Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

ConsolidationViewer: uno strumento web per il consolidamento di documenti legislativi

Relatore:
Chiar.mo Prof.
Fabio Vitali

Correlatrice:
Chiar.ma Prof.ssa
Monica Palmirani

Presentata da:
Giulia Ferrigno

Sessione II
Anno accademico 2023/2024

Abstract

Obiettivo di questa tesi è la descrizione e l'analisi del ConsolidationViewer, uno strumento web per il consolidamento di documenti legislativi che seguono lo standard Akoma Ntoso. Il ConsolidationViewer gestisce e implementa tutte le operazioni per realizzare il consolidamento diretto di un documento e fa parte del web editor Simplex, il quale fornisce supporto al processo di drafting. Ruolo fondamentale gioca lo standard Akoma Ntoso il quale, usando uno specifico vocabolario XML, permette al redattore non solo la scrittura del documento in sé ma anche la gestione dei metadati e il markup delle modifiche, fondamentali al processo di consolidamento. Esso consiste nell'integrazione in un solo documento di un atto originale e di tutte le sue successive modifiche. Il nuovo documento, pur non avendo effetto giuridico, consente di risalire alle norme giuridiche applicabili in un determinato periodo. Il ConsolidationViewer non rappresenta il primo tentativo di realizzare programmaticamente il consolidamento e risente sicuramente del predecessore di Simplex: Lime. Il progetto però si pone come obiettivo il superamento dei limiti della precedente implementazione e l'inserimento di nuove funzionalità, con un occhio di riguardo all'esperienza utente e con il desiderio di rendere l'interfaccia di facile utilizzo anche per i nuovi utenti.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Con l'avvento dell'informatica numerosi mestieri sono stati rinnovati e nuovi posti di lavoro sono stati creati. Il desiderio dell'informatica è sempre stato quello di automatizzare le attività umane al fine di semplificarle. Il processo legislativo, ossia il procedimento che comprende la stesura e revisione delle leggi fino alla loro approvazione, poggia le sue basi nella storia antica e inevitabilmente nei secoli si è evoluta sempre di più adattandosi alle nuove tecnologie, nel senso più ampio del termine. Il primo sviluppo di un sistema digitale in campo legislativo risale agli anni '60, quando vennero introdotti i primi database di documenti legali. Tra la fine degli anni '90 e l'inizio degli anni 2000 il processo di produzione e applicazione delle leggi è stato coinvolto da questa metamorfosi e, con la diffusione di Internet, anche le comunicazioni tra organizzazioni legali si sono digitalizzate [Po24]. Basti pensare al processo tributario in Italia, il quale è totalmente telematico già da ben dieci anni. Geoffrey Bowman, a proposito del processo legislativo, disse: "[...] if five drafters were set on the same Bill, each might emerge with a different product. [...] Now, if five different drafters would produce five different Bills, it suggests that legislative drafting is an art rather than a precise science [Bo06]."
Se il processo legislativo è un'arte antichissima è comunque innegabile la necessità

di omogeneità nel mondo digitale, al fine di mediare tra istituzioni diversissime per organizzazione, lingua e legislazione. In questo contesto si inserisce l'ideazione di Akoma Ntoso, standard internazionale per la codifica e gestione di documenti legislativi. Akoma Ntoso, dotandosi di un proprio vocabolario XML, intende creare un modello comune sia per la strutturazione dei documenti che della gestione e conservazione dei metadati ad essi associati, permettendo la condivisione fra diverse organizzazioni.

La nascita di un nuovo standard ha come naturale proseguimento lo sviluppo di nuovi strumenti per la sua gestione. Nel nostro caso, ciò ha portato allo sviluppo di diversi web editor che supportassero il legislatore nel processo di redazione della legge, rispondendo all'esigenza sia della creazione del nuovo contenuto, ma soprattutto della marcatura di quest'ultimo. In questa tesi ci soffermeremo sul web editor Simplex, teorizzato dal professor Fabio Vitali e dal dott. Andrea D'Arpa e finanziato dall'Università di Bologna [D'A24]; più in particolare la trattazione esaminerà il macrocomponente ConsolidationViewer, da me realizzato.

Il ConsolidationViewer, composto da un componente Angular, un *service* e vari microcomponenti, gestisce il processo di consolidamento all'interno dell'editor Simplex. Il consolidamento consiste nell'integrazione in un solo documento di un atto originale e di tutte le sue successive modifiche e correzioni, permettendo così all'utente di risalire alle norme giuridiche applicabili in un determinato periodo [Eur17].

Il progetto si occupa di implementare il consolidamento diretto, un particolare tipo di consolidamento che prevede la presenza di un documento modificante e un documento modificato; il modificante è il documento contenente le modifiche da applicare al documento modificato, quali inserimenti, sostituzioni, abrogazioni.

La soluzione proposta intende offrire un'esperienza completa e guidata all'utente e, allo stesso tempo, superare le limitazioni e problematiche delle scorse implementazioni; fra queste è impossibile non citare il predecessore di Simplex: Lime. Tuttavia,

l'analisi di Lime ha costituito uno dei pilastri fondamentali di questa ricerca, nei suoi difetti ma anche e soprattutto nei suoi numerosi pregi.

Il ConsolidationViewer integra in sè una nuova attenzione all'esperienza utente: un'interfaccia sobria ed essenziale, una migliore gestione degli errori, purtroppo non rari in documenti di una tale complessità, e nuove feature quali la possibilità di annullare il processo, l'evidenziazione degli elementi coinvolti dalle modifiche nei rispettivi documenti e un algoritmo di ordinamento per le modifiche. Infine, seguirà un'analisi dell'efficacia del nuovo componente realizzato. Il progetto risulta completo nell'intero processo di consolidamento, ma non si esclude che in futuro possano essere gestiti nuovi casi non ancora implementati, soprattutto in base alle esigenze dei singoli utenti adottanti il software.

