constituents and identify interesting word-equivalent objects, such as e-mail addresses, phone numbers, and other basic units. Similarly to Cerno's Parse step, structural analysis of legal texts identifies both structural elements of the document, and word-equivalent objects. These are basic legal concepts, such as legal actors, roles, procedures and other entities normally defined in the Definitions section of the law.

Next, the *Markup* phase conducts Semantic analysis. An annotation schema specifies the rules for identifying domain concepts by listing concept identifiers with their syntactic indicators. These indicators can be single words and phrases, or names of previously parsed basic entities. The annotation schema can be populated with such syntactic indicators either through (semi-)automatic techniques that use keywords of the conceptual model – if available – or hand-crafted from a set of examples.

Lastly, the *Mapping* phase of Cerno is optional and depends on how one proposes to use generated annotations. In the analysis of legal documents, we assume that a human analyst may want to correct or enrich tool-generated results. In this setting, annotations must remain embedded in the documents so that the user can easily revise them using textual context. After this revision, the annotations can be indexed and stored in a database.

Although the basic steps in Cerno clearly apply well to our problem, the annotation process we envision for legal documents required a significant adaptation effort at the design and implementation level in order to make the framework not only applicable in the legal domain, but also usable by requirements engineers – and possibly legal consultants. To this end we designed and developed the GaiusT system as an extension and specialization of Cerno that includes new components specific to the annotation of legal documents in different languages. To enhance the usability of the system, a GUI was added to better support human interaction.

The main extensions to Cerno are described next. Firstly, although Cerno was well adapted to the sequence-of-paragraphs form of English text – such as newspaper articles, web pages and advertisements, it could not handle the specialized semi-formal structure of legal texts. We therefore extended the TXL-based Cerno parser so that it can process the nested structural form of legal documents, including numbering and nesting of sections, paragraphs and clauses.

The labels of legal document sections are also context-dependent. For example, “clause (b) above” implicitly refers to the one in the current paragraph and section. To resolve this problem, the parser was enhanced with an additional TXL source transformation step to contextualize section, paragraph and clause names to make them globally unique. For example, the label of clause “(b)” in paragraph 2 of section 5 becomes “5.2(b)”. A subsequent reference resolution transformation was added to use these globally unique labels to link local and global references to the clauses, paragraphs and sections they refer to. For example, the reference “except as in (b) above” in paragraph 2 of section 5 is resolved to “except as in 5.2(b) above”. The set of Cerno word-equivalence patterns, which recognize items such as phone numbers, email addresses, web URLs and the like, was extended to recognize legal items such as regulation and law names (e.g., “42 U.S.C. – 405(a) (2006)”).

The set of concept patterns was also extended and enhanced to include terms and language for legal obligations, rights and other concepts, and patterns were added to recognize temporal clauses such as “within a month”.

Finally, Cerno’s word pattern specification notation was enhanced to process Italian regular and irregular verb forms in order to enable processing of Italian legal documents.

In the following section, we describe the design of the GaiusT tool considering both the structural and semantic analysis steps of the extended Cerno process.

4 Semantic Annotation of Legal Documents

4.1 Structural Analysis

A fundamental feature of legal documents is their deep hierarchical structure. Identifying document structure is important to semantic annotation for two main reasons: (i) recognition of potential annotation units, thus allowing for different annotation granularity, and (ii) flattening the document’s hierarchical representation, consisting of sections, subsections, etc. Annotation granularity can be defined as the fragment of text that refers to a concept instance, or an amount of context that must be annotated together with the concept. Different granularities can be used to address different objectives. In legal documents, for instance, single word granularity is convenient for annotating basic concepts, such as stakeholders or cross-references.

From a semantic viewpoint, the semi-formal structure of legal documents is used, to support the identification of complex deontic concepts, inferring a relationship between inferred concepts from their relative positions in the structure. The identification of structural elements in the process of semantic annotation also improves accuracy of the results, e.g., by treating a list as a single object for a given statement.

In general, any document has two levels of structure, a *rhetorical structure* and an *abstract structure*, both of which contribute to meaning [46]. According to Rhetorical Structure Theory [35], a document can be represented as a hierarchy where some text units are salient, while others constitute support. Abstract document structure, on the other hand, relies on graphical visualization of the text and includes syntactic elements used for text

Table 1 Document structure classification

Level	Text unit	What
0	Word	a minimal unit that can be annotated which is normally a sequence of non-spacing symbols
1	Phrase	a sequence of words delimited by punctuation symbols or conjunctions
2	Title	a sequence of words usually preceded and followed by one or more new lines
3	Sentence	a combination of two or more phrases delimited by a fullstop
4	List	two or more sentences that start with an itemized or enumerated list and end with a punctuation symbol
5	Paragraph	its start is indicated by a new line and the end is the beginning of another new line without running to the next passage
6	Section	composed of a headline followed by one or more paragraphs
7	Document	corresponds to the boundaries of the file

representation [20]. Both rhetorical and abstract document structures are important for the design of a structure analysis component.

Due to the variety of existing document formats and contexts, it is impossible to define a common structure valid for all documents, even though several efforts have been devoted to the problem [54]. There are several possible strategies to automatically capture the structure of documents [61,23]. The most widely used approach is based on feature-based boundary detection of text fragments. This approach exploits patterns, such as punctuation marks to identify text units. These patterns are typically composed of regular expressions and include non-printable characters such as carriage return for new line identification, symbol list or indentation for text blocks, graphical position of certain elements such as titles, etc. To capture the deep hierarchical structure of legal documents, we have defined a classification of document text units, using a hierarchy where a text unit at a given level is composed of one or more units at the next level down (Table 1). This classification facilitates the management of a large class of heterogeneously-structured documents with the only assumption that they are written in languages of the European family. Moreover, this classification can then be specialized according to the type of application and granularity level needed for a particular semantic analysis.

In GaiusT we have defined grammatical patterns, using TXL's ambiguous context-free grammar notation, to capture text units at different levels of this document hierarchy.

To capture the complex hierarchical structure and cross-references of legal documents, we adopted "The Guide to Legal writing" [55] that presents styles for proper writing and structuring of legal documents. Based on the hierarchy reported in the "The Guide to Legal writing" we specialized our grammatical patterns (see for example Tables 5 and 10). For cross-references, since loops in cross-references chains are intractable (see for example *HIPAA* paragraph 164.528(a)(2)(i)), our approach consists of recognizing and annotating cross-references, so that the annotator can take them into account and decide on a case-by-case basis. Special parsing rules catch both external and internal cross-references, and a TXL transformation renames all local labels and references to be globally unique. For example, the transformation changes list element label "(b)" of paragraph 2 of section 5 to "5.2(b)".

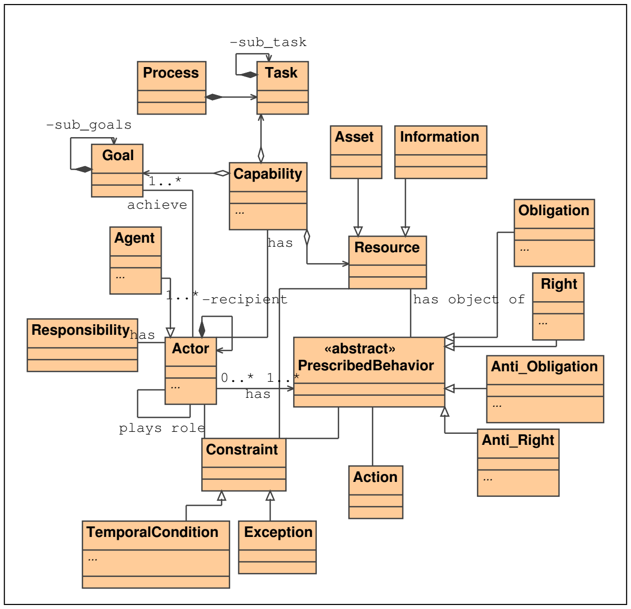


Fig. 1 The conceptual model of legal concepts

4.2 Semantic Analysis of Legal Documents

The semantic analysis of a document requires the development of a conceptual model that represents domain information, in the case of legal documents *deontic concepts*. To design the conceptual model, we used an iterative process. An initial list of important concepts in the domain of legal texts was derived from the "The Guide to Legal writing" [55] and *norm–kernel* concepts [60]. We then applied part of the Semantic Parameterization methodology developed by Breaux and Antón [9], which defines a set of guidelines (textual patterns) for extracting legal requirements from regulations. Our model was then discussed with legal experts and finalized based on their recommendations. The experts were colleagues of the Faculty of Law of the University of Trento, and they helped with the analysis and refinements of legal concepts/terms and their interrelations.

Table 2 Syntactic indicators for deontic concepts

Concept	Concept type and its indicators
Right	may, can, could, permit, to have a right to, should be able to
Anti-Right	does not have a right to
Obligation	must, requires, should, will, would, which is charged with, may not, can not, must not
Anti-Obligation	is not required, does not restrict, does not require

The resulting model, represented in Figure 1, includes as classes only the core deontic concepts. It leaves out others, such as *penalties*, that according to legal experts can be considered as properties of the core deontic concepts. The conceptual model was used as

Table 3 Syntactic indicators for *Exception*

Concept	Syntactic indicators and their occurences
Exception	only (733), but (588), except (547), without (514), limit (499), restrict (435), exclusion (369), other than (156), unless (127), release (105), not include (98), deny (84), eliminate (78), limited to (78), prevent (77), exempt (46), not apply to (41), to object (35), refuse (29), not in (26), objected (25), objection (22), outside of (20), omit (18), reject (16), omission (15), prelude (13), rule out (12), discharge (9), save (8), bar (5), relieve (5), exclude from (3), besides (2), not valid (2), aside from (2), leave out (1), restrict to (1), not listed (1), exonerate (1)

the basis for the definition of an annotation schema, as discussed in Section 6. Concepts of the conceptual model are used to populate the annotation schema, and for each concept a set of syntactic indicators and rules or patterns is defined. For the identification of prescribed behaviours – *rights*, *obligations*, and their antitheses – the annotation schema employs a set of manually constructed patterns, based on modal verbs (see Table 2). For identifying basic concepts – such as *actors*, *resources* and *actions* – we reuse the *Definitions* section of a legal document, where every used term is defined precisely and a reference synonym is assigned [55, 18].

