A tool supported methodology for BPR in Public Administrations

A. Ciaghi, A. Mattioli* and A. Villafiorita

Fondazione Bruno Kessler,
Via Sommarive, 18-38100 Trento (IT), Italy
E-mail: ciaghi@fbk.eu
E-mail: amattioli@fbk.eu
E-mail: adolfo.villafiorita@fbk.eu
*Corresponding author

Abstract: In Public Administration (PA), business process redesign is an essential activity for reducing costs, has strict connections with laws and is carried out by actors with different backgrounds. This work proposes an approach where changes in the law are mapped in process diagrams in order to highlight and review the impact on processes. This allows for a stricter collaboration among the different people usually involved in Business Process Reengineering (BPR). The methodology takes advantage of the UML notation and is supported by a tool, Visual Law Process Modeller (VLPM), which helps to keep traceability between processes and laws. Finally, we discuss an example in which the methodology was applied to model the transition from electoral processes to e-electoral process.

Keywords: BPR; business process reengineering; process redesign; law; e-voting; e-democracy.

Reference to this paper should be made as follows: Ciaghi, A., Mattioli, A. and Villafiorita, A. (2010) 'A tool supported methodology for BPR in Public Administrations', *Int. J. Electronic Governance*, Vol. 3, No. 2, pp.148–169.

Biographical notes: Aaron Ciaghi is a Graduate Student at the University of Trento, Italy and Georgia Institute of Technology. He has been working on processes and law modelling since 2007. He is particularly interested in BPR, software project management and HCI.

Andrea Mattioli obtained his Masters Degree in Computer Science at the University of Trento in 2006. He is interested in requirements engineering, validation and verification of complex system. Currently he is working at the Scientific Research Centre of the Bruno Kessler Foundation (Trento, Italy) on the ProVotE and LTPDA projects.

Adolfo Villafiorita is a Researcher at the Scientific Research Centre of the Bruno Kessler Foundation. His research interests are software engineering, formal methods and critical applications development.

1 Introduction

In several countries, Public Administrations (PA) are turning to IT in order to optimise law making and to facilitate accessibility to public services and regulations. This involves a simplification and formalisation of law-making standards. In this paper we focus on the Italian PA efforts to improve public action effectiveness, reduce administrative costs, shorten the time needed to perform its various activities and, most importantly, increase the participation and control of citizens (Lazzi, 2000).

Not surprisingly, one of the tools to enact this reform is the application of Business Process Reengineering (BPR) techniques. At a very high level and with several simplifications, BPR consists of the following main steps (further details in Valiris and Glykas (1999), and Xenakis and Macintosh (2007)): defining the goals of the reengineering activity (e.g., increase efficiency, reduce costs, introduce a new function), identifying the processes, modelling and analysis of the current processes (the *as-is*), devising the new processes (the *to-be*) and actuate them. Both the *as-is* and the *to be* are represented in some textual or graphical representation.

This approach is not new, nor is its application in PA: we mention, as examples, the initiatives originated by the US federal government and by the US Department of Defense (USA National Performance Review, 1993; Lazzi, 2000) or its use in taxation or in healthcare public services (Thaens et al., 1997; Willcocks et al., 1997).

It is also well known that representing procedures by means of workflow diagrams has various advantages and that the practice of using business process methodology to deal with change management is quite widespread (Wastell et al., 1994). In Kettinger et al. (1997), for example, review of existing methodologies, tools and techniques for business processes can be found, while in Hammer and Champy (1993) the authors present a way of improving organisational, team, and individual performance focusing on customer satisfaction and work processes.

In PA, business process redesign is an activity which involves, independently or in collaboration, lawmakers who amend laws, process designers who try to optimise existing processes, and software developers, to support existing procedures with technology. Modelling facilitates the communication and understanding of the actual organisation among these agents and is helpful in building a shared vision between domain experts and technicians. Moreover, it provides an easier way of analysis in order to evolve towards efficient and higher quality procedures.

Unfortunately, according to Alpar and Olbrich (2005), it is not possible to transfer e-business solutions and development approaches directly to the PA. In fact, one of the major difficulties encountered in this domain is the strong dependency between processes and laws. Since PA processes are defined and regulated by laws, any implementation of a *to-be* requires a parallel action on both the redesigning of processes and on the introduction of law changes. This means that the current law (in a sense: rather than the processes), must be considered as the constraint, the engine, and the target of the reengineering activity (Lazzi, 2000). This link between models and laws raises further issues related to the maintainability of the models over time, since it is necessary to guarantee coherency of the models with the laws in order to have the models retain their value.

For this reason, even if several strategies have been proposed to model processes, spanning from workflow nets to event-driven process chains, from flow-chart diagrams to UML Activity diagrams (Russell et al., 2006), process algebra and

Business Process Execution Language (BPEL) for Web Services, the attempts to model laws and processes are quite recent. In Heib et al. (2003), for example, the authors propose a first attempt to optimise e-Government processes using event-driven process chains, while Olbrich and Simon (2008) proposed an approach to translate the paragraphs into process models using a Semantic Process Language (SPL). The approach we present in this paper is based on a modelling methodology and a tool that we implemented to support it.

The approach we propose has some similarities with the concepts described in Xenakis and Macintosh (2007), but it is more oriented towards increasing *traceability* between law and processes, with the goal of helping the law makers elaborate models in collaboration with software developers or process engineers, and understand the impact of law or process changes to their counterparts.

Although our methodology is applicable to several domains, we focused on PA procedures. In particular, we chose the electoral activities as a working example, mainly because we are involved in a project with concerns in the transition from paper-based voting to electronic voting, and because it is a well known and procedurally rich scenario which particularly requires the contribution of legal and PA experts as stated in Xenakis and Macintosh (2007).

The methodology is supported by a tool, Visual Law and Process Modeller (VLPM), to automate the extraction of some information, analysis and law traceability. Automatic generation of some documentation is also possible, as well as the generation of skeletons of law modifications based on the changes undergone by processes defined by the original law. VLPM can be integrated in the workflow of the Italian PA as it takes as input the XML standard format in which Italian laws can be made available[1] (Bolioli et al., 2002; Lupo, 1999, 2002; Lupo and Spinosa, 2001). VLPM has been applied in two real case studies, namely the modelling of the electoral law and processes in the Province of Trento and the law for the introduction of an e-voting system for a local poll in two municipalities of the Autonomous Region of Friuli Venezia Giulia. In the former case the methodology and the tool were adopted to model town and provincial elections in the Province of Trento (Italy), producing diagrams which describe about 80 processes, 30 actors and over 90 entities, while the two main reference laws include from 80 to 100 articles each. VLPM is built on top of Visual Paradigm (a commercial UML modelling tool) and it is freely available from http://ed.fbk.eu/vlpm. This paper extends (Ciaghi et al., 2009) by providing information about the modelling methodology and more information about the tool and our case studies.