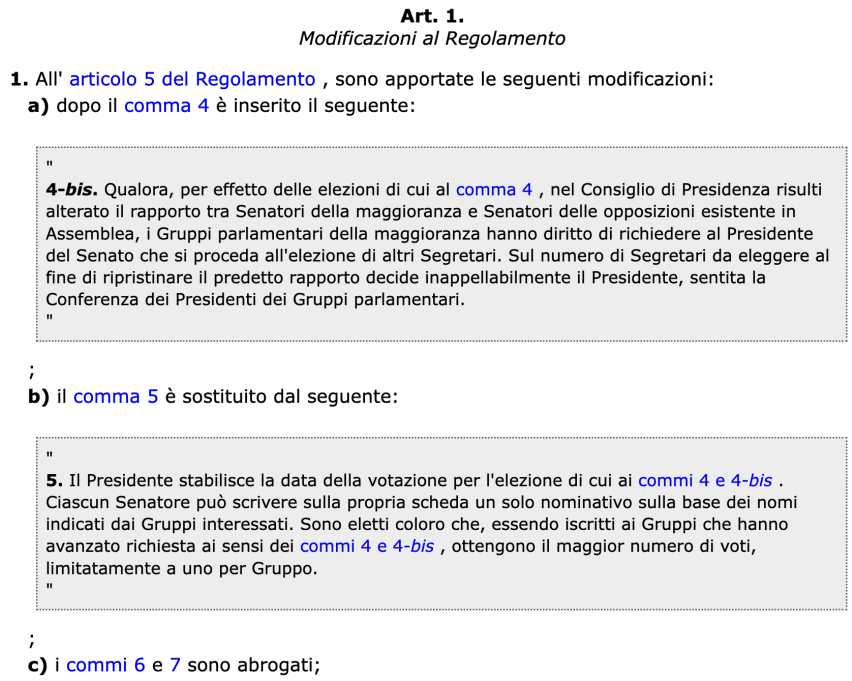
Capitolo 2

Consolidamento di documenti legislativi

Il consolidamento di documenti legislativi consiste nell'integrare in un unico documento di facile lettura un atto originale e tutte le sue successive modifiche e integrazioni. Il processo di consolidamento segue rigorosamente le istruzioni, ossia le disposizioni modificative, indicate nel documento modificante senza alcun cambiamento di sostanza né di forma. Esso costituisce un processo meccanico mediante il quale le disposizioni modificative sono applicate senza che il testo sia alterato, a differenza di codificazioni e rifusioni, le quali quando necessario sopprimono disposizioni obsolete o armonizzano termini e definizioni [CA08]. Tale tecnica si inserisce nel processo di stesura e messa in vigore delle leggi, il processo legislativo. La legislazione è un processo complesso, il cui compito è quello di trasformare le intenzioni astratte dei legislatori in espressioni formali concrete. Di conseguenza, le leggi devono rispettare il delicato equilibrio tra il linguaggio giuridico e un linguaggio chiaro e comprensibile. Il processo legislativo, nella sua profonda complessità, richiede al legislatore una conoscenza approfondita della legge, l'abilità di prevedere potenziali dispute legali future e la capacità di esprimere diritti e doveri in modo che siano liberi da ambiguità e accessibili anche ai non specialisti. È comunque importante

sottolineare che il processo legislativo differisce in base all'Autorità competente.
In Italia, per esempio, il processo richiede che entrambe le camere parlamentari, ossia Camera dei Deputati e Senato, approvino una proposta di legge raggiungendo un accordo su un testo identico e condiviso [CD24]. Le proposte di legge possono essere presentate dal Governo, da singoli parlamentari, da 50.000 elettori, dal Consiglio Nazionale per l'Economia e il Lavoro o dai Consigli Regionali. In seguito, la proposta è assegnata ad una Commissione Permanente, la quale può decidere di unire diverse proposte di legge con argomento affine in un'unica proposta. Successivamente, il testo viene trasmesso da una Camera all'altra fino a quando non viene approvato da entrambe con la medesima formulazione, attraverso il cosiddetto procedimento delle navette. L'ultimo passaggio prima della pubblicazione in Gazzetta Ufficiale consiste nella promulgazione da parte del Presidente della Repubblica, il quale ha il diritto di richiedere eventualmente un riesame.
Invece, il processo legislativo in Europa prevede l'interazione delle tre maggiori istituzioni dell'Unione europea: La Commissione Europea, il Parlamento Europeo e il Consiglio [La23]. Il processo inizia con la Commissione Europea, che propone le nuove leggi. Successivamente, inizia la procedura nota come Procedura Legislativa Ordinaria, che prevede diverse letture per arrivare a conclusione. La prima lettura prevede che il Parlamento adotti il testo nella sua interezza o proponga degli emendamenti. In seguito, il Consiglio decide se concordare con la posizione del Parlamento o se proporre i propri emendamenti. Se non viene raggiunto un accordo neanche alla seconda lettura, interviene un Comitato di Conciliazione, il quale cerca di colmare le differenze giungendo a un testo comune.
Indipendentemente dalla specifica tradizione legislativa di riferimento, il processo di consolidamento risulta fondamentale. Questo processo non solo consente la promulgazione di nuove leggi, attraverso gli emendamenti, ma permette anche di intervenire sulle normative esistenti, modificando, aggiornando o abrogando disposizioni obso-

lete o non più efficaci. Difatti, buona parte dell'attività legislativa consiste nella modifica di precedenti leggi, attraverso l'emanazione di leggi di modifica, le quali però non sono direttamente applicabili dalla cittadinanza. Il processo di consolidamento interviene in questo caso per la creazione della versione finale della legge. Segue un esempio parziale di legge di modifica tratto dal sito del Senato italiano [SR01].



Art. 1.
Modificazioni al Regolamento

1. All' articolo 5 del Regolamento , sono apportate le seguenti modificazioni:
a) dopo il comma 4 è inserito il seguente:

"
4-*bis*. Qualora, per effetto delle elezioni di cui al comma 4 , nel Consiglio di Presidenza risulti alterato il rapporto tra Senatori della maggioranza e Senatori delle opposizioni esistente in Assemblea, i Gruppi parlamentari della maggioranza hanno diritto di richiedere al Presidente del Senato che si proceda all'elezione di altri Segretari. Sul numero di Segretari da eleggere al fine di ripristinare il predetto rapporto decide inappellabilmente il Presidente, sentita la Conferenza dei Presidenti dei Gruppi parlamentari.
"

;
b) il comma 5 è sostituito dal seguente:

"
5. Il Presidente stabilisce la data della votazione per l'elezione di cui ai commi 4 e 4-*bis* . Ciascun Senatore può scrivere sulla propria scheda un solo nominativo sulla base dei nomi indicati dai Gruppi interessati. Sono eletti coloro che, essendo iscritti ai Gruppi che hanno avanzato richiesta ai sensi dei commi 4 e 4-*bis* , ottengono il maggior numero di voti, limitatamente a uno per Gruppo.
"

;
c) i commi 6 e 7 sono abrogati;

Figura 2.1: Esempio di legge di modifica

Il consolidamento rappresenta un'operazione essenziale per garantire coerenza, chiarezza e sistematicità al corpo normativo, riducendo la frammentazione e le sovrapposizioni legislative. Attraverso questo processo, si crea un quadro normativo più facilmente accessibile e comprensibile sia per i cittadini che per gli operatori del diritto, favorendo una maggiore certezza del diritto e migliorando l'efficienza dell'apparato normativo nel rispondere alle esigenze sociali, economiche e istituzionali in continua evoluzione. Nella maggior parte dei casi, il consolidamento normativo

è un processo di natura editoriale, in cui i testi consolidati non vengono sottoposti a verifica o approvazione formale da parte dell'Autorità di riferimento. Tuttavia, questi testi rappresentano comunque un utile strumento per i cittadini, agevolando la consultazione e la comprensione del quadro normativo vigente. Esistono però situazioni, in alcune realtà giuridiche internazionali, in cui il consolidamento assume un carattere autoritativo. In questi contesti, l'Autorità competente non solo supervisiona il processo di consolidamento, ma approva ufficialmente i testi consolidati, attribuendo loro lo stesso valore giuridico dei testi originari. Questo approccio garantisce una maggiore certezza del diritto, eliminando eventuali discrepanze interpretative e conferendo un carattere istituzionale alle versioni consolidate delle norme.

