Legal Markup Generation in the Large: An Experience Report

Nicolas Sannier, Morayo Adedjouma,
Mehrdad Sabetzadeh, Lionel C. Briand
SnT Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
{firstname.lastname}@uni.lu

John Dann, Marc Hisette, Pascal Thill
Central Legislation Department
State Ministry of the Government of Luxembourg
43, Boulevard Roosevelt, Luxembourg, Luxembourg
{firstname.lastname}@scl.etat.lu

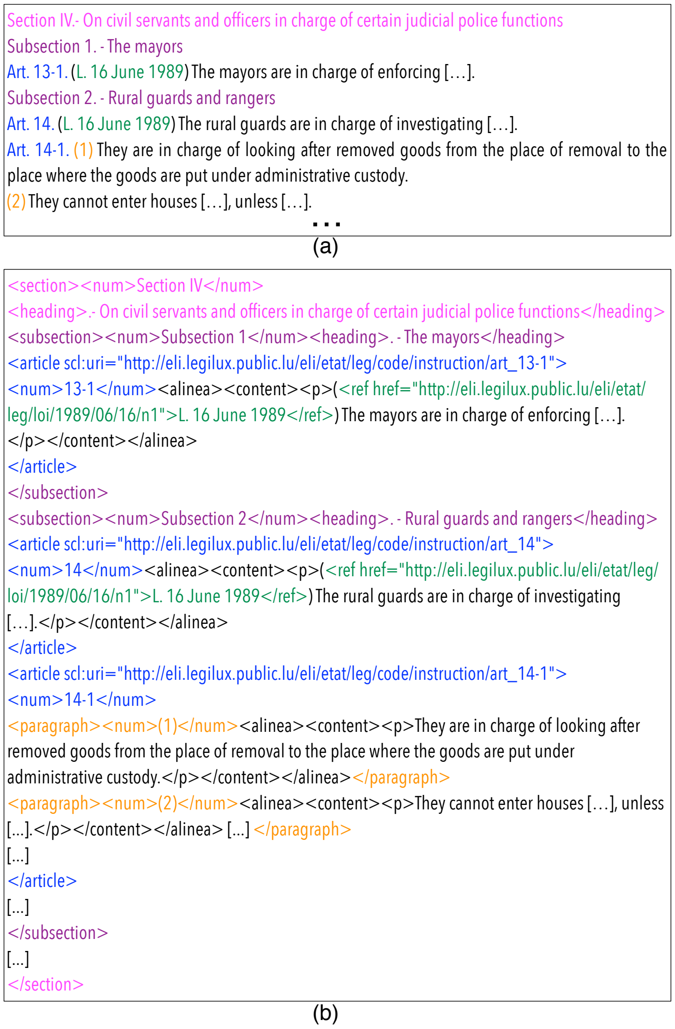
***Abstract*—Legal markup (metadata) is an important prerequisite for the elaboration of legal requirements. Manually encoding legal texts into a markup representation is laborious, specially for large legal corpora amassed over decades and centuries. At the same time, automating the generation of markup in a fully accurate manner presents a challenge due to the flexibility of the natural-language content in legal texts and variations in how these texts are organized. Following an action research method, we successfully collaborated with the Government of Luxembourg in transitioning five major legislative codes from plain-text to a legal markup format. Our work focused on generating markup for the structural elements of the underlying codes. The technical basis for our work is an adaptation and enhancement of an academic markup generation tool developed in our prior research [1]. We reflect on the experience gained from applying automated markup generation at large scales. In particular, we elaborate the decisions we made in order to strike a cost-effective balance between automation and manual work for legal markup generation. We evaluate the quality of automatically-generated structural markup in real-world conditions and subject to the practical considerations of our collaborating partner.**

***Index Terms*—Legal Requirements, Legal Markup, Natural Language Processing (NLP).**

I. Introduction

Legal markup (metadata) is concerned with providing explicit information about the structure and semantics of legal texts. While traditionally studied under the umbrella of legal informatics [2], legal markup has in recent years attracted considerable attention in the Requirements Engineering (RE) community as an enabler for the systematic elaboration of legal requirements and legal compliance analysis. Examples of RE work that relies on or produces legal markup include work by Breaux [3] on legal requirements acquisition, by Massey [4] on legal requirements triage and prioritization, and by Zeni et al. [5] on rights and obligations extraction.

One of the most basic and yet important types of legal markup has to do with the structure of legal texts. The structural markup notably covers the hierarchical organization of these texts as well as the cross references (citations) that link the provisions in these texts. We show in Fig. 1 an example of structural markup over a small snippet of Luxembourg's Code of Criminal Procedure (partial and unofficial translation from French). The markup has been expressed using a combination of Akoma Ntoso [6] –a legal XML schema– and the European Legislation Identifier (ELI) [7] –a specialized URI template for identifying and accessing national and European legislation. We introduce Akoma Ntoso and ELI in Section III.



Section IV.- On civil servants and officers in charge of certain judicial police functions
Subsection 1. - The mayors
Art. 13-1. (L. 16 June 1989) The mayors are in charge of enforcing [...].
Subsection 2. - Rural guards and rangers
Art. 14. (L. 16 June 1989) The rural guards are in charge of investigating [...].
Art. 14-1. (1) They are in charge of looking after removed goods from the place of removal to the place where the goods are put under administrative custody.
(2) They cannot enter houses [...], unless [...].
...

(a)

```
<section><num>Section IV</num>
<heading>.- On civil servants and officers in charge of certain judicial police functions</heading>
<subsection><num>Subsection 1</num><heading>. - The mayors</heading>
<article scl:uri="http://eli.legilux.public.lu/eli/etat/leg/code/instruction/art_13-1">
<num>13-1</num><alinea><content><p>(<ref href="http://eli.legilux.public.lu/eli/etat/
leg/loi/1989/06/16/n1">L. 16 June 1989</ref>) The mayors are in charge of enforcing [...].
</p></content></alinea>
</article>
</subsection>
<subsection><num>Subsection 2</num><heading>. - Rural guards and rangers</heading>
<article scl:uri="http://eli.legilux.public.lu/eli/etat/leg/code/instruction/art_14">
<num>14</num><alinea><content><p>(<ref href="http://eli.legilux.public.lu/eli/etat/leg/
loi/1989/06/16/n1">L. 16 June 1989</ref>) The rural guards are in charge of investigating
[...].</p></content></alinea>
</article>
<article scl:uri="http://eli.legilux.public.lu/eli/etat/leg/code/instruction/art_14-1">
<num>14-1</num>
<paragraph><num>(1)</num><alinea><content><p>They are in charge of looking after
removed goods from the place of removal to the place where the goods are put under
administrative custody.</p></content></alinea></paragraph>
<paragraph><num>(2)</num><alinea><content><p>They cannot enter houses [...], unless
[...].</p></content></alinea> [...] </paragraph>
[...]
</article>
[...]
</subsection>
[...]
</section>
```