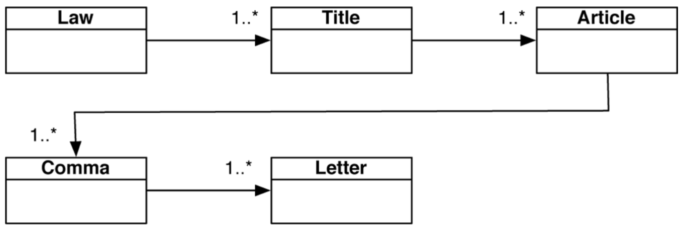
This paper is structured as follows: Section 2 describes one of the main approaches which the Italian PA has been taking to organise and structure legal documents. Section 3 introduces the proposed methodology and its guidelines. Sections 4 and 5 respectively describe the advantages of keeping traceability between laws and processes and the tool which supports the methodology. Section 6 shows a case study concerning a small election in Friuli Venezia Giulia, Italy where the methodology and the tool were used. Last, Section 7 draws conclusions and discusses possible future work.

2 Representing laws in XML

The production of norms in Italy presents both quantitative and qualitative problems (Marchetti et al., 2002). The proliferation of laws and other acts makes it difficult to determine how many laws are in effect (a rough estimate is of about 18,000 articles, while France, for example, is estimated to have only 7000 articles in effect). Italian legal documents are not homogeneous in formality, style and structure. As a result, a number of conflicting (and sometimes unconstitutional) rules are present in the legal system. In addition to this, norms can be created by several different Authorities (such as the Telecommunication Authority) and external legislation bodies (such as the European Union), increasing the ambiguity in legal documents classifications (Marchetti et al., 2002).

One of the countermeasures adopted in Italy to regulate law-making is a circular letter – "Rules and Suggestions for Drafting Legislative Texts" – published on April 20, 2001.[2] The circular letter defines the following structure for Italian legislative texts: a law is divided in 'Titles' which contain 'Articles' that are divided in 'Commas' or 'Paragraphs'.[3] Law paragraphs contain the actual text of the law and can be further subdivided in 'Letters', which are an alphabetically ordered list of clauses. Figure 1 shows this structure in the UML notation.

Figure 1 Law elements class diagram



Mark-up languages, and in particular XML, can provide interesting results at both ends of the legislative process: at the drafting stage, enforcing some or all the drafting rules defined for norms; at the accessibility stage, fostering easy and sophisticated searching and rendering tools for the public at large. Furthermore, XML may constitute a great influence on several other aspects of the legislative process, such as providing support for the consolidation of laws, rationalising the legislative process, improving the referencing and connections among the norms (Marchetti et al., 2002).

The 'Normeinrete'[4] project, started in 1999 by the Italian Ministry of Justice and several other Italian public institutions, defined an XML format for Italian legislative texts that allows the marking of all the elements constituting a law and thus facilitates electronic processing of laws (Bolioli et al., 2002; Lupo, 2002).

'Normeinrete' provides several DTDs to support the representation of legal texts written before and after the publication of Presidenza del Consiglio dei Ministri (2001). The 'loose' DTD supports the former type of texts, while the 'strict' DTD supports and enforces the well-formedness of the latter, namely the adherence to the structure depicted in Figure 1. Both DTDs describe a structure made of a header, an optional preamble, a hierarchical structure (called *articolato*) containing the elements shown in Figure 1, a conclusion and a variable number of annexes. These are complemented by the initial and final formulas of the Italian law texts (Marchetti et al., 2002). Our tool and our methodology focus on the *articolato* and therefore we will disregard the other components of the text from now on.

In addition 'Normeinrete' introduces specific tags to mark resources and entities. This results to be particularly helpful to automate the modelling of responsibilities in processes and extracting a preliminary list of used objects.

In order to univocally identify laws and their elements, 'Normeinrete' uses a URN based system (Lupo and Batini, 2003; de Oliveira Lima et al., 2007). The URNs are defined as a combination of elements according to a specific grammar. The basic elements are: the name of the promulgating authority, type of norm, date, number and a set of more detailed specifications when needed (Lupo and Batini, 2003).

3 The modelling methodology

The XML representation of laws provides several advantages among which are eliminating (some) syntactic ambiguities, and providing formal and automated cross-references. However, when the target of the law is the definition of the operational aspects of the PA, i.e., how the PA works, and the target of the work is the reengineering of processes, workflows and workflows diagrams are more commonly used. As the size of the model increases, issues concerning composition, uniformity (i.e., the same notation is used coherently in all models) and quality of models written by different people quickly arise. For this reason, we devised a modelling methodology and modelling guidelines which are meant to support functional analysts in defining their processes. The modelling phase is a pivotal step to deal with the problem of change management, especially when manual activities are altered by the introduction of some kind of technology. With the purpose of dealing with it and to support process reengineering, we distinguish two phases: business architecture (*as-is*) and software delivery (*to-be*).

The modelling of business architecture and software delivery requires us to consider several aspects *simultaneously* (Katranuschkov, 2006): business processes, actors, inputs and output information, trigger conditions, final outcomes. To capture all this information systematically without decoupling the link with the law, the methodology is organised in three phases, of which the second and the third can be executed in parallel:

- preparation
- static process analysis
- dynamic process analysis.

We chose the UML notation to concretise the methodology mainly because software engineers are familiar with it. Some kinds of diagrams (e.g., use cases) are easily understood by domain experts who are not conversant with software development and it is supported by several COTS tools. The methodology is not strictly bound to UML. However, other notations are not flexible enough to represent some concepts included in the methodology (e.g., multiplicities); on the contrary, too much flexibility could make it difficult to enforce its application.

The methodology mainly uses two types of UML diagrams: *use case diagrams*, which are employed to provide a static hierarchical process view that allows us to concentrate on actors and their responsibilities, and *activity diagrams*, which deal with the dynamic aspects of the system, rendering processes as actions which transform entities and their state. Activities are viewed as *transformation functions*, i.e., they change data and states of their input resources. Furthermore, the modelling borrows and extends some well known formalisms from the RACIV responsibility matrix, and from the CRUD matrix, to characterise relationships between processes and actors, and between processes and resources respectively.

Finally, the transition from the *as-is* model to the *to-be* can be further supported by a methodology to evaluate possible alternatives (Bryl et al., 2007) or by using simulation techniques like in Hlupic (2003).

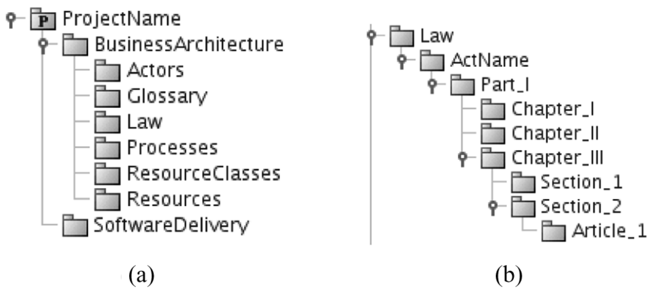
3.1 Preparation

In this phase the data and structure of the processes are prepared through the following activities:

- *Actors and resources identification.* For what concerns actors and resources identification, we further split this phase into:
  - *Actors identification.* Aims at extracting roles, actors (humans or organisations) and groups (e.g., offices) which are involved in the domain under analysis and their connections with other actors.
  - *Resource classes identification.* Types of the assets involved and their relationship with processes and other assets.
  - *Resources identification.* Instance of assets types which are needed to perform the different operations. This phase requires an understanding of the evolution in *state* and *value* of the assets, i.e. how they change after executing a process and if they become less or more critic.
- *Stereotypes identification.* The domain-specific characterising aspects and concepts are identified and agreed among all parties. This allows us to understand to which extent the methodology fits the specific problem and to assess the extension points.
- *Terminology identification.* The domain specific glossary is collected and agreed among all parties.
- *Laws collection.* The enumeration of laws which rule or influence the domain under analysis.