Annotation of conditions of application, are represented in the conceptual model as *constraint* specialized into two sub-concepts, *exception* and *temporal condition*. A list of syntactic indicators can be generated using a combined approach that analyzes the content of legal document with the use of lexical databases such as WordNet[2] or a thesaurus[3]. In our case studies we manually built a list of syntactic indicators for temporal conditions and exceptions and integrated it with lists derived from lexical databases.

4.3 Multilingual Semantic Annotation of Legal Documents

Language diversity in the semantic annotation process is treated at three distinct levels: (i) Structural: a text available in multiple language translations uses different character sets, word splitting, orders of reading, layout, text units, and so on, (ii) Syntactic: the annotation patterns and rules that are used to identify concepts need to be expressed in the language. For example the syntactic indicators used for the concept of *obligation* (see Section 6) must be revised for each new target language; (iii) Semantic: the semantic model of legal concepts should have a language-independent representation, thus the syntactic indicators for each concept in different languages must be carefully defined.

The structural level has been investigated in the information retrieval (IR) area and basically amounts to segmenting text into words in different languages, using word stemming, morphology and the identification of phrase boundaries, using punctuation conventions. There are many tools for handling these issues. However, software developers often underestimate the value of syntactic aspects, while a successful text analysis tool must always cater to them. Issues related to this level are managed by the document structure analysis as described in Section 5.

The syntactic level relates to the fact that people of different nations express the same

[2] `http://wordnet.princeton.edu/`

[3] `http://thesaurus.reference.com/`

concept in different ways. For example, working with Italian legal documents we discovered that statements expressing an obligation normally use the present active or present passive tense, e.g., “the organization sends a request” or “the request is sent to the user”. This form differs from English construction of the same concept, where obligations are usually expressed with modal verbs “must” or “should”. However, syntactic differences do not affect the conceptual model built for the domain of legal documents since the concepts represented are valid also for other languages than English (Figure 1). In our case studies we used the *WFL Generator* and *Lexical Database Manager* components of GaiusT to identify types of syntactic indicators for the concepts of the annotation schema (see Sections 7.1.1 and 7.2.1). Such components perform statistical analysis of the documents and uses lexical database sources of the specific language to integrate results.

The most problematic issue for annotation across languages is the semantic level, because it has to account for alternative world-views and cultural backgrounds. The existence of semantic vagueness, semantic mismatch and ambiguities between languages poses several challenges [58]. The main problems related to this level are caused by proper names, temporal and spatial conventions, language patterns, such as specific or idiomatic ways to express certain concepts, representation of co-reference, time reference, place reference, causal relationships, and associative relationships. For example, consider the concept “week”: in many European countries, a week starts on Sunday and ends on Saturday, whereas in some Eastern-European countries it starts on Monday and ends on Sunday. Thus, the expression “by the end of the week” refers to different time frames depending on the country. The best approach to handle semantic differences consists in developing parallel alignments with the same methodology and the same conceptual model [50]. When using GaiusT with English and Italian documents, such semantic differences were not essential and did not have to be captured in the conceptual model. This section has considered some of the issues that arise in annotating multilingual legal documents. GaiusT components were adopted to deal with these issues. However, for other aspect such as contextual information and implicit assumptions (i.e. ambiguities used to embrace multiple situations), a deeper semantic analysis is required (see, for example, [17]).

5 System Architecture

This section presents the system architecture of GaiusT tool, including its components and technologies used. The overall architecture is shown in Figure 2.

GaiusT integrates and extends existing modules of Cerno (represented in Figure 2 using green color) by adding modules to automate text preprocessing, annotation and annotation schema generation. The extraction process is supported by several components that start from input documents, annotate and map the results into a database for late analysis. The process of annotation used by Cerno is based on a hand-crafted conceptual model, specifying domain-dependent information enriched with lexical information to support identification of semantic categories. The enriched model is the basis for defining an annotation schema. Taking into account that the annotation schema can change according to user interests and goals, and that the schema has a great impact on the annotation process, we developed a series of modules that support the generation of an annotation schema.

GaiusT has been developed on a Microsoft Windows platform using the .NET framework (language C#). Choices about external tools and libraries such as PDF text extractor library, or part-of-speech tagger have been mainly driven by platform compatibility, modularity, integration and multilingual requirements.

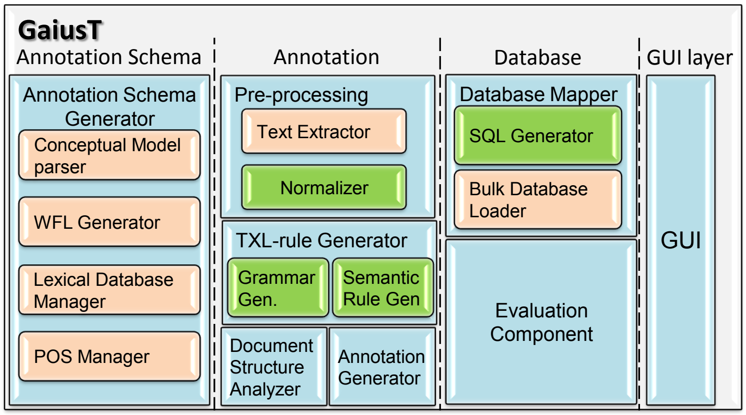


Fig. 2 Architecture of GaiusT

The architecture of GaiusT is composed of four main layers: (a) the annotation schema layer, (b) the annotation layer, (c) the database layer and (d) the graphical user interface layer (GUI), as shown in Figure 2. The entire system is about 50k lines of code and more than 130MBytes of size. Physically, the system includes eight main components: (1) annotation schema generator, (2) pre-processing component, (3) TXL-rule generator, (4) document structure analyzer, (5) annotation generator, (6) database mapper, (7) evaluation component, and (8) GUI.

The components of the architecture are presented in following subsections, taking into account input and output data, as well the functionality of each component.

5.1 The Annotation Schema Layer

The Annotation Schema Layer sets up the core elements for the semantic annotation (SA) process and consists of the ANNOTATION SCHEMA GENERATOR COMPONENT. Its main input is a conceptual model that represents the domain of the document, defined taking into account also of the goals of the annotation. Its main output is the Preliminary Annotation Schema (PAS). The component is implemented using a combination of C# programs and TXL programs. The generator uses four modules to generate the preliminary annotation schema:

- *The Conceptual Model Parser* extracts from an XML Metadata Interchange (XMI)[4] or RDF[5] or OWL[6] source file concepts, properties, relationships and constraints, and produces a list of items organized in a plain text file. This module takes as input a conceptual model generated using a CASE tool that supports the XMI UML interchange language such as UMLSpeed, MagicDraw[7], Visual Paradigm[8] or Protegé[9] – and returns the skeleton of a PAS. The module is a C# library.

[4] XMI is the standard language used for representing Unified Modelling Language (UML) models (http://www.omg.org/technology/documents/formal/xmi.htm)

[5] http://www.w3.org/RDF

[6] http://www.w3.org/OWL

[7] http://www.magicdraw.com

[8] http://www.visual-paradigm.com

[9] http://protege.stanford.edu/

- *The Word Frequency List (WFL) Generator* provides an inverse frequency list of the words from one or more input documents. First the module tokenizes the input and removes, according to the language, stop words. The *WFL Generator* produces two separated lists: one with the frequency of words of the source file and another list with the lemma of the words and their frequency. The process of lemmatization, that reduces inflectional forms of a word to the lemma, is realized by the lemmatizer module of Lucene open source search engine [10] that also supports multilingualism.
- *The Lexical Database Manager* is an interface for lexical resources like WordNet[11], Thesaurus[12] and for the multilingual definitions, Wikipedia[13]. The input to the module are elements extracted from conceptual model, while the output is a list of syntactic candidate indicators for each element of the PAS. The interface provides facilities to query lexical resources both on-line and standalone and filter the output as for example all the options provided by the WordNet command line program to filter the output.
- *The Part-of-Speech (POS) Manager* is a module that can be used to refine the syntactic indicators of the PAS. It uses a Part-of-Speech (POS) tagger program developed by the Institute for Computational Linguistics of the University of Stuttgart[14] [51], to generate a table with the token elements of the document and the probability of their syntactic roles. The tokens can be used to strengthen or enrich the list of linguistic indicators. The POS Manager supports different languages.

5.2 The Annotation Layer

The Annotation Layer includes four components: the PRE-PROCESSING COMPONENT, the TXL-RULE GENERATOR, the DOCUMENT STRUCTURE ANALYZER and the ANNOTATION GENERATOR. It is the core layer of the tool and it pre-processes input documents, delimits the text unit boundaries, generates the grammar and rules for the parser and performs the annotation.

- THE PRE-PROCESSING COMPONENT extracts plain text from input documents and cleans up unprintable characters. *Text Extractor* uses open source libraries[15] to extract plain text from Microsoft Word document (doc), Rich Text Format (rtf) and Adobe Acrobat documents (pdf), while for Web pages of different formats (HTML, XHTML, PHP, ASP, and XML structured documents) it uses a TXL program that extracts plain text and whenever possible, preserves structural information useful for document structure identification. The input of the component is a source file in one of the above formats and the output is plain text. *The Normalizer*, based on an existing TXL module of Cerno, normalizes the plain text document generated by the *Text Extractor* [28], by removing leading characters, unprintable characters and trailing spaces, and producing as output a text document where each line represents a phrase. The module supports the different character sets of European languages.
- THE TXL GENERATOR COMPONENT consists of two distinct domain- independent modules that generate TXL grammar rules and patterns. *The Grammar Generator* is a TXL

[10] http://lucene.apache.org/java/docs/

[11] http://wordnet.princeton.edu/

[12] http://thesaurus.reference.com/

[13] http://wikipedia.org/

[14] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[15] http://www.codeproject.com/KB/cs/PDFToText.aspx and http://www.codeproject.com/KB/cs/DocToText.aspx.

program that given a PAS generates a TXL grammar file used at the annotation phase. This module generates expansion of the lemma of syntactic indicators defined in the Preliminary Annotation Schema. It support English and Italian language, by expanding regular verbs and nouns with proper forms (such as 's' for nouns plural and third person declension of English verb or declension of present and past regular Italian verbs with 'are','ere','ire' suffixes). An example of such a grammar file is represented in Figure 3. *The Semantic Rule Generator* generates a TXL program file with as many rules as the number of patterns and constraints defined in the PAS.