(b)

Fig. 1. Snippet of a Legal Text (a) in Plain-text and (b) in XML Format

Several legal portals exist, aimed at offering structural markup for legal texts. Examples include LegiFrance in France (http://legifrance.gouv.fr), e-laws in Ontario (http://ontario.ca/laws), BelgiumLex in Belgium (http://www.belgielex.be/en/), Overheid in the Netherlands (http://overheid.nl/english/), and the Eur-Lex portal for the European Union (http://eur-lex.

europa.eu). An important challenge when creating markup for a corpus of legal texts in a given country or jurisdiction is the very large volume of legacy texts that need to be enhanced with markup. Manually adding the markup is extremely time-consuming. Despite this, our interactions with several governmental authorities indicate that this task is still done largely manually. Consequently, important compromises have had to be made in all the legal portals mentioned above in order to reduce the cost of building markup representations. Notably, the depth of the markup in the existing portals is shallow. For example, the markup would typically go down only to the level of articles without distinguishing the articles' subdivisions, e.g., paragraphs, numbers and sentences. Another major limitation in the portals is that the cross references are not systematically resolved, and hence the dependencies between different provisions not captured in the markup.

Recent initiatives on building visual markup editors for legal texts, e.g., LIME [8] and AT4AM [9], are likely to reduce the manual effort incurred over adding structural markup to new or existing texts. Nevertheless, due to the sheer scale of the existing non-markup legal corpora and also the inherent difficulty of manually manipulating certain markup elements such as cross references, it is essential to provide automated support for the generation of detailed structural markup.

Existing work in the area of legal informatics primarily targets semantic markup, while assuming a-priori presence of basic structural markup. However, the problem of scale and how to add such basic structural markup to large corpora of legal texts amassed over decades and centuries has received little attention. The RE literature acknowledges the importance of automation for creating structural markup. For example, Breaux [3] and Zeni et al. [5] develop automated structural markup generators as part of broader frameworks for metadata extraction. Similarly, our previous work on cross reference analysis [10] incorporates a structural markup generator [1].

Despite the above-mentioned work, and to the best of our knowledge, there are no reports in the literature on structural markup generation *at large scales*. In this paper, following an *action research* method [11], we report on the experience gained from a case study in which we transformed five of Luxembourg's major legislative codes from plain-text into an XML format based on Akoma Ntoso [6] and ELI [7]. Our case study was prompted by an initiative of the Government of Luxembourg to provide a harmonized encoding of the country's legislative texts. Ultimately, the initiative aims to foster open access to legal information, and to offer a more systematic basis for demonstrating legal compliance. Our participation in the initiative came about on the basis of our previous work on automated markup generation [1]. Specifically, our engagement was meant at assisting the Government in improving the efficiency of markup generation. During the case study, we enhanced and adapted our existing markup generation tool [1] to account for the markup requirements of the Government. Simultaneously, we took advantage of the context provided by the case study to evaluate our structural markup generation approach in fully realistic conditions.

The five legislative codes in our case study constitute over 5,000 provisions (articles) and contain a total of over 20,000 structural elements, e.g., articles, paragraphs and cross references, that require markup. The XML documents resulting from the study have since been adopted by the Government and now feature on Luxembourg's official legal portal.[1]

Contributions. The contributions of this paper are two-fold:

(1) Lessons learned: We reflect on the lessons learned from our case study. Considering the tedious nature of manually creating structural markup and also the significant compromises observed in the (manually-created) markup available on existing legal portals, it is logical to surmise that building high-quality legal markup at large scales would be prohibitively expensive without automation. At the same time, such automation cannot be entirely thorough and accurate, mainly due to the flexibility of the natural-language content of legal texts and variations in how these texts are organized. This means that manual work during markup generation remains inevitable. Our lessons learned examine the tradeoffs between manual work and automation for structural markup generation, with an eye towards striking a cost-effective balance between the two. To this end, we describe the challenges that we faced during structural markup automation and the decisions we made to tackle the challenges in the most efficient way we could.

(2) Evaluation of markup quality: We evaluate the quality of the automatically-generated markup. The main characteristic of our evaluation is that it is done in light of the practical considerations made in our study, including the decisions about what to automate and what to do manually (see *(1)* above), the choice of XML markup representation, and the specific details required for the markup in our context. Taking these considerations into account is important for obtaining a conclusive picture of markup quality.

We note that structural markup is a necessary but *not* sufficient step toward legal requirements analysis and compliance. One would further need semantic markup, e.g., for legal conditions and modalities such as permissions and obligations, in order to support legal requirements engineering tasks [5]. We do not address semantic markup in this paper, nor do we elaborate the relationship between legal markup and compliance. This relationship is already well-established in the literature: both structural and semantic markup are considered essential for compliance analysis through such means as Natural Language Processing, Information Retrieval and ontologies [12]. Our goal in this paper is instead to shed light on some key challenges caused by scale, which necessarily need to be dealt with on the path to compliance automation.

Structure. The remainder of the paper is organized as follows. Section II describes the main criteria that one needs to consider when devising a structural markup generation solution. Section III outlines our proposed solution and tool support. Section IV presents our case study. Section V reports on the

[1] All the country's legal codes are at: http://legilux.public.lu/editorial/codes. The codes in our study are at http://legilux.public.lu/editorial/codes/$\langle C \rangle$, where $\langle C \rangle$ is "civil", "commerce", "penal", "instruction_criminelle", or "procedure_civile".

lessons learned, and on the accuracy of our markup generation approach. Section VI discusses threats to validity. Section VII compares with related work. Section VIII concludes the paper.

II. Legal Markup Generation

Given a legal corpus, whether automation is useful for markup generation and what nature this automation should assume are influenced by two key factors. The first factor is the *size* of the corpus, i.e., the cumulative length of the texts in the corpus. The second factor is the *heterogeneity* of the corpus, covering both intra-text and inter-text heterogeneity. Intra-text heterogeneity has to do with individual legal texts not always being entirely consistent in how they are written. Inter-text heterogeneity has to do with different legal texts in the corpus being organized differently from one another.

In Table I, we show, as a function of size and heterogeneity, what we believe to be the most reasonable strategy for markup generation. As shown in the table, we distinguish three different strategies: (1) Manual processing, (2) Automated processing using a generic (one-size-fits-all) approach, and (3) Automated processing using a customizable approach.