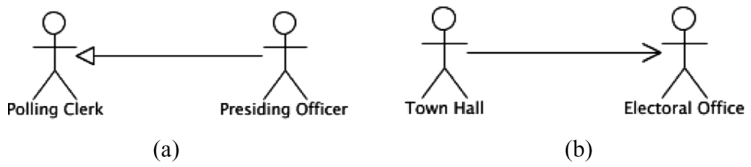
Figure 2(a) shows the main concepts organised in different packages after the preparation phase. The *law package* organises the laws regulating the processes. Each law has its own sub-package which is hierarchically structured in other sub-packages. Each sub-package is a non-terminal law element (e.g., a chapter or a section of the law) while leaf elements like law paragraphs are represented in UML by means of *instance specifications*. Each package may contain an Object Diagram modelling its internal structure and the instance specifications. Title and description (e.g., the text of the law) are attached to both terminal and non-terminal law elements. Figure 2(b) sketches the *law package* structure of an act and its possible subdivision. The activity of representing the elements of a law in UML is carried out completely by our tool.

Figure 2 Package structure: (a) structure of packages and (b) the law subpackage



The *actor package* contains all *UML*-actors representing roles or organisations which can contribute in the realisation of some processes. It can include one or more use case diagrams describing the relationships among them. Each actor is described including what the actor represents (office, person, role, ...), its functions and responsibilities. One or more use case diagrams underline relationships among actors. Possible relationships among actors are *generalisation* and *association*. Figure 3(a) and (b) show an example taken from the electoral domain.

Figure 3 Actors relationships: (a) actor generalisation and (b) actor association

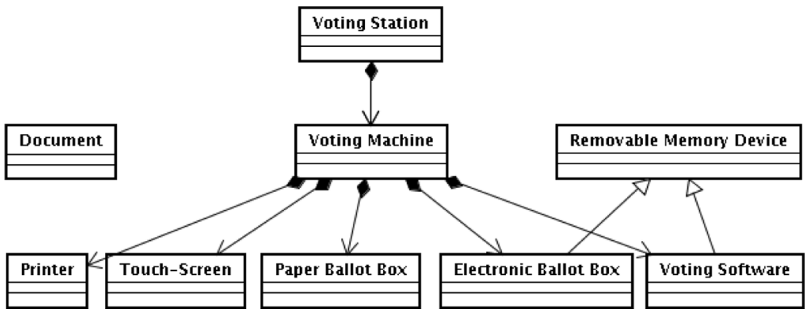


The *resource class package*, *classes* for short, holds all the resource types used as input or output by some process. A class is the abstract definition of a resource related to a process, while the resource itself is an instance of that class lowered in a specific context. An example: the abstract concept of *ballot box* (class) is associated with several instances which describe its evolution during processes execution, for example a *state* change from ‘open’ (e.g., during the voting phase) to “closed & sealed” (e.g., right after the election is over). Also its *value* changes accordingly, which is low when the *ballot box* is empty (before the election) but which becomes very high after a voter deposits his marked ballot.

One or more class diagrams describe the relationships among classes using standard UML relationships. This is in order to provide better characterisation through dependency, aggregation/composition and generalisation relations. Similarly to UML specification, a dependency relationship is used whenever an object defined by a class depends on another class. It is possible to mark a class with the ≪*Tool*≫ stereotype whenever instances of this class are used to accomplish a task. For instance, Figure 4 shows the relationship between a voting machine and its possible components.

Collecting all the instances of the resources involved in some process flow is responsibility of the *resource package*. It mainly contains *UML*-ObjectNodes whose type will be one of the classes defined in the *resource classes* package. Each resource is characterised by a description, one or more *states* (e.g., *open* or *closed*) and

Figure 4 A class diagram example in the resource class package



associated *values* (e.g., *low* or *high*), and a class. The class is one of those collected in the *classes* package.

The *glossary package* includes the domain specific terms used in the model. They can be represented by *UML*-classes, i.e. each word or acronym is described by means of an UML-class.

3.2 *Static process analysis*

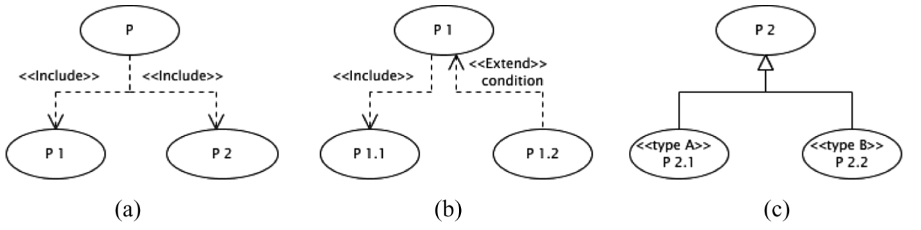
This phase concentrates on actors (who) and processes identification (what). In particular, static view of processes focuses on recognising attributes which are independent from their execution. UML use case diagrams are employed to elaborate this perspective, aiming at:

- *Process breakdown.* Using a top-down analysis, processes are organised as closely as possible into a tree structure, by performing one-to-many decompositions. It includes:
  - Creating a hierarchical process structure.
  - Defining processes dependencies.
- *Actors and responsibilities.* Actors and their responsibilities are associated in the process breakdown structure.
- *Law association.* Law paragraphs are defined and connected to processes.

3.2.1 *Process breakdown*

Processes are structured using a hierarchy of use cases, typically (but not necessarily) a tree: the higher levels represent *abstract processes* (i.e. containers) while the leaves represent *executable processes* or *tasks*. In Figure 5(a) a generic root process P is decomposed in two sub-processes P_1 and P_2. This is also called the first *level of the hierarchy*. P_1 and P_2 can then be further defined using one use case diagram each in order to obtain two new decomposition levels. To keep the overall organisation as simple as possible, each use case diagram shows only one decomposition level. If P can be decomposed in P_1 and P_1 in $P_{1.1}$ then two different use case diagrams should be used.

Figure 5 Static view: relationships between a super-process and its sub-processes using UML: (a) a decomposition level; (b) decomposition types and (c) generalisation



Processes can be 'coloured', labelling them with one or more stereotypes. This results useful to create group of processes which go across the fixed decomposition hierarchy in order to recognise similar types of processes inside the domain. To make an example, local elections and general elections are similar enough[5] to be modelled together, but election specific processes can be marked according to their election's type when differentiation is needed using ≪*local*≫ or ≪*general*≫ stereotypes, for instance.

Another important point for the static view is the relationship among processes. In particular we identified the following types of static relationships:

- decomposition
- exceptional behaviour
- generalisation.