- THE DOCUMENT STRUCTURE ANALYZER annotates text units of a given document with XML tags delimiting its text unit boundaries. The Document Structure Analyzer is composed of a TXL program with a set of rules, grammars and patterns, used to identify text units. The rules and grammars are ambiguous and utilize island parsing techniques to identify relevant parts [15]. Grammar overrides are used to express robust parsing, and to avoid errors or exceptions due to input not explained by the grammar. The input to the component consists of the grammar for the document structure and the normalized plain text document. The output is a document annotated with XML tags that identify structural elements.
- THE ANNOTATION GENERATOR is the core component of GaiusT. It embeds an extended version of the existing Cerno annotation prototype developed in [28]. The component runs the TXL interpreter with the program for annotation and the grammar files created by the *Grammar Rule* and the *Semantic Rule Generator* components. The input of the component is a semi-structured text, i.e., the result of pre-processing and structural analysis, while the output is the text annotated in XML format with the concepts of the conceptual model used for the annotation. The XML output document can be converted to OWL format by using JXML2OWL[16].

```
define Policy
  [act_]
  | 'group [space] 'health [space] [plan_]
  | 'implementation [space] [specification_]
  | [standard_]
  | 'trading [space] 'partner [space] [agreement_]
  | 'covered [space] 'functions
  | 'organized [space] 'health [space] 'care [space] [arrangement_]
end define
```

Fig. 3 An example of grammar file generated by the Annotation Generator component

5.3 The Database Layer

The Database Layer includes the components that support evaluation and the mapping of the annotation data to a relational database.

- THE DATABASE MAPPER accepts the annotated XML document and uploads it into a database using two modules: *The SQL Generator* and *The Bulk Database Loader*. *The SQL Generator* developed as part of a Master's Thesis at Queen's University [52], uses a TXL program to parse an input annotated document and produces an output file with

[16] http://jxml2owl.projects.semwebcentral.org/

SQL data definition language statements along with SQL data manipulation language statements for an SQL Relational Database. The input to the module is an XML document and the set of tags used for the annotation. Using these generated statements, *The Bulk Database Loader* executes the SQL statements produced by *SQL Generator* module according to the target relational database. The input is SQL statements and the output is a populated relational database in a database system such as MySQL, Microsoft SQLServer or Oracle.

– THE EVALUATION COMPONENT is developed to support the assessment of the quality of annotation results. This component facilitates the evaluation process with the following features: (i) *Text preparation*: the annotated text is organized in a table where columns are the set of tags used for the annotation, and rows are the annotates units. For each tag the number of occurrences in text units is reported. This representation provides input for statistical measures of the annotated text; (ii) *Comparison with the reference annotation*: the results of the GaiusT annotation, according to the evaluation framework provided in [28], can be compared with the manual annotated document. The component automatically compares the documents annotated by the tool with their reference annotation, i.e., manually annotated gold standard if there is one, by calculating Recall, Precision, Fallout, Accuracy, Error and F-measure [62]. The component accepts as input the documents with embedded annotations and the list of tags used for the annotation. The output of the process is provided in a database, where each evaluated document is represented as a table. The *Evaluation component* is realized in C# and its input is one or more documents annotated by GaiusT, producing as outputs tables with the evaluation measures. The output can be exported as a spreadsheet.

5.4 The Graphical User Interface

The last layer is the Graphical User Interface (GUI) that supports the management of the entire process of semantic annotation. In particular the GUI supports the process of annotation orchestrating all components and modules (see Figure 4). It provides the control over all input and output operations and parameters of every GaiusT component. The GUI is realized using Windows Form of the Microsoft .NET platform.

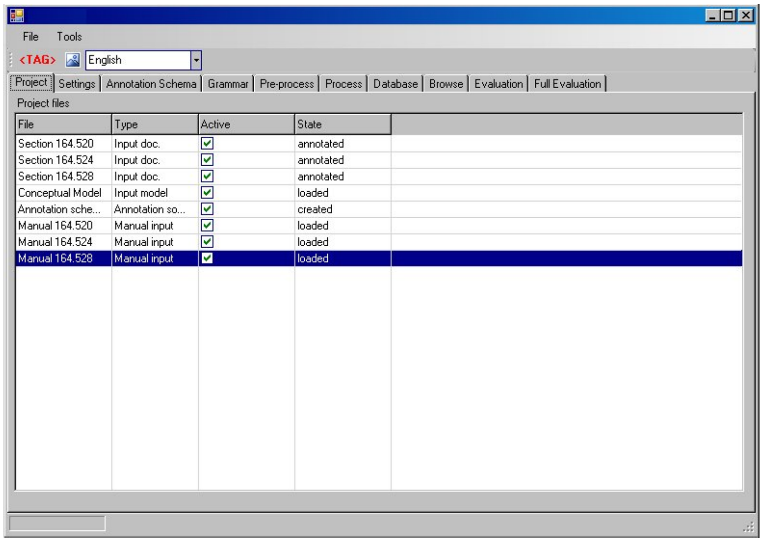
The panels of the GUI are as follows:

– Project: the panel allows the control of the *Annotation Project*. An *Annotation Project* is an XML database file that stores all references to resources used for the annotation, the settings of the system, the input and output files of each component and the parameters used for each component (see Figure 4).
– Settings: the panel allows to control the settings of external resources such as the TXL programs, the database repository for storing annotation, Web resources, third party applications such as WordNet and so on.
– Annotation Schema: this panel allows to control all the components that contribute to the creation of the annotation schema. It is subdivided into different sections related to different modules of the *Annotation Schema* component.
– Grammar: the panel allows managing the collection of grammar files used by the TXL programs. Each item of a grammar file can be managed through a checkbox window that permits to add, edit, or remove grammar items.
– Pre-process: the modules *Text Extractor* and *Normalizer* are controlled through this panel. The panel manages also the *Document Structure Analyzer* component.

Table 4 Components of GaiusT: input and output data type

	Component and modules	Input	Output
1	Annotation Schema Generator		
	1.1 Conceptual Model Support	–	XMI file
	1.2 Conceptual Model Parser	XMI, RDF, or OWL file	preliminary annotation schema (structured text file)
	1.3 WFL Generator	plain text document(s)	database with one table for each file and an inverse word frequent list for each input file and an integrated a summary inverse word frequent list
	1.4 Lexical Database Manager	preliminary annotation schema	table with list of syntactic indicators for the given concepts
	1.5 POS Manager	plain text document	list of the words with the probabilistic syntactic role
2	Pre-processing Component		
	2.2 Text Extractor	doc, rtf, pdf, Web pages (HTML, Php, Asp . . .), XML	plain text file
	2.2 Normalizer	plain extracted text	cleaned plain text
3	TXL Generator Component		
	3.1 Grammar Generator	preliminary annotation schema	grammar file
	3.2 Semantic Rule Generator	preliminary annotation schema	rule file
4	Document Structure Analyzer	plain text	annotated document with structure elements and objects
5	Annotation Generator	annotation schema, grammar file, rule file, TXL program file, plain text document	XML annotated document
6	Database Mapper	XML document	relational database (Oracle, MS SQL Server or MySQL)
	6.1 SQL Generator	XML document	set of SQL statements
	6.2 Bulk Database Loader	set of SQL statement	populated database
7	Evaluation Component	XML document(s), the manually annotated version	database with as many table as many input XML document
8	GUI	–	–

- Process: provides visualization and control of the steps of the SA process so that each operation can be executed and managed independently from the others allowing flexibility and debugging features.
- Database: the database panel provides an interface to manage the *SQL Generator* and the *Database Bulk Loader* modules.
- Browser: this component displays the annotated document, visualizing tags in different colors. The browser allows also other features, like for example editing the annotation and syntax highlighting. It takes as input plain text, rich text (RTF), or Web pages (HTML, XML, Php, Asp, etc.), and displays it, enabling also hyperlink navigation. The visualization of the annotated document with the identified concepts is given through a XSLT program used to transform the XML markup in a HTML document and presenting the results in a table format. Browser can also support the requirements engineer in the process of manual annotation by facilitating tags insertion.
- Quantitative Evaluation: this panel has functionalities to map the annotated text into a table where each column represents the tag used, and each row annotated units. For each tag the partial number of tags per text unit and the overall number of tags in the document are calculated.
- Qualitative evaluation: this panel facilitates comparison of annotated files against reference annotation (if available) and estimation of human disagreement.



Project files

File	Type	Active	State
Section 164.520	Input doc.	☑	annotated
Section 164.524	Input doc.	☑	annotated
Section 164.528	Input doc.	☑	annotated
Conceptual Model	Input model	☑	loaded
Annotation sche...	Annotation so...	☑	created
Manual 164.520	Manual input	☑	loaded
Manual 164.524	Manual input	☑	loaded
Manual 164.528	Manual input	☑	loaded

Fig. 4 The GaiusT GUI

Table 4 lists the components of GaiusT, showing their input and output data types.

6 The GaiusT Semantic Annotation Process

On the basis of extensions of Cerno components described in Section 3, we present the generic process of semantic annotation used by GaiusT to annotate legal documents. Each step of the process can be carried out manually or automatically. Figure 6 shows the entire annotation process of GaiusT including input and output of the components.

6.1 Definition of a conceptual model.

This phase is necessary if there is no pre-existing conceptual model. The process for creating a conceptual model has been studied extensively in the literature and is supported by a variety of tools. The construction of conceptual model of legal concepts has been presented in Section 4.2.

6.2 Text Extraction.

Each input document is converted into plain text and normalized. This is achieved through two steps: (a) Text extraction is performed by the *Text Extractor* module. This phase is critical, especially for PDF file format, given that there are no format tags as for RTF or HTML documents. In PDF documents, the information related to the format of the documents are completely lost or confused and a manual intervention on text extracted by module is required to normalize document structure; (b) The Normalization is achieved by the *Normalizer* module. The output is plain text where each line represents a sentence.