TABLE I
Strategies for Markup Generation

		Heterogeneity	
		Low	High
Size	Large	Generic automation	Custom automation
	Small	Manual processing	Manual processing

Manual processing would be reasonable when the corpus, and consequently the number of markup elements to create, is small. *Generic automation* works best when the corpus is homogeneous and adheres uniformly to pre-defined legal writing guidelines. A corpus written over a short span of time and revised only infrequently may indeed be homogeneous. When this is the case, one can develop a cost-effective solution by merely providing hard-coded markup generation rules for the corpus at hand, without much thought given to how these rules can be generalized to other corpora. *Custom automation* adds a tailorability dimension to automation, enabling the definition of meta-rules that can be instantiated differently for different legal texts and even different fractions of the same text. Custom automation acknowledges that adherence to legal writing guidelines is a moving target, since guidelines (and how stringently they are enforced) may evolve over time. This evolution leads to both intra-text heterogeneity, when legal texts are amended frequently, and inter-text heterogeneity when a corpus is expanded with new texts over time.

In most practical settings, e.g., when considering markup generation for an entire legal jurisdiction, we are faced with corpora that are both large and heterogeneous. Developing custom automation therefore presents the most viable choice. At the same time, pursuing this choice would require making several decisions, e.g., about the level of granularity of the markup to generate, and how far to go with automation before the cost of the endeavor would exceed any tangible benefits obtained in return. These are indeed the types of decisions that we elaborate in Section V-A based on our experience.

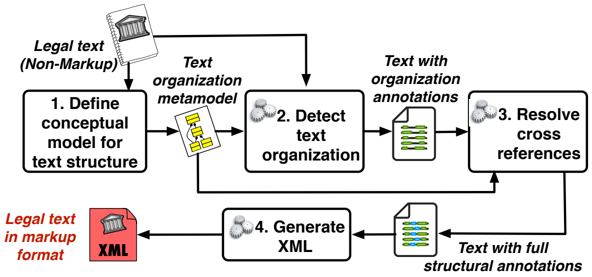

Legal text
(Non-Markup)
Text organization metamodel
Text with organization annotations
1. Define conceptual model for text structure
2. Detect text organization
3. Resolve cross references
Legal text in markup format
XML
4. Generate XML
Text with full structural annotations

Fig. 2. Approach Overview

III. Approach

In this section, we outline our automated structural markup generation approach and its tool support. Further technical details are available in our earlier work [1]. As we already stated in Section I, our focus in this paper is on structural markup exclusively. In the remainder of the paper, we take “markup” to mean “structural markup”.

A. Overview

Our approach, depicted in Fig. 2, has four steps. At a high level, Model-Driven Engineering (MDE) provides the necessary customization facilities for defining the hierarchical organization of legal texts; whereas Natural Language Processing (NLP) provides the ability for automated detection of the text elements and for parsing the cross references.

Step 1 is manual and concerned with defining a conceptual model for a given legal text. In addition to describing the hierarchical organization of the given text in terms of its constituent parts, e.g., chapters, sections and articles, the conceptual model captures (a) the multiplicity constraints that apply to elements and (b) information necessary for element detection. This information notably includes the labels of different element types, e.g., “Art.” for articles, and the specific delimiters (spacing and punctuation) appearing before and after different elements.

In Step 2, we leverage the conceptual model of Step 1 for automated construction and subsequent execution of the NLP scripts that annotate the text under analysis with information about the text’s hierarchical organization. In Step 3, again using the conceptual model of Step 1, we instantiate and execute NLP scripts that detect and resolve the cross references. The scripts in both Steps 2 and 3 are regular expressions for detecting natural-language patterns, enhanced with actions that annotate the text in response to pattern matches. With regard to cross references, the annotations provide information across three main dimensions: (a) explicit versus implicit, denoting whether a cross reference is defined in terms of alphanumeric labels (explicit) or through some adjective, adverb, or anaphor (implicit); (b) complex versus simple, denoting whether a cross reference contains enumerations, ranges and navigation (complex) or not (simple); and, (c) internal versus external, denoting whether the target of a cross reference is inside the text being analyzed (internal) or outside it (external). For (b), our annotations further distinguish cross references that target multiple provisions (multivalued) and those that present information at multiple levels (multilayered). We exemplify

TABLE II
CROSS REFERENCE TYPES AND EXAMPLES

Cross reference dimension				Example
Implicit	Simple		Internal	the next article
			External	N/A
	Complex	Multivalued	Internal	the three previous articles
			External	N/A
		Multilayered	Internal	the first alinea of the same article
			External	N/A
Explicit	Simple		Internal	article 14-1
			External	article 5 of the Law of <date>
	Complex	Multivalued	Internal	articles 5 to 10; articles 3 or 4
			External	articles 5 to 10 of the Law of <date>
		Multilayered	Internal	article 5, paragraph 2a, alinea 3
			External	article 5, alinea 2 of the Law of <date>

cross references in Table II. For further details, see [1]. From Steps 2 and 3, we obtain a legal text with structural annotations. In Step 4, these annotations are transformed in a straightforward manner into the desired markup format.

B. Tool Support

We have implemented the approach in a tool, named Legal Cross reference Analyzer (LeCA). LeCA builds on the Eclipse Modeling Framework and the GATE NLP Workbench [13]. Specifically, Eclipse is used for defining conceptual models for legal texts and for instantiating, based on the defined conceptual models, the NLP scripts that extract the texts' structural metadata. LeCA has been described in our earlier work [1]. As part of our current work, we customize LeCA according to the markup requirements of the Government of Luxembourg. The most important of these requirements are the following: (1) The markup format shall be compliant with Akoma Ntoso; and (2) Element resource names shall follow the European Legislation Identifier (ELI) template.

Below, we briefly present Akoma Ntoso and ELI:

Akoma Ntoso [6] is an XML schema for legal documents. This schema was chosen because of its flexible nature for capturing structural elements. We note that Akoma Ntoso is one among several existing and equally-expressive legal XML schemas. Some alternatives which too have gained traction in practice are Formex [14], MetaLex [15] and LexML [16]. In the example of Fig. 1, the text hierarchy has as root a ⟨section⟩. This section contains a ⟨num⟩ (number), a ⟨heading⟩ (title), and two ⟨subsection⟩s. The subsections each have their own ⟨num⟩ and ⟨heading⟩ and further contain one or more ⟨article⟩s. Each article has a resource name attribute (scl:uri), which is expressed using the ELI naming convention, and a ⟨num⟩. The articles are decomposed into either of the following: (1) ⟨paragraph⟩s with a ⟨num⟩ and containing ⟨alinea⟩s, e.g., as in article 14-1, (2) direct alineas (i.e., without a parent paragraph), e.g., as in articles 13-1 and 14. In our example, alineas contain the actual textual provision, expressed as a ⟨content⟩ and using traditional HTML elements such as ⟨p⟩.