A normal decomposition is represented using the UML-*include* stereotype relationship. Its meaning is that the sub-process is needed for the realisation of its super-process and always executed. The second kind of relationship, exceptional behaviour, is mapped to a UML ≪extend≫ relationship. In this case the sub-processes are executed only if certain circumstances are verified, for example, if a special condition holds. The latter can be specified on the connection itself. Figure 5(b) shows these two types of relationships. In the electoral domain. An example where *extend* can be used is the case in which special procedures are required, e.g., voters who require assistance to vote due to some physical impairment or inability.

Generalisation (Figure 5(c)), occurs when an abstract process needs to be refined according to a specific instance. The tally of results, for instance, can be thought as an abstract process to apply the counting procedure. In that case, it has to be refined according to the specific election type (e.g., a referendum or an election), each one related to peculiar counting algorithms.

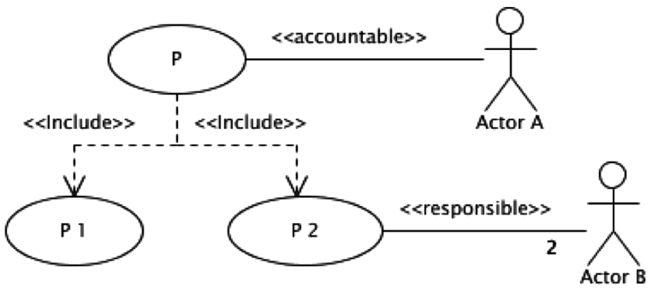
Start and end timing constraints (or other information to include in the process description) can then also be attached to each leaf process to explicitly define when they are supposed to be executed. In UML, this can be carried out by means of tagged values.

3.2.2 Actors and responsibilities

The next step in modelling processes from a static point of view is the specification of participating and responsible actors. This information describes who does what during the execution of a process.

After actors have been identified (Section 3.1), it is possible to extend the hierarchical process structure with actors' information. Figure 6 represents a decomposition level where *Actor A* is accountable for the super-process and two *Actor B*s are responsible for one of its sub-processes. Notice that the UML differs from other notations by being flexible enough to allow the specification of the number of participating actors. There are several examples where this information is useful, for example when two witnesses are required to sign a document. The relationships between actors and processes can be labelled according to the RACIV (Responsible, Accountable, Consult, Informed, Verifies) format and some implicit assumptions can be made. For example, when an actor is accountable for an abstract process, it is implicitly accountable for all of its sub-processes if not otherwise specified, or she or he is also responsible for a sub-process if no other responsible actor is explicitly stated.

Figure 6 Some RACIV relationships among processes and actors



3.2.3 Law association

The static perspective of processes usually derives from domain experts or from a law describing some procedures. Whenever it is necessary to trace a relationship between the process and the law a general UML dependency can be added between the process and the law element (e.g., a law paragraph) contained in the *law package*.

Because assigning dependencies between processes and laws is a labourious task, this activity is mainly intended to be tool supported and will be discussed in the next section. In short, the tool automatically provides a link between a process and a law element.

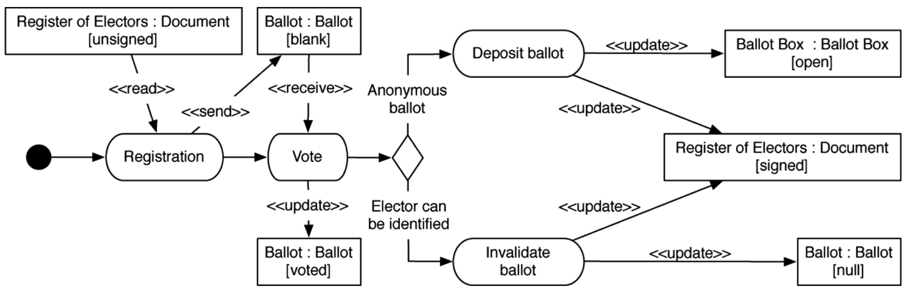
3.3 Dynamic process analysis

This phase analyses the evolution in state and value of resources and activities. UML activity diagrams describe the workflow of processes, emphasising sequential and parallel activities, whose resources are needed and how their states evolve, i.e., how they change after execution.

The diagram also highlights the resources on which the processes operate. Connections between processes and resources are labelled with one or more of the following stereotypes: create, read, update, delete. In addition to the standard notation borrowed from the CRUD matrices, it is also possible to specify *use*, *send* and *receive* as stereotypes. *Use* suggest that the linked object is necessary to perform the task but does not provide data by itself. *Send* and *receive* point out that the responsibility or accountability for the involved resources is changing from the accountable actor of the source process to the accountable actor of the destination process. It usually also implies a 'physical' movement of an asset. The main constraint is that if a connection between process P_i and resource R is labelled *send*, then there also exists a corresponding process P_j linked to R marked with the *receive* stereotype.

Figure 7 shows a simple example of activity diagram from the electoral domain. The register of electors is checked by polling officers to know if the elector is eligible to vote. If so, the responsibility of the *blank* ballot is then transferred to the voter who votes and changes the *state* (and value) of the ballot. After that, either the ballot is anonymous and cast or it is recognisable (e.g., there is anything written to identify the elector) and hence invalidated. If not otherwise specified, the input resource is considered unchanged after execution. For example, the first process *reads* the register of electors leaving it unchanged while the following ones *updates* the register (e.g., the voter has voted). This allows to explicitly specify what action is performed on what entities and how the resource changes its state.

Figure 7 Activity diagram example



Stereotypes can also be applied on the resources themselves in order to highlight crosswise concepts. The methodology suggests the following base stereotypes but other domain-dependent ones can be added: file, document, object (physical), program.

4 Linking laws and models

Traceability between laws and processes must be maintained in order to keep the model up to date with the law and in order to determine how BPR interventions affect the original law. More specifically, traceability allows us to:

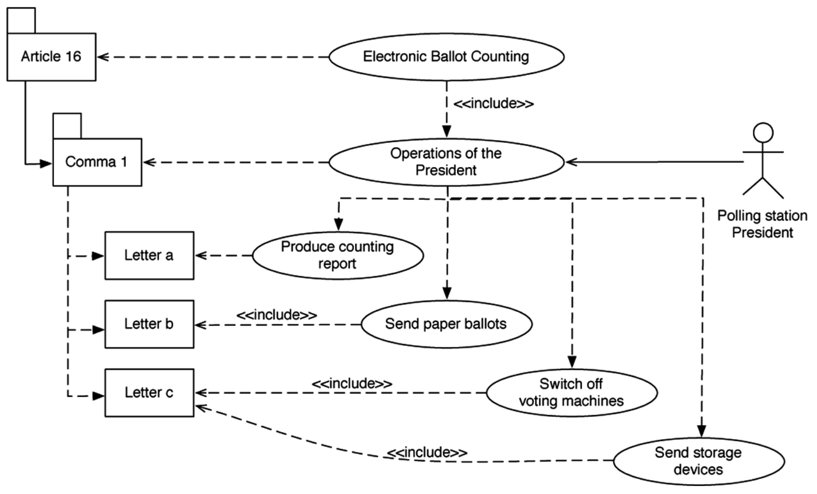
- Justify the existence of a particular process by providing a reference to the parts of the law that define it. This also allows us to link the process to all the constraints in the law that regulate it.
- Understand the impact of a change in the law. When a change is made to the law, being able to identify which processes are defined (or regulated) by the

modified part of the law allows us to modify the process model accordingly. By looking at the model, it is then possible to determine what processes 'interact' with the processes affected by the change in the law. The modification can then be propagated to all the relevant processes and makes the model be always up to date.