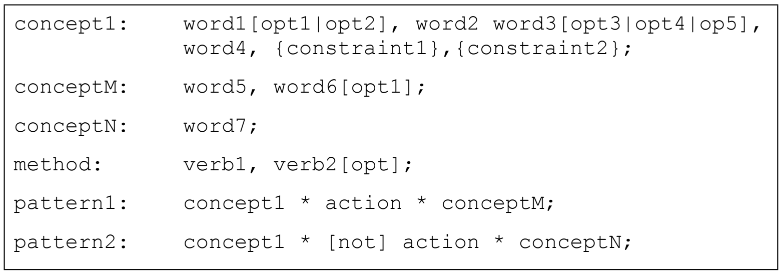
6.3 Construction of the annotation schema.

This step creates the annotation schema elements for the *Annotation* component. The process is divided into three sub-phases:

(a) Construction of the Preliminary Annotation Schema, which includes a set of concepts and other elements extracted from the conceptual model created using XMI, RDF or OWL languages. This preliminary schema is automatically created by using the *Conceptual Model Parser* module. The PAS is structured as a plain text where the concepts, and the other elements such as relationships, properties and constraints, are listed. The relationships are extracted from XMI language and for each relationship a pattern is generated;

(b) Population of the Annotation Schema: The PAS is now revised by adding/filtering out relevant indicators to generate the final version of the *annotation schema*. In particular, each concept and property in the PAS needs to be enriched by syntactic indicators such as keywords related to the concept, i.e., synonyms or hyponyms (Figure 5). The syntactic indicators are generated by using: (1) the *WFL module* to produce an inverse frequency list of keywords[17] of the input document; and (2) the *Lexical Database module* to produce the list of hyponyms and synonyms for each concept and property. The two lists are then merged into one list, by removing the redundant elements. The final output is a table with two columns: the list of concepts and the related keyword retrieved.

(c) Generation of TXL grammar and rules for the Annotation Schema: during this step, the Annotation Schema is parsed by the *Grammar Generator* and the *Semantic Rule Generator*, and two files are produced: a grammar in which the syntactic indicators for each concepts are defined and a program file with all the semantic rules.



```
concept1:     word1[opt1|opt2], word2 word3[opt3|opt4|op5],
              word4, {constraint1},{constraint2};
conceptM:     word5, word6[opt1];
conceptN:     word7;
method:       verb1, verb2[opt];
pattern1:     concept1 * action * conceptM;
pattern2:     concept1 * [not] action * conceptN;
```

Fig. 5 Structure of Annotation Schema

6.4 Annotation

Once the annotation schema is generated, the annotation process can be executed. It is divided into two sub-phases: (a) **The Document Structure Analysis**. In this step the input document is annotated with tags that identify the structure of the document and tags for basic word-equivalent objects. This step is important because it establishes the granularity of the annotation. The granularity can be controlled by the requirements engineer through

[17] The stem of each keyword is considered.

the visual aid provided by the Grammar panel. The annotation is generated through the *Document Structure Analyzer* component, which runs a TXL program with the grammar's items chosen in the Grammar panel. (b) **The Annotation phase.** The *Annotation Generator* annotates the relevant items, according to the annotation schema, generated grammars and rules. Finally, it removes the structural tags used for the boundaries identification of text units.

6.5 Database population (optional).

The last stage of annotation, is the population of a database with annotated documents. There are two modules that perform this operation: the *SQL Generator module* is used to generate a relational database from the XML annotated documents, while the *Bulk Database Loader module* loads the data into the target database. The target database can be any of the relational databases that support the SQL92 standard[18].

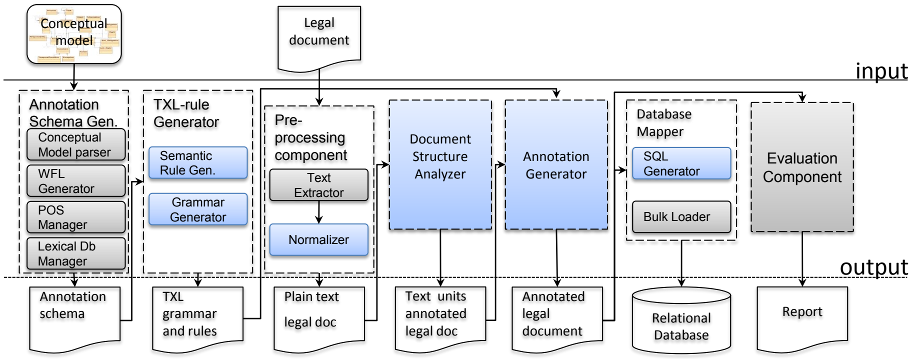


Fig. 6 Annotation Process of GaiusT

7 Evaluation

To evaluate the efficacy and efficiency of the GaiusT tool, two experiments have been conducted, one on the HIPAA Privacy Act (US, in English), and the other on the Stanca law (Italy, in both English and Italian). Our experiments with the tool suggest that it is scalable in the sense that it requires milliseconds to process a HIPAA section, and just a few seconds to process and annotate all of HIPAA (approx. 147K words). In all experiments we did, evaluation was accomplished by comparison against human performance consistently with evaluation framework of [31]. Moreover, the comparison does not consider overhead to set up GaiusT resources, such as the construction of the conceptual model and time used for tuning the syntactic elements of the annotation schema. This is an one-time only overhead for a given language and can be at least partially offset by overhead to train people so that they are comfortable with legal concepts and the annotation process for legal documents. We

[18] `http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt`

Table 5 The document structure of American Law

Level	Unit	Example
1	Volume	VOLUME I
2	Title	Title XVIII Health Insurance for the Aged ...
3	Chapter	CHAPTER III–SOCIAL SECURITY ...
4	Subchapter	Subchapter B. Transfer of Credit
5	Part	PART 160 – GENERAL ADMINISTRATIVE ...
6	Subpart	Subpart A – General Provisions
7	Section	§160.101 Statutory basis and purpose
8	Subsection	(a) Except as otherwise provided, the standards, ...
9	Subdivision	(2) A health care clearinghouse...
10	Subsubdivision	(i) On behalf of such covered entity ...
11	Subsubsubdivision	(A) A function or activity involving the ...
12	Sentence	Hybrid entity means a single legal entity that ...

have not attempted a comparison of GaiusT with other tools, for pragmatic reasons. None of the tools we reviewed has comparable features or scope with GaiusT.

7.1 The HIPAA Privacy Rule

7.1.1 Case study setup

The U.S. Health Insurance Portability and Accountability Act (HIPAA)[19] is a part of the Social Security Act. The HIPAA Privacy Rule defines standards for the protection of personal medical records and other personal health information. To perform the structure analysis of American laws, the *Document Drafting Handbook* was consulted. It contains the style Guide to Legal writing for Regulations[20] [55]. Patterns extracted for the law hierarchy are reported in Table 5. The structure of American Law contains 12 levels and so it has been necessary to extend the hierarchical grammar of Table 1 for the analysis of the specific text units. The annotation schema used in this application has been focused on extracting a set of objects of concern: *right, anti-right, obligation, anti-obligation, exception* [13], and some types of *conditions of application*. The list of syntactic indicators for HIPAA was constructed following the guidelines of the Semantic Parametrization methodology, and following the process described in in Section 6, a fragment is shown in Table 6. We have used *Annotation Schema* component to parse the input conceptual model and build the Preliminary Annotation Schema populating the list of syntactic indicators with *WFL Generator*, *Lexical Database Manager* and *POS Manager*.

Some of the indicators are complex patterns that combine literal phrases and basic concepts. Thus, the identification of indicators requires a preliminary recognition of cross-references, policies and actors. As regards *cross-references*, internal ones can be consistently identified by GaiusT using the small set of patterns shown in Figure 7. External ones were not included in the annotation schema, as we have no hints regarding what kind of format they can be in.

[19] U.S. Public Law 104–191, 110 Stat. (1996)

[20] http://www.whitehouse.gov/omb/memoranda/fy2007/m07-07.pdf; http://www.archives.gov/federal-register/write/handbook/ddh.pdf; http://www.archives.gov/federal-register/write/legal-docs/

Table 6 Examples of syntactic indicators for legal requirements in HIPAA.

Concept type	**indicators**
Right	<policy> . . . </policy> permits
Obligation	<actor>which is charged with <policy> . . . </policy> requires
Anti-Obligation	<actor> . . . </actor> is not required <policy> . . . </policy> does not restrict, does not require
Temporal Constraint	for the time, during, no later, within the time
Cross-Reference Constraint	set by <cross-reference>, required by <cross-reference> as (otherwise) provided in <cross-reference>, pursuant to <cross-reference> in <cross-reference>, under <cross-reference> . . .

The introduction section 160.103: "Definitions of HIPAA" has been used where terms are strictly defined and reference synonyms are assigned; a fragment is shown in Figure 8. The reference word is used consistently over the entire document denoting all related terms. For instance, the HIPAA text operates with such terms as "policy", "business associate", "act", "covered entity", and others. In the analyzed sections of HIPAA, other terms that could be generalized into a common, abstract type were found, including *event*, *date* and *information*. Thus, on the basis of the definition section, a list of syntactic indicators for all the basic concepts has been derived. The final list of basic concepts includes *actor*, *policy*, *event*, *date*, and *information*. GaiusT has been applied to the parts numbered 160 "General Administrative Requirements" and 164 "Security and Privacy" of the HIPAA Privacy Rule containing a total of 33,788 words [56]. Automatic annotation of this text by GaiusT base tool required only 3.07 seconds on an Intel Pentium 4 personal computer with a 2.60 GHz processor and RAM 512 MB of memory running Windows XP operating system. As a result, about 1,900 basic entities and 140 rights and obligations were identified. The manual evaluation of the quality of automatic results was carried out for the following sections contained in the analyzed parts: §164.520: Notice of privacy practices for protected health

```
define citation
  '§ [opt space][number][repeat enumeration] | 'paragraph [space]
  [repeat enumeration] | 'paragraph [opt space][decimalnumber]
  [repeat enumeration] |[decimalnumber][repeat enumeration]
end define
define enumeration
  '( [id] ') | '( [number] ')
end define
```

Fig. 7 The grammar for cross-reference objects.