ELI [7] is a European-Union-endorsed initiative aimed at providing a unified legal referencing mechanism. The ultimate goal of ELI is to facilitate access, exchange and reuse of legal knowledge across borders. In particular, ELI defines an intuitive labeling framework based on a generic customizable template to enable the definition of universal resource names independently of countries and legal jurisdictions. ELI is currently used by several European countries including Luxembourg, France, Denmark, Ireland, Italy and also by the EU Publication Office. In the example of Fig. 1, resource identifiers, e.g. the one for article 13-1, and cross references, e.g. "(L. 16 June 1989)", are both specified using ELI. For this example cross reference, the ELI name, eli/etat/leg/loi/1989/06/16/n1, has the following interpretation:

- "eli" is the root of the ELI name;
- "etat" provides information on the source, here a document from the government;
- "leg" describes the nature of the document, here a legislative document;
- "loi" is the type of document, here a law;
- "1989/06/16" is the date when the text was signed; and
- "n1" is a special number to distinguish texts signed on the same day.

The special number above cannot be retrieved without additional information about the law titles and their ELI names. To retrieve the special numbers, we were provided with an index of law titles and their ELI names. We augmented LeCA with a TF-IDF scoring mechanism [17] for identifying, for a given cross reference, the most likely ELI match from this index. In Section V, we examine the accuracy of automated retrieval of these special numbers.

IV. CASE STUDY

Context. The Central Legislation Department (French: Service Central de Législation), abbreviated *SCL*, is responsible for the publication of Luxembourg's legislation and administrative procedures. SCL further maintains the Legilux portal (http://legilux.public.lu), through which it disseminates the national legal texts. Traditionally, the majority of the texts have been provided only in PDF. SCL has already developed semi-automated solutions for the transformation of small legal texts, typically < 10 pages in length, to HTML and XML. However, the existing solutions were not meant to work on complex legal texts, notably the legislative codes, where the organization hierarchy is deep, and where numerous amendments have been made to the texts over decades. The existing solutions further do not automatically handle cross references.

Objective. Our case study attempts, in close collaboration with SCL, to enhance selected laws with markup information. The case study provides a context in which to examine the feasibility and usefulness of our automated legal markup generation approach from the perspective of an organization that has to process very large volumes of legal texts. Specifically, we worked with SCL on transitioning five major legislative codes in Luxembourg from plain-text to an Akoma Ntoso- and ELI-compliant XML format (see Section III-B for brief descriptions of Akoma Ntoso and ELI).

Research Method. Our case study, which was initiated to solve an immediate problem in a practical setting, falls in line with the principles of *action research*. According to

Denscombe [11], the main characteristics of action research are: (1) having a practical nature, (2) inducing change as an integral part of the research, (3) reflection and iterative improvement, and (4) participation from both researchers and practitioners. Our case study possesses all these characteristics.

Case Selection. The selection of legal texts for our case study was based on the priorities of SCL. For the first phase of the transition to markup, SCL chose the following five codes in Luxembourg: the Civil Code, the Commercial Code, the Penal Code, the Code of Criminal Procedure, and the New Code for Civil Procedure. A code consolidates multiple thematically-related legal texts into a single document. Overall, the five codes that we processed contain 5099 articles (see Table III).

Data Collection. Our data collection was targeted at evaluating the effectiveness of our automated markup generation approach and how we could improve it. We processed the selected legislative codes sequentially and in the order listed above. As we analyzed the codes, we kept track of all the issues that we faced during markup generation as well as the decisions we made to address the issues in the way we deemed most cost-effective. For the purposes of this paper, we are obviously not interested in issues that have to do with implementation bugs or refactoring. Our synthesis of the issues (Section V) is thus naturally focused on the inherent difficulties that we observed in our case study context.

Results. In Table III, we summarize the results of our case study. Specifically, the table presents the number of elements of different types that were subject to markup generation as well as the manual effort that was spent by the first two authors over each code.

The elements for which markup was generated are: (1) high-level divisions (part, book, title, chapter, section, subsection), (2) articles, which are the principal content elements of legal texts in most European countries, including Luxembourg, (3) sub-article divisions (paragraphs, alineas, letters (L), numbers (N), and dashes (D)), and (4) cross references. In total, we produced markup for 21111 elements, including 854 high-level divisions, 5099 articles, 11131 sub-article divisions and 4027 cross references. We note that sub-article divisions and cross references collectively represent nearly three quarters (72%) of the elements that are subject to markup. The existing portals that we mentioned in Section I provide no or only sporadic markup for these elements due to the expensive cost of handling these elements manually.

The last column of Table III shows the full-time equivalent (FTE) effort incurred over manual work. This effort is measured in workdays, with each workday being 8 person-hours. The following tasks represent the significant majority of the manual work spent: (1) analyzing the code to understand and specify its conceptual model, (2) preprocessing and cleanup, e.g., removing page numbers, running headers, and erroneous line breaks, (3) fine-tuning the instantiated NLP scripts and the generated Java container classes for the legal text under analysis, (4) manual inspection of the generated XML to verify the correctness of the generated markup, including the markup for cross references, and correcting any issues identified during the inspection, and (5) issue tracking and resolution. Among these, tasks (1) and (3) are specific to our technical approach.

As noted earlier, we analyzed the texts sequentially. When we started working on the first code, i.e., the Civil Code, the researchers, i.e., the first four authors, had no a-priori knowledge of Akoma Ntoso and ELI. In addition to the effort associated with the five tasks above, what we report for the Civil Code includes the time spent on learning Akoma Ntoso and ELI, eliciting SCL's markup requirements, and developing an appropriate XML output generator. This effort is a one-off investment and should ideally be factored out of the effort spent over the Civil Code. However, we are unable to do so in a clear-cut manner: the process we followed was highly exploratory, and the one-off activities were mixed with the tasks performed per legal text. The effort for the Civil Code is therefore not representative.

The learning and reflection process continued well into our analysis of the four remaining codes. Nevertheless, with each new code analyzed, we became more efficient and knowledgeable about how far to go with automation and what issues to anticipate in the next codes. Consequently, as we progressed, our manual activities became more sharply focused on the regular tasks that one has to do for a given code. In this sense, the final three texts in Table III are better indicators for normal productivity when our approach is applied.

We observe from Table III though that the amount of effort is still large, even for the last three texts processed. To this end, we make the following remarks: Between 50% to 60% of the reported effort went into inspecting the legal texts, either during the preprocessing and cleanup of the (plain-text) input, or during the inspection of the generated (markup) output. Even if we had foregone automation completely and not used our approach, this fraction of the effort would still have been incurred. In other words, only 40% to 50% of the reported effort is related directly to markup creation.