- Understand the impact of a change in the process model. The reengineering of processes may result in a need to modify some parts of the law (e.g., a change in the electoral processes should be reflected upon the electoral law in order to make the modified processes valid and legal). Maintaining law-model traceability allows us to automatically identify which parts of the law should be amended by tracing back to the parts of the law that originally defined the modified processes.

The methodology presented in the previous section provides a strategy to maintain traceability, by modelling the information about processes ('process tree') as use cases linked to the model of the law structure (specified as packages and instance specifications) as shown in Figure 8. Note that the model is fully documented since all its elements contain their relevant description (extracted from the text of the law or written by the process analysts).

Figure 8 Example of an article and its defined processes

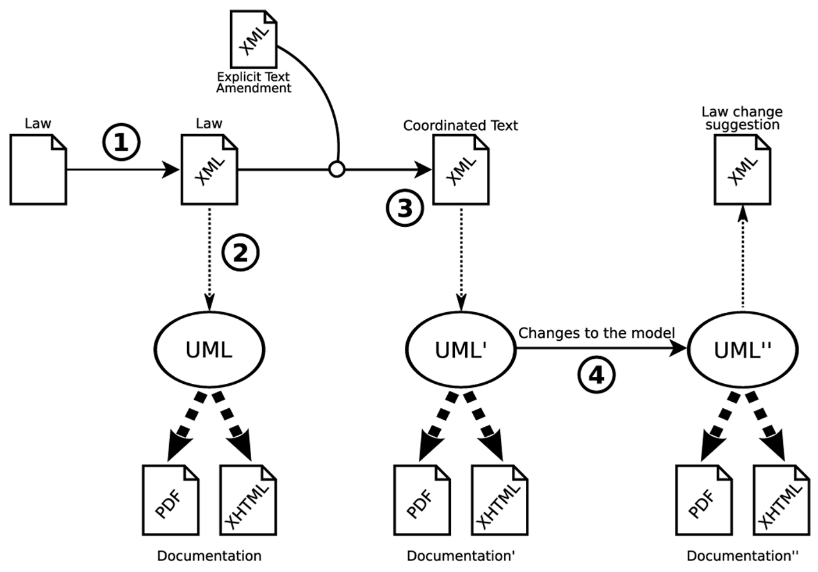


It is not necessary to explicitly draw the relationship among processes and law divisions. In fact, 'Normeinrete' URNs can be used to link the process model with the text of the law, since URNs univocally identify laws and their elements (Lupo, 2002; Lupo and Spinosa, 2001). This choice is compliant with the current (Italian) law editing standards.

5 The VLPM tool

VLPM (Visual Law Process Modeller) is a tool developed in order to support the methodology presented in Section 3 and to facilitate the translation of laws into a process model as well as the maintenance of both when they change.

Figure 9 Law modelling process handled by VLPM



Typical VLPM usage scenarios are depicted in Figure 9. The flow of the modelling activity is the following:

1 A law written in natural language is marked with XML tags. The mark-up can be performed manually or supported by an editor provided by the 'normeinrete' project (e.g., xmLeges (Agnoloni et al., 2007)). The most important tags used by VLPM include legal actors (e.g., subjects, entities, agencies), law structural elements (see 2), and resources (for which the support is limited).

2 The user imports the law formatted in XML, and VLPM generates a skeleton of the model. The user needs to verify and complete the generated model in order to have a reliable *as-is* view of the law. This model can be exported in various formats for documentation purposes (e.g., a website).

3 The user imports an *Explicit Text Amendment* that modifies the law that has been previously modelled with VLPM. The tool highlights the impacts of the amendment on the law and on the model, allowing the user to focus on the affected parts of the model. This greatly simplifies the model revision process.

4 The user modifies the process model, reengineering some processes. At this point documentation can be generated to be shared among the stakeholders and to compare the *as-is* and the *to-be* models. Moreover, VLPM can be used to generate the XML skeleton of a new law that amends the originally modelled law.

At the end of the last step, the XML skeleton of the new law contains all the modifications performed by the process analyst in the process modelling phase. This can be considered as 'what' should be considered by the legislator, but he is then in charge to specify 'how', and refine the final legal version. The advantage for him is that he already has a textual proposal for a new ‘Novella’, which contains all the changes made on the model, and it is already written in the *normeinrete* format.

5.1 Importing a law

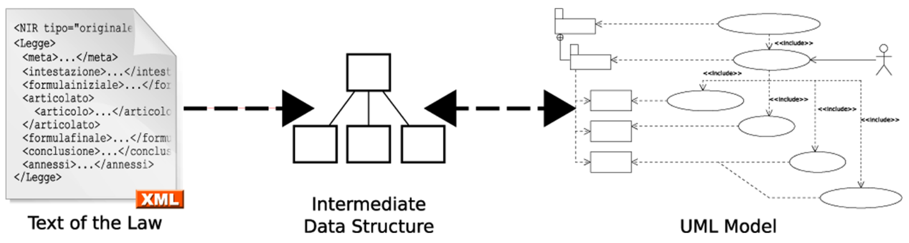
The process of modelling a law with VLPM begins with the generation of a UML model in Visual Paradigm from a law described in XML. Currently we use the Italian ‘Normeinrete’ XML format (Lupo, 2002), but the conversion module could be easily adapted to support formats used in other countries.

Not all the elements of a law can be translated into business processes. Thus, the tool allows the user to choose which law elements will be part of the UML model. The import procedure has been implemented as follows:

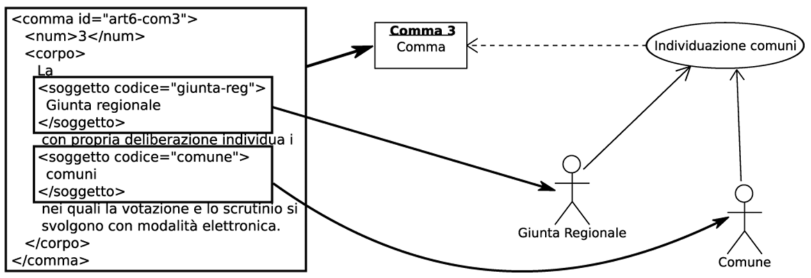
1. The XML file containing the law is parsed and a business process is automatically generated and associated with each element of the law tree. Actors are also identified and associated with their relevant process by looking for XML tags that mark entities.
2. A GUI displays the tree of the law elements, their content, the associated actors and processes. The user selects which actors and which processes to include in the UML model. By choosing a subset of actors and processes, it is possible to study only a specific aspect of the law (e.g., all the operations of which a certain subject is responsible). Moreover, by displaying the content of each law element, the user can decide if it is meaningful to associate a process to that law element.
3. Once the user is satisfied with the model layout, VLPM generates the UML model containing all the elements of the law and a tree of the processes.
   The model thus generated is supported by a serialised data structure that links it to the text of the law in order to maintain traceability (see Figure 10).