Actor: ANSI, business associate(s), covered entit(y|ies), HCFA, HHS, . . . ;
Policy: health information, designated record set(s), individually identifiable
health information, protected health information, . . . ;

Fig. 8 Some of the indicators for basic entities according to the information in the definitions section

information; §164.522: Rights to request privacy protection for protected health information; §164.524: Access of individuals to protected health information; and §164.526: Amendment of protected health information. These sections were chosen because the results acquired by GaiusT can be compared to the results of the manual analysis reported in [13]. The manual analysis by an expert requirements engineer of the chosen fragments, containing a total of 5,978 words or 17.8% of HIPAA, took an average of 2.5 hours per section. Figure 9 illustrates an excerpt of annotated text from §164.526(a)(2) in the HIPAA resulting from GaiusT's application. Each embedded XML annotation is a candidate "object of concern." For example, in the figure 9, the "Index" annotation denotes the sub-paragraph index "(i)" and the Actor annotation denotes the "covered entity"; the latter appears twice in this excerpt.

<Right>A<Actor> covered entity</Actor> may deny an <Actor> individual</Actor> 's request for amendment,</Right> if it determines that the <Information> protected health information</Information> or record that is the subject of the request:
<Index> (i)</Index> Was not created by the <Actor> covered entity</Actor> ,
<Exception> unless the <Actor> individual</Actor> provides a reasonable basis to believe that the originator of <Information> protected health information</Information> is no longer available to <Policy> act</Policy> on the requested amendment </Exception> …

Fig. 9 A fragment of the result generated by GaiusT for HIPAA Sec.164.526

7.1.2 Analysis of results.

According to the evaluation framework presented in [31] in a first stage the tool and four human expert annotators marked up sections Sec. 164.524 and Sec. 164.526. The annotated text of this fragment of HIPAA consisted of 77 sentences. Our human annotators were experienced in both computer science and law. The annotations performed by experts were evaluated pairwise: each annotation was matched against all others and the results of comparisons were aligned adopting the following criteria: (1) first, annotated concepts were considered correct when they overlapped on most meaningful words, so that minor divergences in text fragment boundaries were ignored; (2) then, concepts identified by GaiusT and missed by a human annotator were manually re-visited and validated. We used these annotations to calibrate system annotations as we consider them as an upper bound on what automatic annotation can do. Experts reported an overall performance of 91.6% for recall, and a precision of 82.3%. Adopting expert annotations as gold standard [6,25] and comparing with tool annotation, the tool shows good performances with 84.8% for the F-measure and good accuracy of 77.9% [21].

[21]

- Recall is a measure of how well the tool performs in finding relevant items $\frac{TP}{TP+FN}$;
- Precision is a measure of how well the tool performs in not returning irrelevant items $\frac{TP}{TP+FP}$;
- Fallout is a measure of how quickly precision drops as recall is increased $\frac{FP}{FP+TN}$;
- Accuracy is a measure of how well the tool identifies relevant items and rejects irrelevant ones $\frac{TP+TN}{N}$;
- Error is a measure of how much the tool is prone to accept irrelevant items and reject relevant ones $\frac{FP+FN}{N}$;
- F-measure is a harmonic mean of recall and precision $\frac{2\times Recall\times Precision}{Recall+Precision}$

Table 7 Evaluation rates of Experts and GaiusT annotating HIPAA.

Measure	Overall experts performances	GaiusT calibrated with gold standard
Precision	0.83	0.84
Recall	0.92	0.87
Fallout	0.49	0.42
Accuracy	0.78	0.78
Error	0.22	0.22
F-Measure	0.86	0.85

As for the annotation of basic concepts, calculation of evaluation metrics was not possible, as the human annotator was asked to identify only high-level deontic concepts. By manually revising the annotated results, we can say the tool correctly identified all instances of the concepts actor, policy, event, information and date with a precision of 91%. GaiusT also correctly recognized section and subsection boundaries, titles and annotated paragraph indices with a precision of 96%. These structural annotations were used to disambiguate and manage cross-references.

On the basis of the experimental study, we can conclude that GaiusT was able to identify legal requirements with high precision (from 93 to 100%). Good recall rates were also demonstrated for most concepts (70 to 100%), apart from anti-obligation (33%), where the tool's performance must be improved.

Overall, the GaiusT tool largely reduces human effort in identifying legal requirements by doing some of the work, also by giving hints to human annotators on how to proceed. For example, the tool could identify constraints and subject candidates which the human annotator could then link to identified concept instances.

7.1.3 Productivity evaluation

The goal of this application was to test the usefulness of the tool for non-experts in regulatory text who may have to analyze such documents to generate requirements specifications for a software system. In addition to the expert evaluation, an empirical validation of the proposed tool against inexperienced requirements engineers has been carried out. The problem here is that requirements engineers are not always supported by lawyers when designing new software. For this purpose, sections Sec. 164.524 and Sec. 164.526 of HIPAA was selected for annotation by subjects, who are not working with rules and regulations directly. These parts were selected so that they have an approximately equal number of statements, comprised of 1,205 words and 1,057 words respectively.

The experiment involved eighteen master-level Computer Science students taking a software engineering course. Participants in this experiment were not familiar with legal terms and need some training in semantically annotating legal text. A detailed explanation of the annotation process and examples of the concepts to be identified were available. Moreover, the participants were provided with an user-friendly interface to facilitate insertion and modification of tags in input documents.

The setup of the experiment consisted in dividing participants into two groups, and each group was asked to annotate Sec. 164.524 and Sec. 164.526 sections (containing a total of 2,269 words), in two different time frames. In the first time frame group one worked on Sec. 164.524 section without annotations, while group two on section Sec. 164.526 with

Where TP is the number of items correctly assigned to the category; FP is the number of items incorrectly assigned to the category; FN is the number of items incorrectly rejected from the category; TN is the number of items correctly rejected from the category; and N is the total number of items N = TP + FP +FN +TN. .

Table 8 Average Time spent to annotate HIPAA sections by novice annotators.

Section	Time spent to fully annotate Section	Time spent to fix GaiusT annotations
Section 164.524	00:59:40	0:37:14
Section 164.524	01:00:14	00:42:29
Overall	00:59:57	00:39:52

Table 9 Comparing Novice results vs. Novice assisted by the tool.

Measure	Novice full annotation	Novice improve GaiusT annotation
Precision	0.68	0.91
Recall	0.52	0.96
Fallout	0.51	0.25
Accuracy	0.50	0.90
Error	0.50	0.10
F-Measure	0.55	0.93

annotations previously generated by GaiusT. In the second time frame the sections were switched.

The annotators were asked to incrementally identify rule statements and their components in each of the two texts, inserting markups on the original page for the unannotated text, and modifying GaiusT's output in the annotated text. Our statistics note the time spent for annotation of both texts by each novice annotator, also the number of different entities identified during the annotation process. Table 8 summarizes average times spent by participants to fully annotate the two sections and average times spent to fix already annotated sections by GaiusT. Average time saved by using the output of the tool was 33.52%, a notable saving considering that the sections are only a small fragment of the entire law. The entire HIPAA is 147K words and assuming that annotation time depends linearly on the size of the law, it would require almost three days for complete annotation.

Concerning the performance of novices relative to the gold standard, novices who performed a full annotation without assistance achieved a recall of 52% and an accuracy of 50%. By comparison, novices who simply improved the automated GaiusT annotations reported a much higher rate of precision and recall (95% and 90% respectively) with an accuracy of 90% (see Table 9).

This confirms that the tool is a great help for novices both in terms of productivity, as well as quality of the results. Overall, the GaiusT tool largely reduces human effort in identifying legal requirements by doing some of the work, also by giving hints to human annotators on how to proceed. For example, the tool could identify constraints and subject candidates which the human annotator could then link to identified concept instances. Annotators generally expressed satisfaction with the GaiusT tool and the annotations it provides, finding them easy to read and helpful in interpreting a legal document. However, they also pointed to some areas where the tool can be improved[22].

7.2 The Italian Accessibility Law

7.2.1 Case study setup.

The *Italian Accessibility Law by Stanca* contains a set of provisions to promote access by disabled people to information technology instruments [26]. This law, defines a set of

[22] The complete results and resources are available at `https://docs.google.com/file/d/0B7VFCr6GDi-sZE10MWJfRTRkYWc/edit?usp=sharing`

Table 10 The document structure of Italian Law

Level	Unit	Usage		
		Italian name	**Pattern**	**Example**
1	Book	Libro	‘Libro [num]	Libro I
2	Part	Parte	‘Parte [num]	Parte I
3	Title	Titolo	‘Titolo [num]	Titolo I
4	Chapter	Capo	‘Capo [num]	Capo II
5	Section	Sezione	‘Sezione [num]	Sezione IV
6	Article	Articolo	‘Art. [num] ([word]*)	Art. 234 (della proprietá)
7	Clause	Comma	[num] ‘. [word]*	2. Il regolamento di cui al comma 1
8	Letter	Lettera	[char] ‘) [word]*	a) i criteri e i principi operativi
9	Number	Numero	[num] ‘) [word]*	3) efficacia nell’uso e rispondenza
10	Text	Testo	[word]*	Garantire che tutti gli elementi informativi . . .

guidelines containing the different accessibility levels and technical requirements and the technical methodologies used to verify the accessibility of Web sites, as well as the assisted evaluation programs that can be used for this purpose. The Stanca Law was chosen because it is available both in English and in Italian[23]. The availability of both language versions allows us to perform a more fine grain analysis of the different style and syntax used in the two documents.

For the identification of hierarchical structure for Italian law, the Italian guidelines for Lawmaker *Guida alla redazione dei testi normativi* have been used [47]. The guidelines provide suggestions about style, structure, and the type of verbs to be used to express legal concepts. The structure of Italian law foresees 10 levels instead of the 12 of American Law, and the naming convention used for the text units are different; thus, the hierarchy grammar that had been developed for HIPAA had to be adapted to correspond. The analysis of the text units that characterize the Italian law is reported in Table 10.