Il processo legislativo australiano segue un percorso simile a quello italiano. Una proposta di legge deve essere approvata da entrambe le camere con stessa formulazione e ricevere il *Royal Assent* dal Governatore Generale [DS24]. In ciascuna camera la proposta subisce tre letture ed è eventualmente emendata. Una volta raggiunto un accordo su un testo identico, il Governatore Generale firma la proposta, che diventa legge. Sebbene il consolidamento sia ampiamente utilizzato anche nelle fasi di ratificazione di una proposta di legge, la particolarità dell'esempio australiano è dato dal consolidamento autoritativo degli *act*. Un *act* è una legge già in vigore che tuttavia può essere generalmente emendata o abrogata da un altro *act* [AG24]. In questo caso, il Parlamento si occupa di approvare anche la cosiddetta *compilation* dell'*act*, ossia una versione della legge che mostri in che modo il testo è stato modificato in un particolare istante di tempo. Nella *compilation* sono integrate informazioni come la data di modifica e quali emendamenti sono stati incorporati. La *compilation*, quindi, non è altro che il testo modificato una volta effettuato il consolidamento.

Nel diritto italiano, un esempio di consolidamento autoritativo è costituito dalla

redazione dei *testi unici*, ossia raccolte di norme che disciplinano una determinata materia [MM96]. I testi unici sono stati creati per diminuire il carattere frammentario, caotico e scoordinato della produzione legislativa italiana, anche con l'intenzione di migliorarne la qualità. I testi unici italiani sono compilativi quando si limitano a riordinare le norme esistenti, oppure sono detti normativi nel momento in cui sostituiscono le fonti originali. L'idea del testo unico è quello di formare un sistema normativo ordinato, passando da una pluralità di fonti ad un'unica fonte. Per quanto i testi unici potrebbero ugualmente lasciare irrisolte eventuali questioni relative alla vigenza di certe norme, rimangono senza dubbio un utile strumento volto alla chiarificazione del quadro normativo vigente. Difatti, essi cercano di superare la complessità normativa in una data materia, causata dallo stratificarsi nel tempo di una pluralità di fonti normative. Ad oggi, i testi unici rappresentano sempre fonti del diritto, indipendentemente dalla misura delle innovazioni rispetto alla legislazione che li precede, poiché la loro emanazione avviene mediante decreto legislativo. Spesso la redazione di testi unici non corrisponde ad una mera riscrittura di testi, in quanto avviene comunque una manipolazione dei testi di partenza. In alcuni casi, la redazione di un testo unico comporta anche l'armonizzazione complessiva delle disposizioni legislative, con la conseguente abrogazione di norme precedenti considerate obsolete o superate. Un esempio emblematico è rappresentato dal "Testo unico in materia di salute e sicurezza nei luoghi di lavoro", emanato con il decreto legislativo del 9 aprile 2008. Questo testo ha riformato, unificato e armonizzato la normativa sulla sicurezza, riordinando disposizioni accumulate nell'arco di oltre sessant'anni.

Un ultimo esempio fondamentale è rappresentato dallo *United States Code*, la raccolta e codificazione ufficiale delle leggi federali degli Stati Uniti d'America [OLRC24]. Lo *United States Code* è suddiviso in titoli per argomento: alcuni sono definiti titoli di diritto positivo, mentre altri sono titoli di diritto non-positivo. I primi rap-

presentano le leggi federali vere e proprie, poiché sono approvate direttamente dal Congresso. I secondi, invece, sono compilazioni editoriali di leggi federali, tuttavia non approvati dal Congresso come titoli unici. È bene notare le differenze legali fra i due tipi di titoli: i titoli di diritto positivo sono considerati prova legale del contenuto delle leggi, per cui non è necessario fare riferimento ad altre fonti; i titoli di diritto non-positivo costituiscono una prova preliminare del contenuto delle leggi che includono, di conseguenza possono essere contestati nel momento in cui si dimostra che il testo della legge originale differisce da quello del titolo. La codificazione del diritto positivo è eseguita dall'*Office of the Law Revision Counsel*, il quale si occupa di ricodificare le norme esistenti in modo che diventino titoli di diritto positivo. La codificazione rende lo *United States Code* più utile e accessibile, migliorandone l'organizzazione, eliminando le disposizioni obsolete e risolvendo eventuali conflitti normativi o ambiguità. Il processo di codificazione del diritto positivo inizia con l'*Office of the Law Revision Counsel* che analizza la normativa esistente con l'aiuto di comitati del Congresso ed esperti legali e redige un disegno di legge, in cui è spiegato dettagliatamente come le sezioni originali sono state trasformate. Il disegno di legge è poi sottoposto al *Committee on the Judiciary* della Camera dei rappresentanti. Una volta approvato, il titolo diventa legge ed è incluso nello *United States Code* come titolo di diritto positivo.

Il tema del consolidamento rappresenta una questione concreta e rilevante, nonché ampiamente diffusa, che trascende le differenze tra le varie tradizioni giuridiche. Si tratta di un problema che interessa ordinamenti di ogni tipo, indipendentemente dal sistema legislativo o dall'approccio normativo adottato. La difficoltà nel garantire una coerenza legislativa, unificare testi normativi frammentati e aggiornare regolarmente le disposizioni per eliminare incongruenze e obsolescenze è una sfida universale, che accomuna i legislatori di tutto il mondo. In questo contesto, è fondamentale l'adozione della tecnologia, che progressivamente contribuisce a rendere il

processo legislativo più snello ed efficiente. Nel corso degli anni, la legislazione è passata dall'essere un'attività puramente manuale all'integrazione crescente di software avanzati, progettati per supportare e ottimizzare le varie fasi del processo legislativo. Questi strumenti possono offrire un supporto concreto ai legislatori, semplificando il monitoraggio, l'aggiornamento e l'integrazione delle normative. Inoltre, contribuiscono a ridurre il rischio di errori e incongruenze all'interno dei testi legislativi, garantendo maggiore coerenza e affidabilità al quadro normativo. L'introduzione di tali strumenti non solo accelera i processi di consolidamento, ma migliora anche la qualità delle leggi, facilitando il loro utilizzo da parte dei cittadini e degli operatori del diritto.

2.1 Tipi di consolidamento

Il consolidamento non è un processo unico riconosciuto universalmente, ma ogni giurisdizione lo applica ed esegue concordemente con il proprio processo legislativo e peculiarità culturali. È possibile riconoscere diversi tipi di consolidamento in base ai documenti prodotti dal sistema legislativo nel momento in cui bisogna modificare una legge, possiamo quindi distinguere [Pa23]:

- il consolidamento diretto: prevede la presenza di un documento modificante, il quale riporta le disposizioni modificative, e un documento modificato, che costituisce la versione precedente, su cui applicare le modifiche.
  Questa tecnica è utilizzata per il consolidamento degli atti della Corte di Cassazione in Italia.

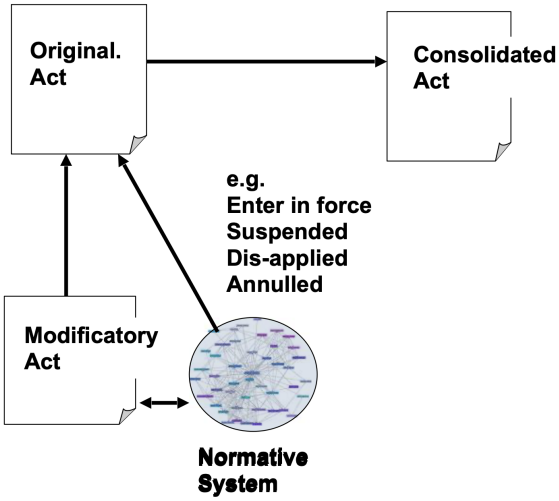


Figura 2.2: Schema relazionale dei documenti nel consolidamento diretto

- il consolidamento inverso: il documento originale viene annotato con le modifiche decise dal sistema normativo, il documento modificante viene creato successivamente.