An example of a translation of a part of a law to UML is shown in Figure 11. The piece of law is properly tagged to automatically recognise the law element (‘comma’), and two actors (‘subjects’). VLPM identifies the actors involved in a process by searching for tags that identify legal players. VLPM then automatically includes them in the model, together with the law element that provides the documentation for the process. A manual operation is then needed to assign more meaningful names or refine processes in sub-processes (described later).

Figure 10 Storage of law data (see online version for colours)



```
<NIR tipo="originale
<Legge>
  <meta>...</meta>
  <intestazione>...</intest
  <formulainiziale>...</for
  <articolato>
    <articolo>...</articolo
  </articolato>
  <formulafinale>...</formu
  <conclusione>...</conclus
  <annessi>...</annessi>
</Legge>
```



```
<comma id="art6-com3">
  <num>3</num>
  <corpo>
   La
   <soggetto codice="giunta-reg">
    Giunta regionale
   </soggetto>
    con propria deliberazione individua i
   <soggetto codice="comune">
    comuni
   </soggetto>
    nei quali la votazione e lo scrutinio si
    svolgono con modalità elettronica.
  </corpo>
</comma>
```

Figure 11 Translation of a law paragraph to UML

5.2 *Importing an explicit text amendment*

An *Explicit Text Amendment* (in Italian ‘Novella’ or ‘Modifica Testuale Esplicita’, also called MTE) is a new law that modifies the text or the structure of a previous law. A *Novella* introduces or substitutes old law elements. Its XML representation contains special tags which mark modifications.

When importing a ‘Novella’ on a law that is being modelled, the tool looks for XML modification tags, computes which changes occurred, and shows the differences. The user can then review the effects of the ‘Novella’. In the case of a text change she or he can decide how and if the processes, the relationships, and actors should be modified accordingly in the model. In the case of structural changes, the user is provided with the same interface of the law import phase but can intervene only on the part that is being modified.

When a law amendment is promulgated, it exists in parallel with the original law. For our purposes, we need to have a *Coordinated Text* containing the most updated version of the law (Figure 12). Thus, when the change is applied, our data structure is updated so that the text it contains is modified by the amendment. From that moment onwards, the traceability will be maintained with the *Coordinated Text* instead of the text of the original law.

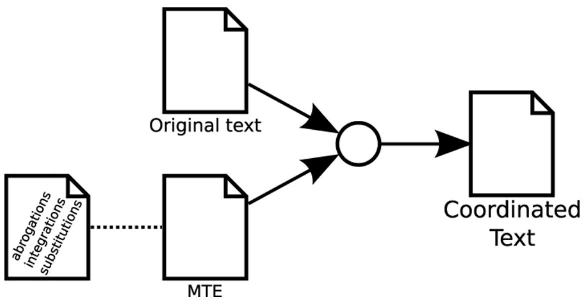


Figure 12 Explicit text amendment

5.3 *Modifying the model*

On top of the build-in features provided by Visual Paradigm, VLPM adds two specific functions to support changes in the models:

- *Model refinement.* Functions, that allow us to maintain traceability with the text of the law. These are meant to be used in order to give a better representation of the procedure defined by the law, and not to add or remove procedures from the actual regulation.
- *Law change suggestions.* Based on the new processes that cannot be traced to any part of the text of the law. When a new process is added to the model, VLPM generates a list of suggestions that can be used to produce an *Explicit Text Amendment* (in 'Normeinrete' XML format) from the changes undergone by the model, thus allowing the law to be realigned to the model.

By providing these two functionalities, VLPM allows the law-model-law *round-trip*, facilitating the law-making process in the case of regulations for PA procedures. The law-model step is obtained by importing a law or a MTE, and by applying model refinement functions. The model-law is a combination of law change suggestions, derived automatically by the tool after the modelling, and manual review by legislators.

5.4 Documenting the model

VLPM is able to generate documentation in various formats, such as PDF or HTML.

This allows an easier sharing of information, involving also stakeholders who do not have access to the modelling tool, facilitating, for example, the publication of information on the internet. Documenting a model and making it publicly available could help citizens without legal writing knowledge to understand complex PA procedures, facilitating a more active participation in lawmaking.

This feature has been successfully used to model traditional voting and e-voting procedures in the Autonomous Province of Trento, Italy and to share information between the technological and administrative stakeholders.

6 Case study: introduction of e-voting

The organisation of elections is a complex PA process: it spans over a time period of months, it involves several actors and various PA offices, and it has a rich set of enforcing rules. Electoral laws precisely define all the steps that have to be performed to run fair elections. Failure to comply with the law may result in litigations in the simplest case and in a threat to citizens' fundamental rights in the worst case.

The methodology described in Section 3 has been applied to model the provincial and regional laws that regulate voting procedures in the autonomous Province of Trento, Italy. It was subsequently used to re-organise voting processes in order to support the ProVotE e-Voting system (Caporusso et al., 2006). VLPM documentation features have been applied to make the model available to all the stakeholders.

VLPM was also applied to rule the introduction of e-voting in a small municipal referendum in Friuli Venezia-Giulia. The referendum involved about 600 people, who used e-voting system with legal value for their first time in Italy.[6]

The law was published on July 27, 2007 and became effective on August 8, 2007 (Venezia-Giulia, 2007). VLPM was used in order to refine an initial draft for this law and to study some possible minor changes related to the usage of electronic systems.

The original law is made of 21 articles, organised in 3 Titles. The first step of our scenario was to add XML tags to the original text, in order to make it compliant with the *complete* 'Normeinrete' DTD. Before importing the original law it was necessary to instruct VLPM to slightly change the default structure automatically proposed:

- to disable law paragraphs which do not define any process
- to associate more processes to the same law paragraph.

As an example, article 17, Comma 2, Letter a defines three atomic and clearly distinguishable procedures: "load data", "data aggregation" and "voters number verification". Using VLPM it is possible to assign to letter a three processes, which represent the three aforementioned procedures.

Title 1, however, contains only general principles and rules regarding referendums that do not define any specific procedure. Thus they are not needed in the processes model and VLPM can be instructed not to generate processes out of this Title.

The law was then imported and used to create a UML process model, following the choices previously described. This model was then used for printing the documentation and facilitating discussion among the stakeholders of the project by providing a visual representation of the processes.