The grammar rules to support the interpretation of the cross-references of an Italian law were derived from the structural elements. The TXL programs have also been extended to deal with the Italian character set, including the accented vowels and the apostrophes used for articles and prepositions.

The annotation schema for the SA of Italian legal documents has been designed keeping in mind the fact that Italian law uses verbs in the present tense to express deontic concepts and the use of modal verbs is left to the Lawmaker [47]. The syntactic indicators used for identifing deontic concepts in American Law are partially valid, but needed to be adapted. Two techniques have been used to address this issue: (1) the translation of modal verbs used for the American Law; (2) the manual analysis of the text for the identification of candidate verbs to express the concepts of right and obligation and their antis. As regards the second technique, the *POS Manager* module has been used to retrieve all the present tense verbs of the Stanca Law, and then the candidate verbs for each concept were manually annotated. For identification of actor instances in the Italian law, we adapted two solutions: (1) some instances were mined manually from the definition section “Definizioni”; (2) in order to catch the actors not mentioned in the definitions, we exploited the results provided by *POS Manager*, i.e., all proper nouns were marked as actors. For resource instances, we followed only the first solution reusing the terms stated in the definition section.

[23] English version reports the following note: The published text was translated in English by the Information Systems Accessibility Office at CNIPA – National Organism for ICT in Public Administration – with the sole aim of facilitating a better comprehension of it. The translation does not have official status, therefore the only official text is the one published in the Official Gazette of the Italian Republic, in Italian.

<table>
<tr><td><b>Action</b></td><td>acced[ere], adegu[are], alleg[are], effettu[are], gest[ire], impegn[are],individu[are],pot[ere], riconosc[ere] …</td></tr>
<tr><td><b>Resource</b></td><td>compit[o|i], ret[e|i], immagin[e|i], indicator[e|i], indirizz[o|i], informazion[e|i] …</td></tr>
<tr><td><b>Actor</b></td><td>Presidente della Repubblica, Repubblica, Amministrazion[e|i], Autorità,Cnipa, Comunità, disabil[e|i], Ministr[o|i], Organizzazion[e|i], valutator[e|i] …</td></tr>
<tr><td><b>Obligation</b></td><td>dov[ere], è fatto obbligo, farla osservare, promuov[ere], comport[are], defin[ire], applic[are], costituiscono * preferenza, defin[ire], dov[ere], è adeguatamente motivata, è adottato, è fatto obbligo, è * aggiornato, è subordinata, effettua, farla osservare, favor[ire], garant[ire], indic[are], inser[ire], porre a disposizione, predisporre, preved[ere], promuov[ere], provved[ere], riconosc[ere], si applica, sono adeguati, sono emanati …</td></tr>
<tr><td><b>Anti Obligation</b></td><td>non dov[ere], non sia, non si applica, non si possono stipulare, non esprim[ere]; non [Obbligazione]</td></tr>
<tr><td><b>Right</b></td><td>po[sso|uoi|uò|ssiamo|tete|ssono|ssa], non [Diritto];</td></tr>
<tr><td><b>AntiRight</b></td><td>non po[sso|uoi|uò|ssiamo|tete|ssono|ssa];</td></tr>
</table>

Fig. 10 A fragment of entities used for identifing categories

In order to identify action verbs, we adapted the following heuristic: annotate all verbs in present tense, passive tense and impersonal tense. The verbs in the listed forms also refer to obligations, in accordance with the instructions for writing Italian legal documents. Thus, the corresponding heuristic rule was adapted for identifying obligations. As for rights, obligations and their antis, it is more difficult to identify them in the Italian language. Unlike English, that mainly uses modal verbs to state prescriptions as for instance "the users **must** present their request", Italian regulations use present active ("gli utenti presentano la domanda"), present passive ("la domanda è presentata") and the passive impersonal tense ("la domanda si presenta") of verbs to describe an obligation. The choice of style is highly author-dependent. Each of these styles is equally recommended by guidelines [47]. Therefore, in identifying these concepts, our strategy included (1) translation of indicators identified by Breaux and Antón (see the list of equivalent indicators in Fig. 11), (2) in addition, annotation of those sentences that contain verbs in the tenses that intrinsically express obligations as instances of obligation. Fig. 10 shows the list of syntactic indicators identified for the deontic concepts and a subset of the grammar for syntactic indicators integrated as a domain-dependent component of Cerno which correspond to various concepts. The annotation of the Stanca law, containing a total of 6,185 words on 280 lines, by GaiusT takes only 61 milliseconds on an Intel Pentium 4 personal computer with a 3 GHz processor and 2 Gb of memory running Windows 2003 server. A fragment of the annotated Stanca document is shown in Fig. 11.

Table 11 Syntactic indicators for Italian laws

Concept type	Indicators
Right	tenses of "potere"
Anti-Right	tenses of "non potere"
Obligation	tenses of "dovere", "obbligare"
Anti-Obligation	tenses of "non dovere", "non obbligare"
Exception	"in caso di", "entro"

```
Art. 10 (Regolamento di attuazione)
<Obligation>
1. <Entro novanta giorni dalla data di entrata in vigore della presente
<Policy>legge</Policy></Constraint>, con <Policy>regolamento</Policy> emanato ai sensi
dell'articolo 17, comma 1, della <Policy>legge</Policy> 23 agosto 1988, n. 400, sono
definiti:
a) i criteri e i principi operativi e organizzativi generali per l'accessibilità;
b) i <Resource>contenuti</Resource> di cui all'articolo 6, comma 2;
c) i controlli esercitabili sugli operatori privati che hanno reso nota l'accessibilità dei
propri siti e delle proprie <Resource>applicazioni</Resource> informatiche;
d) i controlli esercitabili sui <Actor>soggetti</Actor> di cui all'articolo 3, comma 1.
2. Il <Policy>regolamento</Policy> di cui al comma 1 è adottato previa consultazione con
le associazioni delle <Actor>persone disabili</Actor> maggiormente rappresentative, con
le associazioni di sviluppatori competenti in materia di accessibilità e di produttori di
<Resource>hardware</Resource> e <Resource>software</Resource> e previa acquisizione del
parere delle competenti Commissioni parlamentari, <Constraint>che <Action>devono</Action>
pronunciarsi entro quarantacinque giorni dalla richiesta</Constraint>, e d'intesa con la
Conferenza unificata di cui all'articolo 8 del <Policy>decreto</Policy> legislativo 28
agosto 1997, n. 281. </Obligation>
```

Fig. 11 A fragment of the annotated Stanca law

7.2.2 Analysis of results.

To evaluate the annotation results an empirical study was designed, involving annotation of the Stanca law, and compared the performance of the tool with manual identification of instances of rights, obligations, and associated constraints. Table 12 presents the results of the evaluation compared to human annotators. In the case of the HIPAA Privacy Act, the comparison was made against human expert annotators. The human annotations were used as the gold standard and the performance of GaiusT was calibrated against their opinions. We have not yet measured the productivity of GaiusT for the Italian language, as the focus of the Italian experiment was to evaluate the quality rather than the productivity of the tool when changing language. The tool demonstrates good precision for all of the concepts, ranging from the lowest 67% for Actor to the highest rate 100% for Right, Anti-Obligation and Constraint. Recall was not as high as precision, showing the lowest rates of 33% for Right and 35% for Constraint. This fact is caused by the lack of syntactic indicators for the reliable identification of these concepts in Italian legal language. Instances of Anti-Right concept were not identified in the document either by the human or tool.

Table 12 Evaluation rates for legal concepts found in the Accessibility Law

Measure	Right	Obligat.	Anti-Right	Anti-Obligat.	Actor	Action	Resource	Con-straint	Policy	Total
Precision	1.00	0.76	-	1.00	0.67	0.88	0.74	1.00	0.87	0.77
Recall	0.33	0.42	-	0.50	0.61	0.64	0.73	0.35	0.94	0.64
F-Measure	0.50	0.54	-	0.67	0.64	0.74	0.73	0.52	0.91	0.70

Our plan for future work includes further investigation of how the identified drawbacks can be resolved. Particular difficulties of Italian text emerged for both human and tool. For instance, in Italian the subject is frequently omitted, as in passive forms of verbs, or hidden by using impersonal expressions, thus making it difficult to correctly classify the whole text regulatory fragment and find the holder of a right or of an obligation. Surprisingly, the official English translation of the Stanca law in most cases explicitly states this information. Consider the use of verb phrases (in bold) to state the obligation in Italian and English versions of the same fragment, below:

Italian statement: *"Nelle procedure svolte dai soggetti di cui all'articolo 3, comma 1, per l'acquisto di beni e per la fornitura di servizi informatici, i requisiti di accessibilità stabiliti con il decreto di cui all'articolo 11* **costituiscono motivo di preferenza** *a parità di ogni altra condizione nella valutazione dell'offerta tecnica, tenuto conto della destinazione del bene o del servizio.*

English translation: *"The subjects mentioned in article 3, when carrying out procedures to buy goods and to deliver services,* **are obliged***, in the event that they are adjudicating bidders which all have submitted similar offers, to give preference to the bidder which offers the best compliance with the accessibility requirements provided for by the decree mentioned in article 11."*

Overall, the annotation results suggest that the GaiusT process for regulation analysis is applicable to documents that are written in different languages, English and Italian. The effort required to adapt the framework for the new application was relatively small. This experiment also revealed several language differences that we were able to quantify. In our future work we plan to conduct a more extensive analysis that may remove other language effects independently from legislator effects. As the fragment shows, in the English version the translator disambiguated implicit information.

In general, the results of the annotation with GaiusT provide a useful input for software engineers looking for requirements contained in regulations, rather than starting from scratch and suggest that the GaiusT-based process for regulation analysis is applicable also to documents written in different languages.