The question we need to answer then is how this 40% to 50% of the effort would fare against the situation where one would insert all the markup elements manually into the texts. We cannot answer this question conclusively because we do not have a control case where the markup was created without our tool support. Nevertheless, doing a ballpark analysis over the last three texts in Table III, we can state the following: If we consider the effort spent on markup creation to represent 50% of the reported effort, i.e., $(7 + 10 + 7)/2 = 12$ days, and divide this by the total number of elements processed, i.e., $3357 + 3870 + 5201 = 12428$, we obtain an average of $\approx$28 seconds per element. ***A trained expert even with ideal text editing tools would be unlikely to outperform this pace, given the information requirements mandated by Akoma Ntoso and ELI, and also the inherently-difficult task of cross reference resolution. Based on this analysis, and additional considerations, notably the tedious and repetitive nature of manual markup creation, we believe that automated markup generation for large legal texts is beneficial.***

TABLE III
LEGAL CODES PROCESSED IN OUR CASE STUDY

	High-level divisions						Article	Sub-article divisions			CRs	Total	Effort (FTE workdays)
	Part	Book	Title	Chapter	Section	Subsection		Paragraph	Alinea	L/N/D			
Civil Code	3	3	36	131	154	43	2316	33	3474	361	997	7551	24
Commercial Code	0	4	21	14	12	0	261	14	489	85	232	1132	12
Penal Code	0	2	11	86	46	0	671	64	1094	471	912	3357	7
Code of Criminal Procedure	0	3	15	33	36	7	529	542	1315	325	1065	3870	10
New Code for Civil Procedure	2	11	90	19	42	30	1322	206	2316	342	821	5201	7
Total	5	23	173	283	290	80	5099	859	8688	1584	4027	21111	60

TABLE IV
IMPORTANT FACTORS INFLUENCING THE COST-EFFECTIVENESS OF (STRUCTURAL) MARKUP AUTOMATION

Category	Factor	Decision
Granularity of Automation	Depth of text organization markup	Go as deep as possible, i.e., build finest-grained metamodel possible
	URI strategy	Keep URIs at the level of legal texts and articles
Exceptional Cross References	Cross references outside their container element	Choose between rule-based (automated) processing, or manually moving the cross reference in question into the right container, whichever is most appropriate
	Under-specified external cross references	Manually resolve the cross references in question (expert judgement required)
	Cross references using rare patterns	Manually resolve the cross references in question
Labeling and Line-break Issues	Unlabeled divisions	Manually add (dummy) labels during preprocessing
	Variations in division labels	Fix during preprocessing if variation is an anomaly; adapt the metamodel / scripts otherwise to account for the variation
	Physical lines not aligned with logical lines	Manually remove unnecessary line breaks during preprocessing

V. DISCUSSION

Below, we first present the lessons from our case study, focusing on aspects that we observed to be most pertinent to reducing manual effort in legal markup creation. We then evaluate the quality of the markup generated via automated support, subject to the considerations made in the case study.

A. Lessons Learned from the Case Study

Table IV lists the most important factors that we believe, based on the experience from our case study, one should take into account when devising a (structural) markup generation approach for large and complex legal texts. We classify the factors into three categories: (1) granularity of automation, (2) exceptional cross references, and (3) labeling and line-break issues. Among these, the first category addresses some fundamental considerations regarding the scope of automation. The other two categories address more specific practical issues that arise during text processing. For each factor, we provide in the table the decision that we made. Our decisions were all aimed at minimizing the cost of markup generation, while at the same time, not making major compromises over the quality of the resulting markup. In the remainder of this section, we elaborate on the factors in Table IV and our decisions.

Depth of markup for text organization. As noted in Section I, the shallow depth of the markup available on existing portals presents a limitation. This limitation has two consequences: First, without deep markup, one cannot narrow down analytical tasks, e.g., the extraction of semantic knowledge from provisions, to the finest-grained elements of legal texts. Second, the text organization markup is often used for creating an HTML representation of legal texts over web portals. Naturally, the shallower the markup, the more the manual work that needs to be done to manage the HTML representation layout. These observations were confirmed by SCL, who expressed interest in making the text organization markup as deep as possible.

While the advantages of deep markup are clear, generating such markup automatically is not without costs. Particularly, deep markup necessitates that a precise conceptual model should be available for each legal text. In the (theoretical) worst case, each legal text can have a unique conceptual model of its own. Building a detailed conceptual model from scratch for each text in a very large corpus would be too expensive.

In practice, we have observed, both in our previous work over smaller legal texts [1] and in our current case study, that despite the inherent heterogeneity that exists across different texts and no texts having exactly the same structure, the conceptual models for the texts share many commonalities that can be reused across texts. The differences (variabilities) between the conceptual models are typically in the labels, delimiters, and the containment hierarchy relationships. We observed that, once a conceptual model has been built for a particular category of legal texts, e.g., legislation, in a given country or jurisdiction, the conceptual model is easily adjustable to other texts in the same category. In light of this finding and the advantages of generating deep markup, we recommend that the text conceptual models should be made as precise as possible.

URI strategy. The key question that we need to answer here is: Which division types (see Table III) need to be assigned URIs? In our previous tool [1], where we did not use ELI, we created URIs for *all* division types via unique ids generated by a global counter. For example, the URI for "Section IV" in Fig. 1 would simply be a number, as in `<section uri="504">...</section>`. This scheme is however not compliant with ELI. An ELI-compliant URI for the mentioned section would look like eli/etat/leg/code/instruction/livre_1/titre_1/chap_1/sect_4/, where the section is captured together with all its context (containing parents). The verbosity of this URI stems from the fact that the numbering of both high-level and sub-article divisions (see Table III) is reset when the containing parent changes. For example, the numbering of sections is reset when a new chapter appears. The conclusion reached after consulting SCL was that an across-the-board assignment of URIs to all division types would create a large number of URIs that would be difficult to verify and maintain. This difficulty would further be exacerbated by the fact that one cannot have a harmonized ELI structure for URIs in an entire corpus, because the conceptual models of the texts in the corpus would invariably differ.

Due to the above, the strategy that we chose was to provide URIs only for legal texts and articles. These are the only elements common across all texts. With respect to articles, since their numbering is not reset in a given text, providing the full containment hierarchy in the URIs is unnecessary. For example, for "Art. 14-1" in Fig. 1, the URI can simply be eli/etat/leg/code/instruction/art_14-1 rather than eli/etat/leg/code/instruction/livre_1/titre_1/chap_1/sect_4/art_14-1.

Our URI strategy has implications on how precisely cross references can be resolved. In particular, the strategy would restrict citations to high-level divisions to be linked to the encompassing legal text, and citations to sub-article divisions to be linked to the encompassing article. These implications were indeed desirable for SCL, again considering the associated manual verification and maintenance costs, and the fact that, despite the loss of some precision, the resulting cross reference links would still allow users to easily navigate to the target provisions. This presents a major improvement over the current situation, where cross reference navigation is unsupported.