  Questo metodo è utilizzato dal Parlamento Europeo tramite emendamenti.

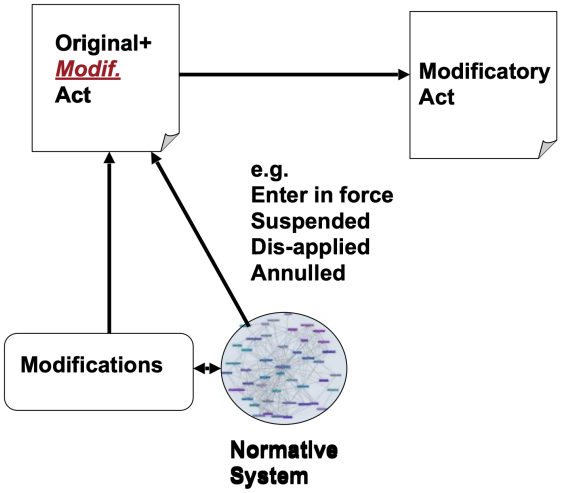


Figura 2.3: Schema relazionale dei documenti nel consolidamento inverso

- il consolidamento per differenze: sono prese in considerazione la versione precedente e successiva dello stesso documento per dedurre le modifiche osservandone le differenze.
  Tale tecnica è utilizzata dalla Gazzetta ufficiale canadese.

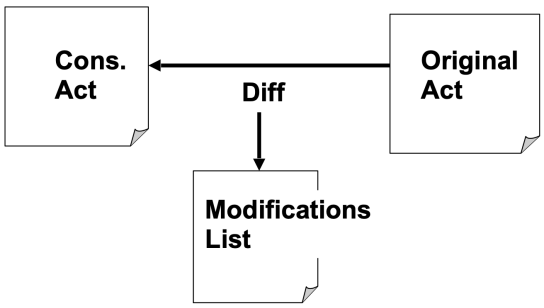


Figura 2.4: Schema relazionale dei documenti nel consolidamento per differenze

Ogni tecnica porta con sé i propri benefici e svantaggi: il consolidamento diretto permette una migliore gestione del modello temporale, mentre il consolidamento per differenze è efficace per le modifiche testuali, ma non particolarmente per modifiche non testuali. Il consolidamento può anche differenziarsi per il metodo utilizzato, ossia il modo in cui le disposizioni modificative sono formulate o presentate:

- metodo deliberativo: prevede la presenza di un documento modificante il quale riporta le disposizioni modificative che avranno effetto su un altro documento. È utilizzato dal Senato italiano;

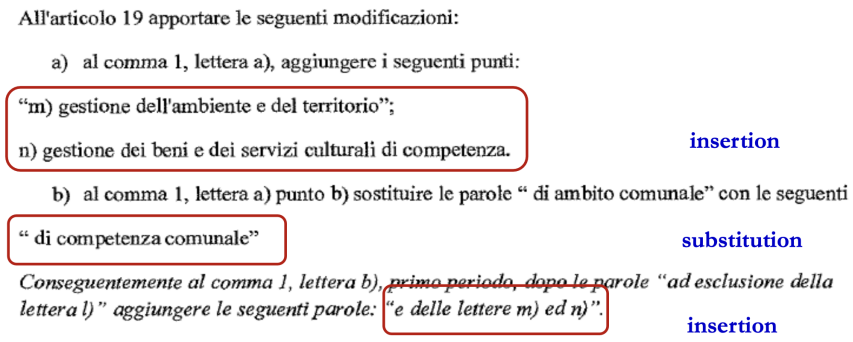


Figura 2.5: Esempio di documento modificante nel metodo deliberativo

- metodo tabellare: nel documento affetto dalle modifiche, in corrispondenza delle parti testuali modificate, sono posizionati a fianco il testo originale e il testo modificato proposto dall'emendamento.
  È utilizzato dal Parlamento europeo;

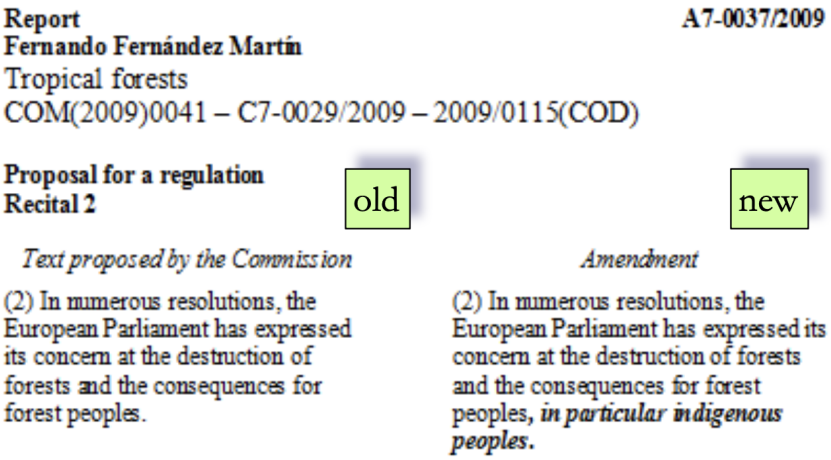


Figura 2.6: Esempio di documento modificato nel metodo tabellare

- metodo redlining: il documento consolidato presenta il testo rimosso barrato, da qui il nome, e il testo inserito evidenziato.
  Questo metodo è utilizzato in Sud Africa e nei paesi anglofoni;

BE IT ENACTED by the Parliament of the Republic of South Africa, as follows:—

Amendment of Preamble to Act 13 of 2002, as substituted by section 1 of Act 19 of 2004

1. The Preamble to the Immigration Act, 2002 (Act No. 13 of 2002) (hereinafter referred to as the principal Act), is hereby amended by the substitution for paragraph *(c)* of the following paragraph:

"*(c)* interdepartmental coordination and public consultations enrich the **[functions]** management of immigration **[control]**;".

Figura 2.7: Esempio di documento modificato nel metodo redlining

- metodo a elenco: consiste nell'elencare le disposizioni modificative in una lista, il tipo di modifica si deduce dalla comparazione con il documento modificato. È utilizzato in Camerun.

destination — *Art. 80 cpv. 1 e 2, frase introduttiva e n. 1bis, 2 e 3*

substitution — [1] Se il credito è fondato su una decisione giudiziaria esecutiva, il creditore può chiedere in giudizio il rigetto definitivo dell'opposizione.

insertion — [2] Sono parificati alle decisioni giudiziarie:

1bis. i documenti pubblici esecutivi secondo gli articoli 347–352 CPC[119];

2. le decisioni di autorità amministrative svizzere;

repeal — 3. *abrogato*

Figura 2.8: Esempio di documento modificante nel metodo a elenco

Da qui in avanti, nella presentazione del progetto realizzato, quando si parlerà di consolidamento mi riferirò a un processo di consolidamento diretto con metodo deliberativo.