8 Related Work

Automatic analysis of legal documents is an old research field of Artificial Intelligence [1, 7], but due to the complexity of such documents, the goal has still not been achieved [38]. In particular, there is an intense debate on the limits of the logical representation for law in deontic logic [24,45]. The idea of using contextual patterns or keywords to identify relevant information in prescriptive documents is not new. A number of methodologies based on similar techniques have been developed. However, tools to realize and synthesize these methods under a single framework are lacking. In [14], the authors suggest an algorithm for detection and classification of non-functional requirements (NFRs). In a pilot experiment, the indicator terms were mined from catalogs of operationalization methods for security and performance softgoal interdependency graphs. These indicators were then used to identify NFRs in fifteen requirements specifications. The results have shown a satisfactory recall and precision for the security and performance keywords. Antón proposed the Goal-Based Requirements Acquisition Methodology (GBRAM) to manually extract goals from natural language documents [2]. The GBRAM has been applied to financial and healthcare privacy policies [4]. Additional analysis of the extracted goals led to new semantics for modeling goals [9,8], which distinguish rights and obligations, and new heuristics for extracting these artefacts from regulations [13,10]. Manually marked sections of the HIPAA [56] are used to automatically fill in frameworks defined according to their conceptual model of laws [11].

Ghanavati et Al. report a large systematic review of goal oriented approaches to model legal documents by reviewing 88 papers, comparing results and summarizing the contributions and research questions in this area see [21]. Another survey of researches in analysing legal documents for compliance requirements can be found in [41]. Among the other issues, authors underline the need to develop a comprehensive system to support requirements engineers in the requirement compliance task. A set of 9 high level functions have been identified as relevant for such a system: annotation of legal statement is the one addressed by GaiusT [40]. Other requirements are partially fullfielled by GaiusT, among them, the "Semi-automated Navigation and Searching", including cross-references identification (see [27]) and Prioritization of Regulations and Exceptions. To facilitate reasoning with regulations, Antoniou et al. [5] introduce the regulations analysis method based on defeasible logic [39]. For this purpose, the facts manually found in the regulation document are represented as defeasible rules.

Analysis of requirements texts has been automated in several ways. LIDA [42] is an iterative model development method based on linguistic techniques. It uses a part-of-speech tagging to derive instances of the UML abstractions. First, the tool proposes a list of nouns as class candidates to the requirements engineer who should remove irrelevant ones, then the list of adjectives is given for choosing relevant attributes, and finally, the list of verbs is provided for identification of relevant methods and roles. Along similar lines, the NL-OOPS tool was devised to support requirements analysis by generating object oriented models from natural language requirements documents [36]. As the core technology, the tool exploits the LOLITA system for domain-independent natural language processing [19]. In [59], the authors addressed the problem of correct identification of requirements providing the detailed analysis of NASA requirements documents and recommendation on writing clear specifications. As a part of this work, the authors discovered that good requirements specifications use imperative verbs, such as *shall, must, must not*, and others, as in the statements explicitly pointing out to requirements, constraints or capabilities. They also introduced the notion of *continuances*, i.e., phrases that introduce the specification of requirements at a lower level. Such phrases usually appear after the words *below, as follows, listed*, etc. This analysis relates to this application to some extent, a set of heuristic rules to detect requirements were derived by regularities in language of regulations documents. The notion of continuances to identify complete lists was also considered in the analysis. The proposed tool is able to identify such lists correctly and relate them to a specific upper-level concept where they belong. In aspect-oriented requirements engineering, several methods approach the task of identification and separation of concerns using text analysis methods. For instance, the EA-Miner [49] tool supports separation of aspectual[24] and non-aspectual concerns and their relationships by applying natural language processing techniques to requirements documents.

One of the early methods intended to facilitate the transition from informal to formal requirements specifications is the Requirements Apprentice [48]. In order to avoid dealing with the full complexity of natural language, this tool needs a mediator, a skilled requirements engineer who enters the information obtained from the end-user in a command-line style using a high-level language. The system then generates a formal internal representation of a requirement from a set of statements and uses a variety of techniques to identify ambiguities and contradictions. To address the complexity of full natural language in general, and for legal texts too, it is often required to rewrite requirements in structured natural language, reducing the advantages of applying automatic tools. Not rarely, this manual pre-processing comes together with vocabulary restrictions limiting the scope of the system to specific knowledge domain. An example of such approaches is [57]. Many tools have developed to support the visualisation and the storage of manually annotated features. At different level of complexity, such functionalities are offered by tools like Annozilla[25],

[24] Aspects are abstractions used to modularize cross-cutting concerns in software development. Examples include such concerns as security, distribution, functionality, and real-time constraints.

[25] http://annozilla.mozdev.org/index.html

SHOE [26], Yawas [27], and SMORE [28]. However, none of them support automatic annotation, as GaiusT annotation module does; also, most of them are limited in the input documents they can deal with (for example, accepting only HTML documents) or in usability (for example, requiring knowledge of a specific markup language). Beside, GaiusT allows to track the annotation process in many ways, satisfying another requirement for an integrated platform to support all the activities related to compliance with HIPAA and other data privacy and security requirements [34], [12].

9 Conclusions and Future Work

We have proposed the GaiusT tool to facilitate requirements elicitation in the domain of legal documents. The tool supports successive analysis phases, each producing a new level of annotation. The main contribution of this work rests in the integration of a number of techniques from Software Engineering, the Semantic Web and Natural Language Processing to facilitate the elicitation of requirements from legal documents. Our contribution has been evaluated through a series of experiments involving human subjects, with positive results.

There are many technical aspects of the work reported in this paper that need to be improved in future research. In particular the population of the annotation schema with the *Annotation Schema Generator* can be improved by using clustering algorithms [43,53], or Independent Component Analysis [22].

Concerning the semantic annotation of legal documents, the syntactic indicators and patterns used for the identification of types of constraints should be updated with a revision of the annotation schema and indicators, and the set of patterns used for the identification of the subjects of conjunctions or disjunctions – or, and – needs to be completed. This task is problematic even for full-fledged linguistic analysis tools. Experimentation using other kinds of regulation documents is also planned. In particular, contract documents are going to be considered, in order to verify the generality of the proposed method, and improve its effectiveness.

Finally, the multilingual aspect of SA needs to be further investigated, by considering other languages and other types of multilingual documents, such as Web pages and news stories.

References

1. Alchourrón, C., Bulygin, E.: Normative Systems. Springer Verlag, Wien (1971)
2. Antón, A.I.: Goal-based requirements analysis. In: Proceedings of the 2nd International Conference on Requirements Engineering (ICRE'96), IEEE, pp. 136–144. IEEE Computer Society, Washington, DC, USA (1996)
3. Antón, A.I., Earp, J.B., Carter, R.A.: Precluding incongruous behavior by aligning software requirements with security and privacy policies. Information & Software Technology **45**(14), 967–977 (2003)
4. Antón, A.I., Earp, J.B., He, Q., Stufflebeam, W., Bolchini, D., Jensen, C.: Financial privacy policies and the need for standardization. IEEE Security and Privacy **2**(2), 36–45 (2004)
5. Antoniou, G., Billington, D., Maher, M.J.: On the analysis of regulations using defeasible rules. In: Proceedings of the 32nd Annual Hawaii International Conference on System Sciences (HICSS'99), vol. 6, p. 6033. IEEE Computer Society, Washington, DC, USA (1999)

[26] http://www.cs.umd.edu/projects/plus/SHOE

[27] http://www.keeness.net/yawas/index.htm

[28] http://www.mindswap.org/2005/SMORE

6. Artstein, R., Poesio, M.: Inter-coder agreement for computational linguistics. Computational Linguistics **34**(4), 555–596 (2008)
7. Bing, J.: Designing text retrieval systems for conceptual searching. In: Proceedings of the 1st International Conference on Artificial Intelligence and Law (ICAIL'87), pp. 43–51 (1987)
8. Breaux, T.D., Antón, A.I.: Analyzing goal semantics for rights, permissions, and obligations. In: Proceedings of the 13th International Requirements Engineering Conference (IEEE'05), pp. 177–186. IEEE Computer Society, Washington, DC, USA (2005)
9. Breaux, T.D., Antón, A.I.: Deriving semantic models from privacy policies. In: Proceedings of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05), pp. 67–76. IEEE Computer Society, Washington, DC, USA (2005)
10. Breaux, T.D., Antón, A.I.: Mining rule semantics to understand legislative compliance. In: Proceedings of the 2005 ACM Workshop on Privacy in the electronic society (WPES'05), pp. 51–54. ACM Press, New York, NY, USA (2005)
11. Breaux, T.D., Antón, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Transactions on Software Engineering **34**(1), 5–20 (2008)
12. Breaux, T.D., Antón, A.I., Spafford, E.H.: A distributed requirements management framework for legal compliance and accountability. Computers & Security **28**(1-2), 8–17 (2009)
13. Breaux, T.D., Vail, M.W., Antón, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), pp. 46–55. IEEE Computer Society, Washington, DC, USA (2006)
14. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: The detection and classification of non-functional requirements with application to early aspects. In: Proceedings of the 14th International Requirements Engineering Conference (RE'06), pp. 36–45. IEEE Computer Society, Washington, DC, USA (2006)
15. Cordy, J.R.: Generalized selective xml markup of source code using agile parsing. In: Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03), p. 144. IEEE Computer Society, Washington, DC, USA (2003)
16. Cordy, J.R.: The TXL source transformation language. Science of Computer Programming **61**(3), 190–210 (2006)
17. Dini, L., Peters, W., Liebwald, D., Schweighofer, E., Mommers, L., Voermans, W.: Cross-lingual legal information retrieval using a wordnet architecture. In: Proceedings of the 10th international conference on Artificial intelligence and law, ICAIL '05, pp. 163–167. ACM, New York, NY, USA (2005)
18. van Engers, T.M., van Gog, R., Sayah, K.: A case study on automated norm extraction. In: Proceedings of the 17th Annual Conference of Legal Knowledge and Information Systems (Jurix'04), pp. 49–58. Elsevier Science Publishers B. V., Amsterdam, Netherlands (2004)
19. Garigliano, R., Morgan, R., Smith, M.: The LOLITA system as a contents scanning tool. In: Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language Processing (ICAI'93) (1993)
20. Geoffrey, N.: The linguistics of punctuation. Lecture Notes 18. Center for the Study of Language of Information, Stanford, CA (1990)
21. Ghanavati, S., Amyot, D., Peyton, L.: A systematic review of goal-oriented requirements management frameworks for business process compliance. In: Requirements Engineering and Law (RELAW), 2011 Fourth International Workshop on, pp. 25 –34 (2011). DOI 10.1109/RELAW.2011.6050270
22. Grant, S., Skillicorn, D., Cordy, J.R.: Topic detection using independent component analysis. In: Proceedings of the Workshop on Link Analysis, Counterterrorism and Security (LACTS'08), pp. 23–28 (2008)
23. Groza, T., Handschuh, S., Möller, K., Decker, S.: Salt - semantically annotated \$LaTeX\$ for scientific publications. In: Proceedings of the 4th European Conference on The Semantic Web (ESWC'07), *Lecture Notes in Computer Science*, vol. 4519, pp. 518–532. Springer-Verlag, Berlin, Heidelberg (2007). DOI http://dx.doi.org/10.1007/978-3-540-72667-8_37
24. Horty, J.F.: Agency and Deontic Logic. Oxford University Press, New York, NY (2001)
25. Hripcsak, G., Rothschild, A.S.: Technical brief: Agreement, the f-measure, and reliability in information retrieval. JAMIA **12**(3), 296–298 (2005)
26. Italian Parliament: Stanca Act, Law. no. 4, January 9 2004: Provisions to support the access to information technologies for the disabled. Gazzetta Ufficiale 13, Rome (17 January 2004). URL `http://www.pubbliaccesso.gov.it/normative/legge_20040109_n4.htm`
27. Jeremy C. Maxwell, A.I.A., Swire, P.: Discovering conflicting software requirements by analyzing legal cross-references. In: 19th IEEE International Requirements Engineering Conference (RE 2011), Trento (2011). To be published
28. Kiyavitskaya, N.: Tool support for semantic annotation. Ph.D. thesis, University of Trento, Department of Information Engineering and Computer Science (2006)