Cross references outside their container elements. Legal texts are amended regularly. When a provision *P* in a legal text is revised according to a set of amendments, the amendments are cited as part of *P* for traceability. For example, consider "Art. 14. (L. 16 June 1989)" in Fig. 1. Here, the cross reference "(L. 16 June 1989)" indicates that the article incorporates into it the amendments proposed in the cited (amendment) law. In our case study, we came across many amendment citations where the citation appears *before* and not within the amended provision. For instance, instead of the above, one could have "(L. 19 June 1989) ⟨*line break*⟩ Art. 14.". Had this been the case, it would have been unclear, without expert judgment, whether "(L. 19 June 1989)" is attached to the preceding subsection ("Subsection 2. - ...") or to "Art. 14." While automation is feasible for handling such exceptional cross references, one has to weigh the benefits, as the automation could also erroneously attach to the next provision a cross reference that is well-placed (i.e., contained in the element to which the cross reference truly belongs). If automation is found not to be worthwhile, such cross references would need to be manually moved into the right container elements.

Under-specified external cross references. As mentioned in Section III-B, provided with an index of law titles, we attempt to resolve external cross references via applying the TF-IDF similarity metric and finding the closest matching title for a given cross reference. On many occasions, external cross references are under-specified and do not contain adequate information to support such similarity matching. In particular, if the cross reference mentions only the date when the law was signed, e.g., "L. 19 June 1989", but not the title of the law, one has no precise way of determining the special number part of the ELI name (see Section III-B), unless there is only one law signed on that specific date. Due to this factor, automated resolution of external cross references is only a best-effort attempt; a human expert needs to manually resolve any external cross reference that has not been successfully mapped to its ELI name via automation. The quality of resolution for external cross references is quantitatively assessed in Section V-B.

Cross references using rare patterns. Our cross reference detection mechanism is pattern-based. As we have also noted in our previous work [1], we can never be entirely sure that such a set of patterns is complete. In our case study, we came across a small number of cross references ($< 1\%$ of the total number of cross references in the five analyzed codes) that our tool could detect only partially. These cases arose due to incompleteness in our patterns. However, the patterns underlying the problematic cross references were highly irregular and too complex to be captured and interpreted in a straightforward manner. Given the small proportion of cross references missed by our current pattern catalog, we did not find it cost-effective to develop complex pattern matching rules for rare cases. Instead, we recommend that cross references with rare patterns should be resolved manually.

Unlabeled divisions. On occasion, high-level divisions had no explicit labels. To illustrate, consider "Subsection 1. - The mayors" in Fig. 1. This division could well have appeared as "The mayors", i.e., without "Subsection 1. -". In the absence of explicit labeling, high-level divisions are distinguishable by humans via such information as a legal text's table of contents and visual cues such as the font shape and size used for the division titles. We found this information difficult to salvage automatically. In particular, detecting unlabeled divisions would have required parsing tables of contents and considering rich-text metadata (e.g., fonts). This information, while available, varies from one text to another. We therefore found automation not to be worthwhile for dealing with unlabeled divisions. Instead, we recommend that a manual analysis of the table of contents of a given text should be performed during preprocessing for identifying any unlabeled divisions and assigning them dummy (but explicit) labels. These labels allow the markup generator to process these divisions in a standard way, rather than as exceptions. The dummy labels can then be automatically removed during postprocessing.

Variations in division labels. We observed two main sources of variation in division labels. The first are incorrect characters that visually look the same as or similar to the intended characters. Notable examples include using the letter 'l' (ell) instead of the roman number I or the arabic number 1, and the letter 'O' instead of number 0 (zero). The second source of variation are changes in the numbering scheme. For example, a legal text could start with arabic numbering (1, 2, 3, . . .) for the chapters, and then half way through the text, switch to alphabetical numbering (A, B, C, . . .) for the remaining chapters. Although incorrect use of characters is trivial to fix, detecting such issues is difficult and time-consuming, unless one has a-priori awareness about such potential problems. We recommend that one should look specifically for incorrectly-used characters in headers and fix them during preprocessing to avoid latent problems in the markup generation scripts. As for changes in the numbering scheme, we recommend that all the used schemes (e.g., arabic, alphabetical, and alphanumeric) for a given division type (e.g., chapters) should be made explicit in the text conceptual model so that the markup scripts that are automatically instantiated based on the conceptual model will function correctly when encountering different schemes.

Physical lines not aligned with logical lines. Before the advent of markup, laws were written primarily with the intention of being read by humans and not necessarily being machine-analyzable. Incorrectly-used characters (e.g., using the letter 'O' instead of number 0), discussed above, was already a manifestation of this fact. A similar issue arises with line breaks. A physical line break placed in the content to improve the formatting of the print output poses serious problems for automation, due to the difficulty of distinguishing between line breaks that are for layout purposes (and thus should be treated as space characters) and the logical line breaks that indicate the completion of divisions. We used sentence splitting –a commonly available technology in NLP toolkits– to assist in identifying and removing layout line breaks. However, we observed that this technology was not very effective over legal texts. Notably, the technology could not accurately delineate unlabeled lists and alineas that start with lower-case letters and end without a period at the end; such lists and alineas were common in some of the codes in our case study. An even more difficult case is when layout line breaks are inserted at the beginning of new (and unlabeled) sentences. Without carefully analyzing the content and the flow, even an expert may be unable to readily determine the right container for a text segment. Despite its seemingly trivial nature, removing line breaks inserted for layout was the *most time-consuming* part of preprocessing. The issue was further exacerbated when we had to extract the text of laws from PDFs, rather than from editable formats such as MS Word. Our recommendation for dealing with unwanted line breaks is to manually remove as many as possible during preprocessing, and subsequently, inspect the generated XML markup (e.g., via an HTML presentation of it) to detect other potentially-undesirable line breaks.

B. Quality of Automatically-generated Markup

The practical considerations outlined in Section V-A have an impact on quality. For example, our decisions about the depth of text organization markup and the URIs (see Table IV) directly influence the amount of information captured in the markup, and hence the overall accuracy of the markup generation process. Similarly, our choices about what issues to address during preprocessing and what exceptional circumstances to leave to manual corrections in order to increase cost-effectiveness would naturally have accuracy implications.

In this section, we report on the quality of automated markup generation in the specific context of our case study, i.e., in light of the decisions presented in Section V-A. To measure quality, we need a gold standard (ground truth). Since our case study took place in a production environment, building the ground truth manually would have been prohibitively expensive. In fact, had we been able to do this task manually, our automation would have been useless. The procedure that we followed for quality analysis is as follows: Let D be an automatically-generated markup document *before* the document undergoes any manual corrections. Let G denote the same document *after* D has been inspected and the necessary manual corrections have been applied to it. In our analysis, we take G to be the gold standard. Specifically, for each of the five legal codes in our case study, G was the version used for publication on the Legilux portal (see the footnote on page 2), i.e., after internal validation by SCL. To calculate quality statistics, we analyzed for each code the differences between D and G using text differencing.