Capitolo 3

Formati ed editor per il processo legislativo

Nel corso del tempo, il problema del consolidamento è stato affrontato attraverso diverse soluzioni, soprattutto per quanto riguarda la sua codifica. Uno degli approcci più efficaci prevede di esprimere i documenti legislativi in formato Akoma Ntoso, uno standard XML progettato appositamente per documenti del mondo giuridico. Esso consente di rappresentare tutte le peculiarità tipiche di tali documenti e, grazie al suo ricco vocabolario, cerca di catturare la complessità sia dei testi normativi sia delle organizzazioni giuridiche da cui derivano, incluse le varie tradizioni legali. Una delle caratteristiche distintive di Akoma Ntoso è l'introduzione di tag specifici per le disposizioni modificative, rendendolo uno strumento prezioso per affrontare il consolidamento in modo strutturato ed efficace. Tra gli strumenti che supportano la creazione e la gestione di documenti in formato Akoma Ntoso, si distingue Lime, un web editor che ha cercato di implementare funzionalità avanzate per il consolidamento e che rappresenta il punto di partenza per lo sviluppo del ConsolidationViewer.

3.1 Akoma Ntoso

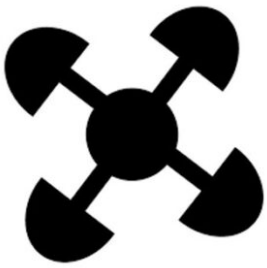


Figura 3.1: Simbolo akoma ntoso

Akoma Ntoso (in italiano "cuori uniti") è uno standard internazionale per la rappresentazione di documenti legislativi, giudiziari e parlamentari. Esso permette lo scambio di documenti tra diverse istituzioni nel mondo e la creazione di un modello comune per dati e metadati, promuovendo così la creazione di strumenti che possano essere condivisi ed estesi da tutti i partecipanti. Lo standard ha come obiettivo la definizione di un formato che permetta la conservazione a lungo termine dei documenti, permettendone l'accesso, la ricerca e la visualizzazione [VP18]. Akoma Ntoso ha come obiettivo, inoltre, rendere la struttura e il significato dei documenti legali "*machine readable*", in modo che un computer possa analizzare le informazioni al loro interno e usare processi simili al ragionamento deduttivo. Questo si ottiene attraverso il cosiddetto "*markup*", ossia l'atto di aggiungere annotazioni sulle diverse parti del documento. Nel caso di Akoma Ntoso, queste annotazioni consistono in tag XML. È necessario quindi che vengano identificati con precisione i confini di ogni segmento di testo, dando un corretto identificatore che rappresenti al meglio il suo ruolo strutturale o semantico. I segmenti infatti devono possedere un *name* a partire da una vasta scelta predefinita. Questi includono concetti come preamboli, sezioni, paragrafi, affermazioni e riferimenti.
Altri obiettivi di Akoma Ntoso sono:

- utilizzare meccanismi comuni per il collegamento delle risorse, ovvero gli URI;
- essere auto-esplicativo, ovvero essere in grado di fornire informazioni attraverso una semplice analisi dello schema;
- essere estendibile, in modo che sia possibile attuare delle customizzazioni locali senza sacrificare l'interoperabilità con altri sistemi.

Infine, lo standard cerca di attuare un'esplicita separazione tra gli autori, i quali sono responsabili del contenuto dei documenti, e gli editor, coloro che decidono il layout finale e si occupano della pubblicazione. I principali blocchi di un documento Akoma Ntoso sono il blocco <meta> per la gestione dei metadati e il blocco <body> per il contenuto in sé del documento.

Il blocco <meta> in un documento Akoma Ntoso ha il compito di contenere tutti i metadati del documento legislativo. I metadati, per definizione, sono di natura editoriale, ossia non sono contenuto ma affermazioni riguardo il contenuto. I metadati sono fondamentali per migliorare la comprensione e la classificazione del documento, soprattutto in riferimento al desiderio di rendere i documenti "*machine readable*" [VP18]. I metadati quindi sono un insieme di informazioni, quali data di pubblicazione, autore, lingua, formato e molto altro. La sezione ad essi dedicati si articola in diverse sottosezioni, fra le principali è possibile trovare:

- *identification*: contiene tutte le informazioni relative a date e autori;
- *publication*: informazioni riguardo la pubblicazione;
- *lifecycle*: include la lista degli eventi che modificano il documento;
- *analysis*: elenco delle qualificazioni delle disposizioni e delle asserzioni contenute nel testo;

- *references*: lista delle risorse esterne connesse al documento, quali individui, organizzazioni e concetti rilevanti per comprendere il contenuto e la storia del documento.

```
<body>
    <article eId="art_8">
        <num eId="art_8__num">
            Article 8
        </num>
        <heading eId="art_8__heading">
            Conditions applicable to child's consent in relation to information society service
        </heading>
        <paragraph eId="art_8__paragraph_1">
            <num eId="art_8__paragraph_1__num">
                1.
            </num>
            <content eId="art_8__para_1__content">
                <p eId="art_8__paragraph_1__p_1">
                    Where point (a) of Article 6(1) applies, in relation to the offer of
                    <term refersTo="/akn/ontology/concept/noConcept" status="incomplete" eId="art_8__paragraph_1__term_1">
                        information society services
                    </term>
                    directly to a
                    <term refersTo="/akn/ontology/concept/noConcept" status="incomplete" eId="art_8__paragraph_1__term_2">
                        child
                    </term>
                    , the
                    <term refersTo="/akn/ontology/concept/noConcept" status="incomplete" eId="art_8__paragraph_1__term_3">
                        processing of the personal data
                    </term>
                    of a child shall be lawful where the child is
                    <term refersTo="/akn/ontology/concept/noConcept" status="incomplete" eId="art_8__paragraph_1__term_4">
                        at least 16 years old
                    </term>
                    . Where the child is below the age of 16 years, such processing shall be lawful only if and to the extent that consent is given or authorised by the holder of parental
responsibility over the child.
                </p>
            </content>
        </paragraph>
    </article>
</body>
```

Figura 3.2: Esempio di blocco body

Il body di un documento Akoma Ntoso è strutturato in modo da rappresentare la struttura gerarchica, spesso complessa, dei documenti legali. Sono presenti elementi gerarchici, come capitoli, sezioni e articoli, i quali facilitano la navigazione all'interno del documento [D'A24]. All'interno delle strutture gerarchiche sono presenti elementi di contenuto, quali paragrafi, sottoparagrafi e punti, nei quali si inseriscono le disposizioni. È possibile inoltre includere altri elementi come annotazioni e note a piè di pagina oppure riferimenti ad altri documenti rilevanti.

Il blocco body può presentare inoltre tag specifici per indicare le modifiche all'interno del documento stesso. Due esempi sono i tag <ins> e <del>, i quali sono elementi *inline* per specificare inserimenti ed eliminazioni di tipo editorale; questi possono essere usati nel momento in cui si applicano le modifiche durante il processo di consolidamento [VP18].

Altro tag specifico è l'<omissis>, il quale è un elemento *inline* che contiene una

porzione di testo che sostituisce una parte omessa [VP18]; questo può essere utilizzato per riportare in un testo l'espressione: "L'articolo num. è stato abrogato." nel momento in cui si applica una modifica abrogativa.

La trattazione si soffermerà sulle sezioni fondamentali alla realizzazione del processo di consolidamento del blocco <meta>, ovvero *identification* e *analysis*.