29. Kiyavitskaya, N., Zannone, N.: Requirements model generation to support requirements elicitation: The secure tropos experience. Automated Software Engg. **15**(2), 149–173 (2008). DOI http://dx.doi.org/10.1007/s10515-008-0028-6
30. Kiyavitskaya, N., Zeni, N., Breaux, T.D., Antón, A.I., Cordy, J.R., Mich, L., Mylopoulos, J.: Automating the extraction of rights and obligations for regulatory compliance. In: Proceedings of the 27th International Conference on Conceptual Modeling (ER'08), *Lecture Notes in Computer Science*, vol. 5231, pp. 154–168. Springer-Verlag, Berlin, Heidelberg (2008). DOI http://dx.doi.org/10.1007/978-3-540-87877-3_13
31. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Cerno: Light-weight tool support for semantic annotation of textual documents. Data & Knowledge Engineering **68**(12), 1470–1492 (2009). DOI 10.1016/j.datak.2009.07.012. URL http://dx.doi.org/10.1016/j.datak.2009.07.012
32. Kiyavitskaya, N., Zeni, N., Mich, L., Cordy, J.R., Mylopoulos, J.: Text mining through semi automatic semantic annotation. In: Proceedings of Practical Aspects of Knowledge Management (PAKM'06), *Lecture Notes in Computer Science*, vol. 4333, pp. 143–154. Springer-Verlag (2006)
33. Kiyavitskaya, N., Zeni, N., Mich, L., Cordy, J.R., Mylopoulos, J.: Annotating accommodation advertisements using cerno. In: ENTER, pp. 389–400 (2007)
34. Lazzarotti, J.: Automating hipaa compliance tracking and audit preparation (2011). URL http://www.workplaceprivacyreport.com/2011/11/articles/hipaa-1/automating-hipaa-compliance-tracking-and-audit-preparation/
35. Mann, W., Matthiessen, C., Thompson, S.: Rhetorical Structure Theory and text analysis. In Mann, C., & Thompson, S. (eds) (1992)
36. Mich, L.: NL-OOPS: from natural language to object oriented requirements using the natural language processing system lolita. Natural Language Engineering **2**(2), 161–187 (1996)
37. Moulin, B., Rousseau, D.: Knowledge acquisition from prescriptive texts. In: Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE '90), pp. 1112–1121. ACM Press, New York, NY, USA (1990)
38. Nakamura, M., Nobuoka, S., Shimazu, A.: Towards translation of legal sentences into logical forms. In: Proceedings of the 2007 Conference on New frontiers in artificial intelligence (JSAI'07), *Lecture Notes in Computer Science*, vol. 4914, pp. 349–362. Springer-Verlag, Berlin, Heidelberg (2008)
39. Nute, D.: Defeasible reasoning. In: Proceedings of the 20th Hawaii International Conference on Systems Science (HICSS'87), pp. 470–477. IEEE Press (1987)
40. Otto, P.N., Antón, A.I.: The role of law in requirements engineering. Tech. Rep. TR-2007-07, North Carolina State University (2007)
41. Otto, P.N., Antón, A.I.A.I.: Managing Legal Texts in Requirements Engineering Design Requirements Engineering: A Ten-Year Perspectives, chap. 374-393, pp. 374–393. Springer Berlin Heidelberg (2009)
42. Overmyer, S.P., Lavoie, B., Rambow, O.: Conceptual modeling through linguistic analysis using LIDA. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), pp. 401–410. IEEE Computer Society, Washington, DC, USA (2001)
43. Periklis, A., Panayiotis, T., Renée, J.M., Kenneth, C.S.: Limbo: Scalable clustering of categorical data. In: Proceedings of the 9th International Conference on Extending Database Technology (EDBT'04), *Lecture Notes in Computer Science*, vol. 2992, pp. 123–146. Springer-Verlag, Berlin, Heidelberg (2004)
44. Pietrosanti, E., Graziadio, B.: Advanced techniques for legal document processing and retrieval. Artificial Intelligenge and Law **7**(4), 341–361 (1999)
45. Pizzo, A.: Pensiero pratico e logica deontica: assenza o presenza di razionalitá (in Italian) (2007). URL http://www.filosofia.it. [Online; accessed 25-Feb-2008]
46. Power, R., Scott, D., Bouayad-Agha, N.: Document structure. Comput. Linguist. **29**(2), 211–260 (2003). DOI http://dx.doi.org/10.1162/089120103322145315
47. Presidenza Consiglio dei Ministri: Guida alla redazione dei testi normativi. Gazzetta Ufficiale (in Italian) **101**(2), 1–80 (2001). URL http://www.guritel.it/free-sum/ARTI/2001/05/03/sommario.html
48. Reubenstein, H.B., Waters, R.C.: The requirements apprentice: Automated assistance for requirements acquisition. IEEE Transactions on Software Engineering **17**(3), 226–240 (1991)
49. Sampaio, A., Chitchyan, R., Rashid, A., Rayson, P.: EA-Miner: a tool for automating aspect-oriented requirements identification. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05), pp. 352–355. ACM Press, New York, NY, USA (2005)
50. Sarcevic, S.: New approach to legal translation. Kluwer Law International (1997)
51. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of International Conference on New Methods in Language Processing (ICNMLP'94). Manchester, UK (1994). URL http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
52. Shruti, A.: Relational views of XML for the semantic web. Master's thesis, School of Computing, Queen's University at Kingston, Canada (2007). http://hdl.handle.net/1974/736

53. Souza, V., Zeni, N., Kiyavitskaya, N., Andritsos, P., Mich, L., Mylopoulos, J.: Automating the generation of semantic annotation tools using a clustering technique. In: Proceedings of the 13th International Conference on Natural Language and Information Systems (NLDB'08), *Lecture Notes in Computer Science*, vol. 5039, pp. 91–96. Springer-Verlag, Berlin, Heidelberg (2008). DOI http://dx.doi.org/10.1007/978-3-540-69858-6_10
54. Taylor, S.L., Dahl, D.A., Lipshutz, M., Weir, C., Norton, L.M., Nilson, R.W., Linebarger, M.C.: Integrating natural language understanding with document structure analysis. Artif. Intell. Rev. **8**(2-3), 255–276 (1994). URL http://dblp.uni-trier.de/db/journals/air/air8.html#TaylorDLWNNL94
55. US Federal Register: Document Drafting Handbook. Federal Agency (1998). URL http://www.nara.gov/fedreg
56. US Goverment: Standards for privacy of individually identifiable health information, 45 CFR part 160, Part 164 subpart E. In Federal Register **68**(34), 83348381 (Feb. 20, 2003)
57. Uusitalo, E., Raatikainen, M., Mannisto, T., Tommila, T.: Structured natural language requirements in nuclear energy domain towards improving regulatory guidelines. In: Requirements Engineering and Law (RELAW), 2011 Fourth International Workshop on, pp. 67 –73 (2011). DOI 10.1109/RELAW.2011.6050274
58. Viegas, E.: Multilingual computational semantic lexicons in action: the WYSINNWYG approach to NLP. In: Proceedings of the 17th International Conference on Computational Linguistics (COLING'98), pp. 1321–1327. Association for Computational Linguistics, Morristown, NJ, USA (1998). DOI http://dx.doi.org/10.3115/980691.980784
59. Wilson, W.M., Rosenberg, L.H., Hyatt, L.E.: Automated analysis of requirement specifications. In: Proceedings of the 19th International Conference on Software Engineering (ICSE'97), pp. 161–171. ACM Press, New York, NY, USA (1997)
60. von Wright, G.H.: Norm and Action: a logical enquiry. Routledge & Kegan Paul (1963)
61. Yacoub, S., Peiro, J.A.: Identification of document structure and table of content in magazine archives. In: Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR'05), pp. 1253–1259. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/ICDAR.2005.133
62. Yang, Y.: An evaluation of statistical approaches to text categorization. Information Retrieval **1**(1-2), 69–90 (1999)
63. Zeni, N., Kiyavitskaya, N., Mich, L., Mylopoulos, J., Cordy, J.R.: A lightweight approach to semantic annotation of research papers. In: Natural Language Processing and Information Systems, pp. 61–72. Springer (2007)