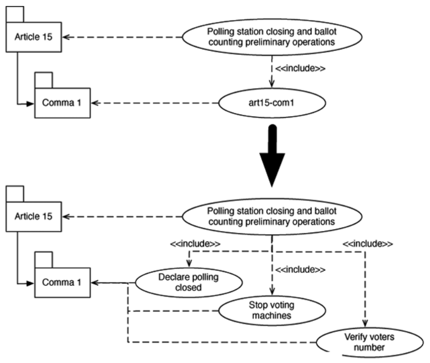
As the election came closer and some constraints became more evident, a 'Novella' was then written, containing some changes for the original law (Figure 13). In summary, the following change types can be recognised:

1 changes to the text of the law which do not influence the structure of the processes (Figure 13(a))

Figure 13 The application of a 'Novella' and its effects: textual changes (on the left), are automatically integrated, and structural changes (on the right), which are detected and supported by the tool, could require a manual refinement: (a) a 'Novella' in its XML representation and (b) the *split* function applied to Art. 15, Comma 1

```
<mod id="mod1">
        In
        <rif xlink:href="2007-09-02;1#art7-com2">
                art. 7, co 2, legge 2 set 2007, n. 1
        </rif>
        , le parole:
        <virgolette id="mod1-vir1">
                nei giorni immediatamente precedenti
        </virgolette>
                sono sostituite dalle seguenti:
        <virgolette id="mod1-vir2">
                il giorno immediatamente precedente
        </virgolette>
</mod>
```

(a)



(b)

2 changes in the relationship between actors and processes

3 Structural changes such as the subdivision of a paragraph in letters, each one defining a particular sub-process.

An example of structural modification is in Article 17, where Comma 4 was introduced. The comma was divided into three letters that specify three atomic sub-procedures of the activities that the Regional Electoral Service must perform at the end of an election. The result of this modification is that the process associated with Comma 4 has now three new sub-processes.

Another example of change to a law is Article 7, Comma 2, where the 'Novella' modified a time constraint. Instead of requiring the installation of the voting machines in polling station some day before the election day, the new text requires them to be installed the day immediately before the election day. Despite this being only a case study, this change is realistic. The reason for this change could be the need for more security in order to avoid leaving the voting machine exposed to tampering after its installation at the polling station.

Model refinement functions can be used to split the leaf processes that do not describe elementary procedures. For example, Article 15, Comma 1 specifies three operations and is by default associated with only one process. It is reasonable to split the process associated with it in three separate processes (see Figure 13(b)). This course of action is more compliant to our methodology and allows a more rigorous analysis of the procedures associated with Comma 1.

Finally, by having a look at Article 11, Comma 3, we discovered that it does not specify in a sufficiently detailed way the suitability check of a voting machine. Since this is a crucial issue for both user confidence and actual security, it is reasonable to envisage a more specific set of operations and thus create new processes that define the sequence of operations required to verify the suitability of a voting machine (or to define more stringent constraints). In this case, VLPM identifies the new processes as 'orphans' and suggests a 'Novella' containing a structural modification in order to include them in the law. Such a template can then be 'translated' in a more specific language.

7 Conclusion and future work

The application of BPR and Software Engineering in law editing can facilitate the work of PAs and favour the involvement of citizens in the law decision process. The definition of strict constraints for the structure of a law (Presidenza del Consiglio dei Ministri, 2001) facilitates its readability and editing, but – in the case of laws defining procedures – the use of visual representations can take this even further.

This paper presented a tool supported methodology to help simplifying BPR and law comprehension activities.

The work we presented in this paper is not the first experience of the application of UML and process reengineering in relation to laws and PA. Various studies have been going on regarding the representation and effective implementation of e-voting procedures with the context of business process. See for example (Xenakis and Macintosh, 2005), in which the authors investigate the need for applying BPR to electoral process. Moreover, Bryl et al. (2007) presented an approach to reason on security properties of the *to-be* models (which are derived from *as-is* models) in order to evaluate procedural alternatives in e-voting systems.

Regarding the analysis of laws, previous work has been mainly focused on the structure of regulation and on the recognition of natural language, as in the works described in Boschi et al. (2005) and van Engers et al. (2004). Our work differs in the sense that it allows us not only to perform an analysis but also a reengineering of PA processes, offering support to law editing and allowing the cooperation of stakeholders belonging to different (not only technological) domains via the exchange of documentation in a standard format and the use of a common methodology. VLPM is not meant to work directly on the text of a law but leaves this task to more specialised people. It focuses instead on the *workflows* of PA procedures and applies a traceability strategy resembling the ones used in the field of *Requirement Management*.

At the present time, VLPM is the merger of two separate tools: one dealing with law modelling and one dealing with model documentation. Future work will comprise a partial re-design of the current architecture with the objective of completing the merger and allowing the inclusion of additional modules to perform a more extensive analysis of processes. In order to achieve this goal we plan to extend the set of supported 'Noreminrete' XML tags, thus increasing the number of types of supported laws.

Integration with the law at the XML level makes our approach applicable also to different other law systems by adapting the law-to-model translation rules. Among the other XML law representation experiences (Marchetti et al., 2002) mention the Uris project by the Swedish Parliament, the Canadian LIMS project and the XML format adopted by the United States of America House of Representatives.[7] Other notable projects – mentioned in de Oliveira Lima et al. (2007) – are the Brazilian LexML, the African Akoma Ntoso and the Dutch Metalex.

Finally, the increasing demand for a reasoning based more on procedural security (and not only on technological security) requires our work to be extended in order to allow a dynamic analysis of 'what if?' scenarios. Our current focus is on the possibility of connecting our work to Weldemariam and Villafiorita (2008), allowing us to perform formal procedural analysis on the processes extracted from laws. This will require further refinement of the formalisation of the current law-to-model translation techniques.

References

Agnoloni, T., Francesconi, E. and Spinosa, P. (2007) 'xmLegesEditor: an OpenSource visual XML editor for supporting legal national standards', *Proc. of V Legislative XML Workshop*, Citeseer, pp.239–252.

Alpar, P. and Olbrich, S. (2005) 'Legal requirements and modelling of processes in e-government', *Electronic Journal of e-Government*, Vol. 3, pp.107–115.

Bolioli, A., Dini, L., Mercatali, P. and Romano, F. (2002) 'For the automated mark-up of Italian legislative texts in XML (www.jurix.nl/pdf/j02-03.pdf)', *Legal Knowledge and Information Systems*. Jurix 2002: *The Fifteenth Annual Conference*. pp.21–30.

Bryl, V., Dalpiaz, F., Ferrario, R., Mattioli, A. and Villafiorita, A. (2007) 'Evaluating procedural alternatives. A case study in e-voting', *Proceedings of 1st International Conference on Methodologies, Technologies and Tools Enabling (METTEG'07)*. An extended version has been published as a Technical Report DIT-07-005, Informatica e Telecomunicazioni, University of Trento, Camerino, Italy.

Boschi, L., Mercatali, P., Romano, F. and Spinicci, E. (2005) 'Automatic translation from textual representation of laws to formal models through uml', *Proceedings of 18th Annual Conference on Legal Knowledge and Information Systems (JURIX 2005)*, Vrije Universiteit Brussel, Auditorium D2.01, pp.71–80.

Caporusso, L., Buzzi, C., Fele, G., Peri, P. and Sartori, F. (2006) 'Transition to electronic voting and citizen participation', in Krimmer, R. (Ed.): *Electronic Voting*, Vol. 86, GI, pp.191–200.