3.1.1 <identification>

```
<identification source="#source">
    <FRBRWork>
        <FRBRthis value="/akn/al/act/ligj/2012-07-19/74/!main"/>
        <FRBRuri value="/akn/al/act/ligj/2012-07-19/74"/>
        <FRBRdate date="2012-07-19" name=""/>
        <FRBRauthor href="#" as="#"/>
        <FRBRcountry value="al"/>
        <FRBRsubtype value="ligj"/>
        <FRBRnumber value="74"/>
        <FRBRprescriptive value="false"/>
        <FRBRauthoritative value="false"/>
    </FRBRWork>
    <FRBRExpression>
        <FRBRthis value="/akn/al/act/ligj/2012-07-19/74/alb@/!main"/>
        <FRBRuri value="/akn/al/act/ligj/2012-07-19/74/alb@"/>
        <FRBRdate date="2012-07-19" name=""/>
        <FRBRauthor href="#" as="#"/>
        <FRBRlanguage language="alb"/>
    </FRBRExpression>
    <FRBRManifestation>
        <FRBRthis value="/akn/al/act/ligj/2012-07-19/74/alb@/main.xml"/>
        <FRBRuri value="/akn/al/act/ligj/2012-07-19/74/alb@.xml"/>
        <FRBRdate date="2024-10-10" name=""/>
        <FRBRauthor href="#" as="#"/>
        <FRBRformat value="xml"/>
    </FRBRManifestation>
</identification>
```

Figura 3.3: Esempio di blocco identification

Akoma Ntoso utilizza il modello FRBR (*Functional Requirements for Bibliographic Records*), il quale attravero il modello relazionale intende dare una rappresentazione formale alle informazioni bibliografiche [VP18]. Tali informazioni, in un documento Akoma Ntoso, sono inserite nel blocco <identification>, il quale possiede una sezione per ogni entità fondamentale:

- *Work*: rappresenta il documento come entità unica e originaria.
  Esempio: L'opera teatrale "Amleto" di William Shakespeare;

- *Expression*: indica la versione del documento il cui contenuto è diverso dalle altre versione per vari criteri, come per esempio la lingua.
  Esempio: Il primo quarto dell'opera teatrale "Amleto" (1601);

- *Manifestation*: rappresenta il modo in cui la versione si "manifesta", quindi qualsiasi formato fisico o elettronico dell'*Expression*.
  Esempio: Una delle versioni stampate del primo quarto dell'opera teatrale "Amleto";

- *Item*: rappresenta un singolo esemplare di una manifestazione.
  Esempio: La mia copia di una delle versioni stampate del primo quarto dell'opera teatrale "Amleto".

Gli *Item*, pur possedendo una propria sezione, al momento non sono utilizzati da nessuna giurisdizione che adotta Akoma Ntoso.
La gerarchia FRBR è fondamentale in quanto uno stesso documento legale può assumere diverso contenuto dopo essere stato revisionato o emendato nel corso del suo ciclo vita. Il documento quindi fa riferimento a un singolo *work*, dal quale si generano diverse *expression* ogni volta che il contenuto subisce modifiche, per esempio attraverso un emendamento. Ognuna di queste *expression* ha la possibilità di essere espressa in un qualche formato elettronico, come un PDF, un documento HTML o, nel nostro caso, uno specifico documento XML. Ognuno di queste genera almeno una *manifestation* [VP18].
Per favorire la permanenza e la condivisibilità dei documenti, evitando così di essere vincolati da una specifica architettura o filesystem, i documenti Akoma Ntoso sono dotati di particolari URI, i quali rispettano la gerarchia FRBR. La *naming convention* per gli URI è pensata in modo che ogni *manifestation* del documento abbia

un proprio identificatore unico, al fine di poter gestire al meglio il versionamento, soprattutto in casi come quello affrontato da questa tesi, in cui è necessario tenere traccia del fattore temporale e della successione di modifiche e revisioni. Gli URI sono composti da tre parti, da sinistra verso destra:

- la prima relativa al *Work*, contenente il prefisso "/akn", la nazione, il tipo di documento, opzionalmente un sottotipo e l'autore, la data di creazione e un numero o un titolo per eventualmente disambiguare documenti con stesse informazioni;
- la seconda relativa all'*Expression*, contenente la lingua della versione corrente e, opzionalmente, una data, la quale solitamente indica l'entrata in vigore del nuovo contenuto, l'autore della versione o qualsiasi altro elemento che differenzi una versione dall'altra;
- la terza e ultima relativa alla *Manifestation*, in cui opzionalmente è possibile inserire un autore o una data per segnalare un cambiamento significativo di markup e infine il formato in cui si presenta il documento (e.g. .xml, .pdf, .docx).

Gli URI sono stati realizzati con l'obiettivo di rispettare il modello relazionale fra *Work*, *Expression* e *Manifestation*, facilitando così la classificazione e il recupero dalle banche dati e favorendo anche la possibilità di filtrare in maniera efficiente durante le ricerche [D'A24]. Tenendo quindi presente la *naming convention* per i documenti Akoma Ntoso è possibile ottenere informazioni fondamentali su un documento in seguito a una semplice lettura del suo URI. Prendendo in considerazione i metadati mostrati nella Figura 3.3, possiamo dedurre che il documento in questione sia un atto albanese, più precisamente la legge 74 del 2012. La versione selezionata è in lingua albanese e il formato di tale versione è XML.

3.1.2 <analysis>

```
<analysis source="#source">
    <parliamentary/>
    <activeModifications>
        <textualMod eId="pmod_1" type="insertion">
            <source href="#art_1__para_1__mod_1"/>
            <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_2__para_15"/>
            <new href="#art_1__para_1__mod_1__qstr_1"/>
        </textualMod>
        <textualMod eId="pmod_2" type="repeal">
            <source href="#art_1__para_2__mod_2"/>
            <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_2__para_21"/>
        </textualMod>
        <textualMod eId="pmod_3" type="substitution">
            <source href="#art_2__para_1__mod_3"/>
            <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_6__para_1"/>
            <new href="#art_2__para_1__mod_3__qstr_1"/>
        </textualMod>
        <textualMod eId="pmod_4" type="repeal">
            <source href="#art_2__para_2__mod_4"/>
            <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_6__para_3"/>
        </textualMod>
        <textualMod eId="pmod_5" type="substitution">
            <source href="#art_2__para_3__mod_5"/>
            <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_6__para_4"/>
            <new href="#art_2__para_3__mod_5__qstr_1"/>
        </textualMod>
```

Figura 3.4: Esempio di blocco analysis

Il blocco <analysis> include tutti i metadati giuridici che provengono da una specifica interpretazione della sorgente legale. Esso contiene le informazioni riguardanti le modifiche e restrizioni degli effetti normativi come, per esempio, per via di limitazioni di giurisdizione o risultati di sentenze. Sono di seguito elencate alcune delle sue principali sottosezioni:

- *activeModifications*: per la gestione delle modifiche descritte nel documento corrente, con effetto su un altro documento;
- *passiveModifications*: per la gestione delle modifiche che hanno effetto sul documento corrente, provenienti da un altro documento;
- *restrictions*: per la gestione delle limitazioni dell'effetto normativo;
- *judicial*: contentente informazioni riguardanti metadati giudiziari, come il risultato di una sentenza.