Table V presents the results for the five legal codes. Each row in the table represents one markup type, e.g., Book or Article. Like in Table III, statistics for letters (L), numbers (N), and dashes (D) have been combined. Furthermore, for the purpose of quality analysis, we distinguish internal and external cross references as ELI necessitates different handling of these markup types, due to the special number that is needed for most external cross references (see Section III-B). FC, PC, and M in Table V are defined as follows: A markup element is fully correct (FC) if it occurs in both D and G and the occurrences match in terms of the markup attributes and the text span. An element is partially correct (PC) if it occurs in both D and G, but modifications were observed either to the attributes or to the text span in G. An element is deemed missing (M) if it only exists in G but not D.

We make two remarks about the above definitions. First, a discrepancy identified between D and G over a nested element, e.g., an Article, would not propagate to its parent (container) element, e.g., Chapter, as long as the parent element has the correct attributes and text span in D. Second, cases where markup is present in D but not in G always manifest as PC elements in G. We did not thus count such cases separately.

To quantify the overall quality of automated markup generation for a given markup type, we define a metric, Q, computed as the proportion of FC over the total number of elements of the given type (i.e, $FC+PC+M$). Because of the stringent accuracy requirements for legal documents, any automatically-generated markup for these documents has to be thoroughly reviewed to ensure correctness. What Q, or more precisely $(100\% - Q)$, represents is the additional effort that one incurs during the review process over fixing problems in the generated markup. We note that Q is a *conservative* metric for quality: it treats PC and M the same way although fixing a partially correct markup element is, in general, much easier than providing markup from scratch for a missing element.

TABLE V
STATISTICS FOR AUTOMATED MARKUP GENERATION

	Civil Code				Commerce Code				Penal Code				Code of Criminal Procedure				New Code of Civil Procedure			
	FC	PC	M	Q	FC	PC	M	Q	FC	PC	M	Q	FC	PC	M	Q	FC	PC	M	Q
Part	0	0	3	0,00%	0	0	0	NA	0	0	0	NA	0	0	0	NA	2	0	0	100,00%
Book	2	1	0	66,67%	4	0	0	100,00%	2	0	0	100,00%	3	0	0	100,00%	11	0	0	100,00%
Title	36	0	0	100,00%	17	4	0	80,95%	10	1	0	90,91%	15	0	0	100,00%	90	0	0	100,00%
Chapter	129	0	2	98,47%	14	0	0	100,00%	70	12	4	81,40%	33	0	0	100,00%	19	0	0	100,00%
Section	154	0	0	100,00%	9	3	0	75,00%	38	1	7	82,61%	34	2	0	94,44%	42	0	0	100,00%
Subsection	39	4	0	90,70%	0	0	0	NA	0	0	0	NA	7	0	0	100,00%	30	0	0	100,00%
Article	2305	10	1	99,53%	258	3	0	98,85%	655	15	1	97,62%	529	0	0	100,00%	1321	1	0	99,92%
Paragraph	33	0	0	100,00%	14	0	0	100,00%	64	0	0	100,00%	541	0	1	99,82%	206	0	0	100,00%
Alinea	3358	53	63	96,66%	461	10	18	94,27%	1075	17	2	98,26%	1278	23	14	97,19%	2297	12	7	99,18%
L/N/D	318	12	31	88,09%	50	0	35	58,82%	467	1	3	99,15%	320	4	1	98,46%	307	3	32	89,77%
Internal CR	436	10	0	97,76%	109	1	0	99,09%	422	33	6	91,54%	336	12	0	96,55%	367	21	1	94,34%
External CR	245	297	9	44,46%	65	47	10	53,28%	106	317	28	23,50%	318	396	3	44,35%	74	355	3	17,13%

Collective statistics for the five legal codes in the case study

FC ratio (Q) across all element types: 90,69%
PC ratio across all element types: 7,96%
M ratio across all element types: 1,35%

FC ratio for Internal CRs only: 95,21%
FC ratio for External CRs only: 35,55%

Legend
FC: Fully Correct
PC: Partially Correct
M: Missing
Q: Quality (FC / Total)

When considering all the 21111 markup elements in the five codes (see Table III), we observe that: 90,69% of the markup elements are fully correct, 7,96% of the elements are partially correct, and the remaining 1,35% are missed.

Problems in the text organization markup elements (i.e., all elements except cross references) originated primarily from the labeling and line-break issues which we had not handled properly at the preprocessing stage. These issues were elaborated in SectionV-A. As indicated by Table V, the majority of the problems in the text organization markup have to do with alineas, letters, numbers, and dashes. The challenge here is that letters, numbers and dashes do not have a fixed hierarchical order, and further, are sometimes left implicit, meaning that they are separated only by line breaks and without explicit delimiters. Consequently, letters, numbers, and dashes may erroneously get classified as alineas and thus not correctly be nested into the alinea where they belong. In each such case, the text span for the parent alinea would be incomplete. The markup generated would thus be only partially correct.

As for cross references, we observe that automation is highly effective for internal ones, with 95.21% of them being annotated with fully correct markup across the five legal codes. Nevertheless, only 35.55% of external cross references are annotated with fully correct markup. The main reason for this low performance is the under-specification of cross references, in turn complicating the identification of the ELI special number that needs to be assigned. This issue was already discussed in Section V-A. The issue is often exacerbated by the fact that the same under-specified cross reference can be repeated dozens of times. To improve the quality of markup automation for external cross references, one necessarily has to take into account additional information, e.g., the context in which these cross references appear. We leave this to future work. However, we need to emphasize that the markup we generate for external cross references is still highly usable: virtually all problems in the markup for external cross references are localized to the (typically two-character) special number at the end of the ELI resource name. This number, once known, is very easy to insert manually into the markup.

In summary, the results of Table V provide confidence that the amount of manual effort required for fixing problems in the automatically-generated markup is low, suggesting that automation is effective for structural markup generation.

VI. THREATS TO VALIDITY

As stated earlier, the case study that we reported on in this paper falls under *action research*. Kock [18] points out three main validity threats for action research: uncontrolability, contingency, and subjectivity.

Uncontrolability has to do with the researcher's degree of control over the case study's environment and having unexpected internal or external factors impacting the case study. Our case study spanned five legislative codes, rather than just one. This to a large extent mitigates the risk of external factors affecting the case study.

Contingency is synonymous with the standard *external validity* threat, and has to do with the generalizability of action research findings. With regard to contingency, we note that our approach has also been applied, albeit at much smaller scales, to Canadian legal texts [1], [10]. While mitigating contingency requires additional large case studies, in our limited analysis of Canadian legal texts, we observed the main characteristic that has motivated our current automated solution, namely the existence of large volumes of heterogeneous legal texts without precise markup. We therefore anticipate our solution to offer value in contexts beyond that of our case study.