Ciaghi, A., Villafiorita, A. and Mattioli, A. (2009) 'VLPM: a tool to support BPR in public administration', *The Third International Conference on Digital Society (ICDS 2009)*, 1–7 February, Cancun, Mexico, pp.289–293.

de Oliveira Lima, J.A., Palmirani, M. and Vitali, F. (2007) "http' or 'urn' URIs for legal resources? How about both?', *Jurix 2007 Workshop on Legislative XML*, Leiden, http://www.leibnizcenter.org/~winkels/legXMLAbstract.pdf

Hammer, M. and Champy, J. (1993). 'Reengineering the corporation: a manifesto for business revolution', *Business Horizons*, Vol. 36, No. 5, pp.90, 91.

Heib (Autor), R., Kruppke, H. and Scheer, A-W. (2003) *E-Government: Prozessoptimierung in der Öffentlichen Verwaltung*, 1st ed., Springer, 13 June, ISBN-10: 3540034382, ISBN-13: 978-3540034384, 180 pages.

Hlupic, V. (2003) 'Business process modelling using discrete event simulation: potential benefits and obstacles for wider use', *International Journal of Simulation: Systems, Science and Technology*, Vol. 4, pp.62–67.

Katranuschkov, P. (Ed.) (2006) 'Process modelling, process management and collaboration – editorial', *ITcon Vol. 11, Special Issue Process Modelling, Process Management and Collaboration*, July, pp.447, 448, http://www.itcon.org/2006/33

Kettinger, W.J., Teng, J.T.C. and Guha, S. (1997) 'Business process change: a study of methodologies, techniques, and tools', *MIS Quarterly*, Vol. 21, pp.55–80.

Lazzi, G. (2000) *La reingegnerizzazione dei processi Contribution to the book 'Sistemi Informativi per la Pubblica Amministrazione: tecnologie, metodologie, studi di caso'*, Batini, C. and Santucci, G. (Eds.), Agosto 1999, Presidenza del Consiglio dei Ministri, Scuola Superiore di Pubblica Amministrazione, http://www.cnipa.gov.it//site/it-IT/La_Documentazione/In_archivio/Monografie

Lupo, C. (1999) 'Il progetto intersettoriale 'Normeinrete", *Bollettino AIPA Informazioni*, Italian Authority for ICT in Public Administration (AIPA/CNIPA), November–December, pp.11–12, http://www.nir.it/stdoc/nirinbollettinoaipa.doc

Lupo, C. and Spinosa, P. (2001) *Consiglio Nazionale delle Ricerche Istituto per la Documentazione Giuridica ITER LEGIS Informazione e Critica Legislativa*, RISL, Anno I, July–October, http://www.nir.it/stdoc/artcicolournperil4.doc (in Italian).

Lupo, C. (2002) 'Formato per la rappresentazione elettronica dei provvedimenti normativi tramite il linguaggio di marcatura XML', *Allegato tecnico alla Circolare 22 aprile 2002, No. AIPA/CR/40*, Technical Report, http://www.nir.it/stdoc/allegato_aipacr40.doc, http://www.cnipa. gov.it/site/it-IT/Normativa/Circolari_e_Deliberazioni/

Lupo, C. and Batini, C. (2003) 'A federative approach to laws access by citizens: the Normeinrete system', *Lecture Notes in Computer Science*, Springer, pp.413–416.

Marchetti, A., Megale, F., Seta, E. and Vitali, F. (2002) 'Using xml as a means to access legislative documents: Italian and foreign experiences', *SIGAPP Appl. Comput. Rev.*, Vol. 10, No. 1, pp.54–62.

Olbrich, S. and Simon, C. (2008) 'Process modelling towards e-government visualisation and semantic modelling of legal regulations as executable process sets', *The Electronic Journal of e-Government*, Vol. 6, No. 1, pp.43–54.

Presidenza del Consiglio dei Ministri (2001) *CIRCOLARE 2 maggio 2001, n.1, Guida alla redazione dei testi normativi*, (GU n. 101 del 3-5-2001 – Suppl. Ordinario n.105), http://www.diritto.it/rubriche/normativa/circolari/circolare1_2001.html (in Italian).

Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Wohed, P. (2006) 'On the suitability of UML 2.0 activity diagrams for business process modelling', *Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling*, Vol. 53, pp.95–104.

Thaens, M., Bekkers, V. and van Duivenboden, H. (1997) 'Business process redesign and public administration: a perfect match?', in Taylor, J.A., Snellen, I.Th.M. and Zuurmond, A. (Eds.): *Beyond BPR in Public Administration: An Institutional Transformation in an Information Age*, IOS Press, Amsterdam, pp.15–36.

U.S.A. National Performance Review (1993) *Report: Executive Summary – Creating a Government that Works Better and Costs Less* (http://govinfo.library.unt.edu/npr/library/nprrpt/annrpt/redtpe93).

Valiris, G. and Glykas, M. (1999) 'Critical review of existing BPR methodologies: the need for a holistic approach', *Business Process Management Journal*, Vol. 5, pp.65–86, MCB UP Ltd.

van Engers, T.M., van Gog, R. and Sayah, K. (2004). 'A case study on automated norm extraction', in Gordon, T. (Ed.): *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference*, IOS Press, Amsterdam, pp.49–58.

Venezia-Giulia, C.R.F. (2007) *Legge regionale n. 261. Norme sullo svolgimento dei referendum consultivi in materia di circoscrizioni comunali, Voto e scrutinio elettronico*, 27 July, http://www.issirfa.cnr.it/4332,46.html?PHPSESSID=e20d1d2a4ee3bf6ce0005299af25331d, Law (in Italian).

Wastell, D.G., White, P. and Kawalek, P. (1994) 'A methodology for business process redesign: experiences and issues', *Journal of Strategic Information Systems*, Vol. 3, pp.23–40.

Weldemariam, K. and Villafiorita, A. (2008) 'Formal procedural security modeling and analysis', *Risks and Security of Internet and Systems, CRiSIS '08. Third International Conference on*, Tozeur, Tunisia, 28–30 October, pp.249–254.

Willcocks, L., Currie, W. and Jackson, S. (1997) 'Radical reengineering and information systems: evidence from UK public services', *Proceedings of the Fifth European Conference in Information Systems*, Cork Publishing Ltd., Cork (ISBN 1-86076-953-5), pp.460–480.

Xenakis, A. and Macintosh, A. (2005) 'Using business process reengineering (BPR) for the effective administration of electronic voting', *The Electronic Journal of e-Government*, Vol. 3, No. 2, pp.91–98.

Xenakis, A. and Macintosh, A. (2007) 'A methodology for the redesign of the electoral process to an e-electoral process', *Int. J. Electronic Governance*, Vol. 1, p.416.

Notes

[1]See also http://www.normeinrete.it
[2]Available at http://www.giustizia.it/cassazione/leggi/circ1_01.html (in Italian)
[3]Respectively 'Articles', 'Sections' and 'Clauses' in the American legal system.
[4](Laws on the Net) http://www.normeinrete.it
[5]As far as Italian elections are concerned.
[6]This has not been, however, the first election with legal value in Italy using electronic systems, even though little information is available concerning other initiatives in the area.
[7]http://xml.house.gov