Le modifiche attive

In tutti i tipi di documento, Akoma Ntoso permette di modellare le disposizioni modificative, ossia affermazioni che modificano in una qualche misura un altro documento. Gli elementi testuali sono modellati attraverso le *quotedText*, contenenti semplici stringhe, e le *quotedStructure*, le quali contengono intere strutture, quali per esempio un paragrafo o un intero articolo con i suoi paragrafi. Il blocco *activeModifications* contiene una serie di informazioni semantiche come il tipo di modifica (se testuale, di efficacia, di significato, ecc...), l'origine della modifica, il punto di destinazione della modifica, parametri temporali, limitazioni della modifica. Come accennato, le modifiche possono essere di diversi tipi: *efficacyMod*, *forceMod*, *legalSystemMod*, *meaningMod*, *scopeMod* e *textualMod* [VP18].
Segue adesso un approfondimento sulle *textualMod*, le quali sono le modifiche coinvolte dal processo di consolidamento.

textualMod Le *textualMod* modellano le modifiche testuali ai documenti, ossia modifiche al contenuto o alla struttura del documento. Le *textualMod* presenti in *Akoma Ntoso* sono le seguenti:

- *insertion*: aggiunzione di nuovo testo o strutture testuali, come punti, paragrafi, articoli;
- *repeal*: abrogazione di articoli, paragrafi o eliminazione di frasi, parole;
- *substitution*: sostituzione di testo semplice o strutture testuali;
- *join*: unione di due strutture allo stesso livello gerarchico in una sola;
- *split*: divisione di una struttura in due strutture distinte;
- *renumbering*: rinumerazione di una struttura avente un numero, per esempio un articolo.

Ogni *textualMod* contiene delle informazioni per la corretta codifica ed eventuale esecuzione della modifica sul documento target. La *source* contiene un riferimento al frammento o porzione di testo dove la modifica è espressa. La *destination* contiene l'IRI del frammento o porzione di testo su cui la modifica deve essere applicata. L'*old* contiene il riferimento alla *quotedText* o *quotedStructure* dove trovare il testo originale che deve essere modificato. Il *new* contiene invece il riferimento alla *quotedText* o *quotedStructure* con il nuovo testo da inserire o sostituire [VP18].

```
<textualMod eId="pmod_1" type="insertion">
    <source href="#art_1__para_1__mod_1"/>
    <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_2__para_15"/>
    <new href="#art_1__para_1__mod_1__qstr_1"/>
</textualMod>
```

Figura 3.5: Esempio di *textualMod*

In Figura 3.5, è possibile vedere il blocco di metadati relativo a una *textualMod* proveniente da un documento reale. La modifica è di tipo *insertion*. La disposizione modificativa di tale inserimento si trova nel paragrafo 1 dell'articolo 1, come è possibile dedurre dalla *source*. L'inserimento dovrà avvenire nel paragrafo 15 dell'articolo 2 del documento 10019 del 2008, come identificato dall'URI del tag *destination*. Infine, l'elemento *new* identifica la locazione della *quotedStructure* che deve essere inserita; in questo caso si trova nel paragrafo 1 dell'articolo 1 del documento corrente.

Le modifiche passive

Akoma Ntoso permette non solo la modellazione delle modifiche attive, ma anche l'espressione delle modifiche passive, ovvero le modifiche che hanno effetto sul documento corrente provenienti da un altro documento. A ogni modifica specificata nelle *activeModifications* del documento modificante, corrisponderà una simmetrica modifica nelle *passiveModifications* del documento modificato. *Source* e *destination*

si riferiranno alle stesse porzioni di testo nei due documenti. In questo modo, dalla lettura di un documento modificato a cui sono state applicate delle modifiche è possibile risalire sia alle porzioni di testo modificate, sia al documento che ha disposto tali modifiche, permettendo di tenere traccia meticolosamente del processo di versionamento. Riprendendo l'esempio di modifica testuale mostrata nella Figura 3.5, si mostra adesso la corrispondente *textualMod* nel documento modificato.

```
<textualMod eId="pmod_1" type="insertion">
    <source href="/akn/al/act/ligj/2012-07-19/74/!main/#art_1__para_1__mod_1"/>
    <destination href="#art_2__para_15"/>
    <new href="#art_2__para_15-1"/>
</textualMod>
```

Figura 3.6: Esempio di *textualMod* in *passiveModifications*

La *source* non rappresenta più un riferimento all'interno del documento stesso, ma si riferisce a un documento esterno, ovvero al documento modificante. Allo stesso modo, la *destination* ora corrisponde a un riferimento a una sezione del proprio documento. Il riferimento in *new*, invece, continua a indicare il testo nuovo, ma questa volta all'interno del documento modificato.

3.2 Strumenti precedenti: Lime

Per il design e l'implementazione del ConsolidationViewer è stato necessario analizzare precedenti strumenti che realizzano il consolidamento; in particolare lo studio si è concentrato su Lime, web editor per il markup di documenti Akoma Ntoso sviluppato presso il CIRSFID e l'Università di Bologna.
Lime è risultato la scelta naturale per un'analisi preliminare in quanto Simplex, editor nel quale si inserisce il ConsolidationViewer, ha come principale obiettivo il superamento dei suoi limiti. Uno di questi è la creazione di diverse versioni in base al cliente, in quanto se da un lato è stato così possibile fornire delle versioni altamente personalizzate e aderenti alle esigenze dei clienti, dall'altro ha portato a una

difficile manutenibilità, essendo più complicato debellare i bug fra le varie versioni. La trattazione si concentrerà su Lime Albania, essendo una delle principali versioni per cui è stato realizzato il consolidamento.

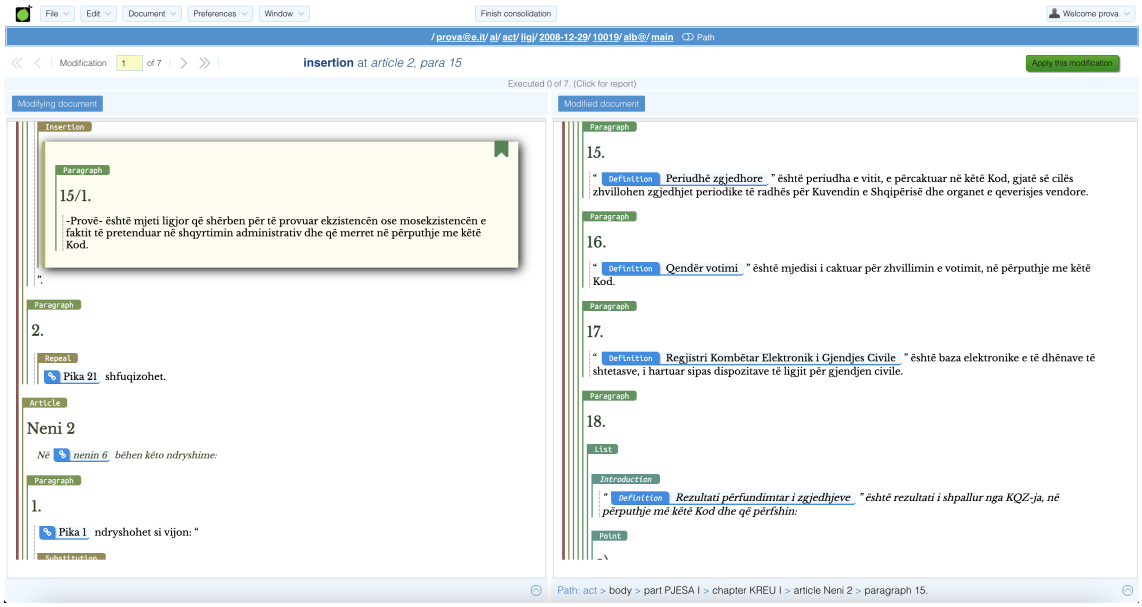


Figura 3.7: Interfaccia di Lime Albania

Dopo la selezione dei documenti su cui eseguire il processo di consolidamento, l'utente può visualizzare ad una ad una le diverse modifiche e decidere se applicarle o meno tramite apposito bottone. È mostrata in alto una descrizione della modifica corrente ed è possibile visionare il report delle modifiche già eseguite. L'architettura si basa principalmente su due *controller*:

- *ConsolidationController*: implementa le funzioni del consolidamento, gestisce l'interfaccia e la scelta dei documenti;
- *ModificationController*: implementa le funzioni di modifica, la ricerca delle destinazioni e l'esecuzione vera e propria.