Subjectivity has to do with the direct involvement of the researchers as agents of change. This may lead to biased results. In our case study, we were driven by the specific needs of our partner, e.g., regarding the choice of legal texts to analyze and the type of markup to generate. Our limited influence on these key decisions helped mitigate subjectivity. Another dimension of subjectivity has to do with the environment. In a change-averse environment, any change, beneficial or not, may be (subjectively) perceived as negative and met with resistance; whereas in a change-friendly environment, the attitude towards change is generally positive. In our case study, our partner was very forthcoming and provided us with constant support and commitment. This was an important factor in the success of our case study; the results could have been different had we not received such level of support from our partner.

VII. RELATED WORK

We distinguish two categories of work on structural legal markup: editing tools that facilitate the application of manual markup, and techniques for automated transformation of non-markup, e.g., plain-text, documents into markup documents.

In the first category, there are a number of authoring tools with a dedicated focus on legal texts. We outline two of these: LIME and AT4AM. LIME [19] is an open-source editor for writing legal texts in a markup format. AT4AM [9], [20] is a web-based authoring tool developed to assist EU-member countries in amending EU parliamentary documents that are already in an XML format. AT4AM provides basic features for manual insertion of legal markup into texts and has already been applied with success in practice. When using such tools, one needs to manually select the text segments to which markup needs to be added, and then apply the appropriate markup tags. Although such tools ease the manipulation of legal markup, specially when new texts are being drafted, such tools alone are not sufficient for adding markup to large volumes of legacy (non-markup) texts. Automated assistance for markup generation therefore remains a necessity.

With regard to the second category of related work, namely, automated markup generation, Sannier and Baudry [21] develop a customizable structural parser for standards and regulations. Their approach, which is based on NLP and modeling, helps identify generic divisions and other constructs, such as requirements and recommendations, in the text of standards and regulations. Breaux [3] and Massey [4] devise pattern-based parsers for extracting the structure of the US Health Insurance Portability and Accountability Act (HIPPA). Nakamura and Katuka [22] build a converter tool for transforming Japanese municipal regulations, already provided in an XML format, into relational databases and HTML pages. Boer et al. [23], without providing details, mention a parser they have built for transforming rich-text documents to the MetaLex format. Kerrigan et al. [24] describe a tool for transforming environmental regulations from PDF and HTML into XML. Finally, Zeni et al. [5] propose a broad markup framework, GaiusT, for semi-automated parsing and annotation of legal texts. Among various other metadata items, GaiusT identifies articles and their subdivisions as an intermediate step for extracting semantic metadata such as rights and obligations.

Our markup generator, which is the basis for the work we report here, takes inspiration from several of the above work strands, as we explain in our previous work [1]. Nevertheless, our case study in this paper is focused on something different: to highlight the issues that arise when automated markup generation is applied at large scales, and how to cost-effectively deal with these issues. The above work strands do not specifically address the scalability challenge for markup generation and how to increase the industry readiness level of the existing (mainly academic) markup generation tools.

VIII. Conclusion

In this paper, we reported on a case study targeted at generating structural markup for Luxembourg's legislative codes. Structural markup is a prerequisite for performing more advanced analysis over legal texts and for the systematic elaboration of legal requirements. An important observation from our case study is that, despite its conceptually simple nature, building structural markup for legal texts cannot be automated with full accuracy. One therefore has to carefully consider how far it is worthwhile to go with automation, and what aspects of structural markup would better be done manually. Drawing on the experience gained from our case study, we discussed a number of important factors that affect the cost-effectiveness of automation for structural markup generation. We then evaluated the quality of automatically-generated markup when these factors are accounted for.

In the future, we would like to conduct case studies on *semantic* legal markup, e.g., for modalities, conditions, and rules. This will enable us to examine the quality of automatically-generated semantic markup in real conditions, where one has to handle potentially thousands of provisions. Another aspect of our future work is to improve the customization facilities that instantiate our NLP scripts for a given legal text.

Acknowledgements. This work is partially supported by SCL and FNR under grant FNR/P10/03.

References

[1] N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. Briand, "An automated framework for detection and resolution of cross references in legal texts," *REJ*, vol. 22, no. 2, 2017.

[2] S. Erdelez and S. O'Hare, "Legal informatics: Application of information technology in law," *Annual Review of Information Science and Technology*, vol. 32, 1997.

[3] T. Breaux, "Legal requirements acquisition for the specification of legally compliant information systems," Ph.D. dissertation, North Carolina State University, 2009.

[4] A. Massey, "Legal requirements metrics for compliance analysis," Ph.D. dissertation, North Carolina State University, 2012.

[5] N. Zeni, N. Kiyavitskaya, L. Mich, J. R. Cordy, and J. Mylopoulos, "GaiusT: supporting the extraction of rights and obligations for regulatory compliance," *REJ*, vol. 20, no. 1, 2015.

[6] "Akoma Ntoso," http://www.akomantoso.org.

[7] "ELI - the European Legislation Identifier," http://eur-lex.europa.eu/eli-register/about.html.

[8] M. Palmirani, L. Cervone, O. Bujor, and M. Chiappetta, "RAWE: an editor for rule markup of legal texts," in *RuleML'13*, 2013.

[9] "AT4AM," hhttp://www.at4am.org.

[10] N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. C. Briand, "Automated classification of legal cross references based on semantic intent," in *REFSQ'16*, 2016.

[11] M. Denscombe, *The Good Research Guide: for small-scale social research projects*, 4th ed. McGraw-Hill, 2010.

[12] M. Palmirani, R. Brighi, and M. Massini, "Automated extraction of normative references in legal texts," in *ICAIL'03*, 2003.

[13] Cunningham et al, "Developing Language Processing Components with GATE," 2016. [Online]. Available: http://gate.ac.uk/sale/tao/tao.pdf

[14] "Formex," http://formex.publications.europa.eu/formex-4/formex-4.htm.

[15] "MetaLex," http://www.metalex.eu.

[16] "LexML," http://projeto.lexml.gov.br/documentacao/resumo-em-ingles.

[17] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, 1988.

[18] N. Kock, "The three threats of action research," *Decision Support Systems*, vol. 37, no. 2, 2004.

[19] "LIME - The Language Independent Markup Editor," http://lime.cirsfid.unibo.it.

[20] C. Fabiani, "AT4AM: The XML Web Editor Used by Members of European Parliament," http://tinyurl.com/hr6s6l2, 2013.

[21] N. Sannier and B. Baudry, "INCREMENT: A mixed MDE-IR approach for regulatory requirements modeling and analysis," in *REFSQ'14*, 2014.

[22] M. Nakamura and T. Kakuta, "Development of the elen regulation database to support legislation of municipalities," in *JURIX'14*, 2014.

[23] A. Boer, R. Hoekstra, and R. Winkels, "METALex: Legislation in XML," in *JURIX'02*, 2002.

[24] S. Kerrigan and K. H. Law, "Logic-based regulation compliance-assistance," in *ICAIL'03*, 2003.