Per quanto funzionalmente completa, l'interfaccia di Lime risulta datata, con un *look and feel* non più in linea con i software moderni. I glitch e i bug non sono rari, un esempio è l'evidenziazione delle origini e destinazioni: implementate dal punto di vista del codice ma spesso non visibili. Le modifiche inoltre non sono presentate all'utente in ordine di comparsa nel documento modificante, ordine solitamente usato dai giuristi per applicarle. La mancanza di pulsanti per il reset dei documenti o l'annullamento del processo rendono inoltre l'esperienza utente ancora più macchinosa. Alla luce delle carenze del software Lime, si è deciso di sviluppare un nuovo strumento, il ConsolidationViewer, concepito per realizzare il consolidamento e superare i limiti riscontrati.

Capitolo 4

ConsolidationViewer

4.1 Funzionamento del ConsolidationViewer

Il ConsolidationViewer realizza, all'interno del web editor Simplex, il processo di consolidamento deliberativo diretto. L'utente ha a disposizione un'interfaccia attraverso la quale potrà visualizzare e scegliere i documenti dal database *eXist*. Sono mostrati a schermo gli URI dei documenti selezionati, come è possibile vedere nella seguente figura.

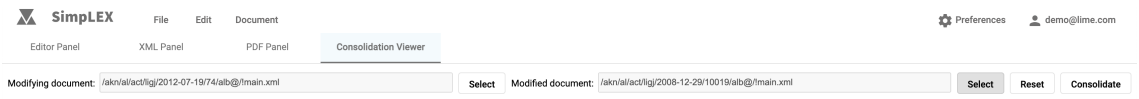


Figura 4.1: Interfaccia dopo la selezione di due documenti

L'utente in questa fase può decidere di ripristinare l'interfaccia e procedere nuovamente con la selezione in caso di errori. Una volta scelti entrambi i documenti, l'utente inizia il consolidamento attraverso l'apposito pulsante. Il sistema, prima di inizializzare gli editor, procede a fare dei controlli sui documenti:

1. controlla se effettivamente entrambi i documenti sono stati selezionati ed eventualmente avvisa l'utente tramite *dialog*;

2. verifica che il documento modificante abbia data più recente del documento modificato: per definizione stessa le modifiche di un documento non possono essere state emendate prima del documento su cui si applicano. Nel caso in cui il controllo fallisca si propone all'utente di fare lo scambio dei documenti. Questa feature è stata realizzata in seguito a una diretta richiesta dei committenti;

3. si controlla che il documento modificante abbia effettivamente delle modifiche attive: senza di esse il consolidamento non può essere eseguito e secondo lo standard Akoma Ntoso un documento non deve necessariamente contenere *activeModifications*.

Se i documenti hanno superato tutti i controlli preliminari, questi vengono aperti e inseriti in due editor paralleli (Figura 4.2).

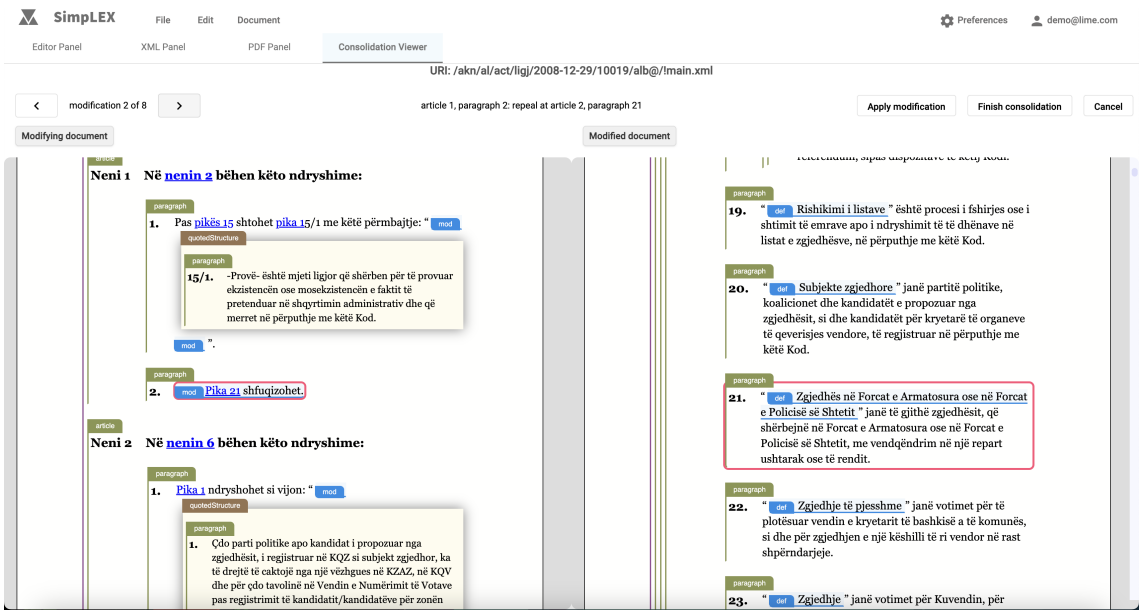


Figura 4.2: Interfaccia dopo il *setup* degli editor

In alto è riportato l'URI del documento modificato. Le modifiche attive sono estratte ed elaborate, in modo che siano marcate come non eseguite e ordinate in ordine di comparsa nel documento modificante. Le modifiche sono solitamente applicate dagli utenti in quest'ordine, pertanto è fondamentale che anche la visualizzazione lo segua, in modo da migliorare l'esperienza utente ed evitare di saltare da un punto all'altro del documento, rendendo più difficoltosa la comprensione del processo da parte di utenti non necessariamente esperti in informatica. A questo punto l'interfaccia si posiziona sulla prima modifica disponibile, mostra in alto una breve descrizione testuale della modifica riportante: il tipo, la posizione nel documento modificante e la posizione nel modificato. Un esempio è riportato nella Figura 4.3.

Figura 4.3: Esempio di descrizione di una modifica

Infine sono mostrati sui rispettivi documenti tali punti di interesse. A sinistra, nel documento modificante, la *source*, la quale comprende la porzione di testo in cui è riportata la disposizione modificativa con eventualmente il testo da sostituire o rimuovere e/o quello da inserire. Nell'*editor* di destra invece sono possibili due scenari:

- il sistema ha rilevato un'unica destinazione, pertanto la porzione di testo sarà opportunamente evidenziata (come mostrato in Figura 4.2);
- il sistema ha rilevato più destinazioni, allora sarà disponibile per l'utente un *dialog* ricollocabile che mostri le diverse destinazioni trovate, in modo che possa visualizzarle una per una con i propri *id* nei diversi punti del documento e scegliere quella desiderata (Figura 4.4).

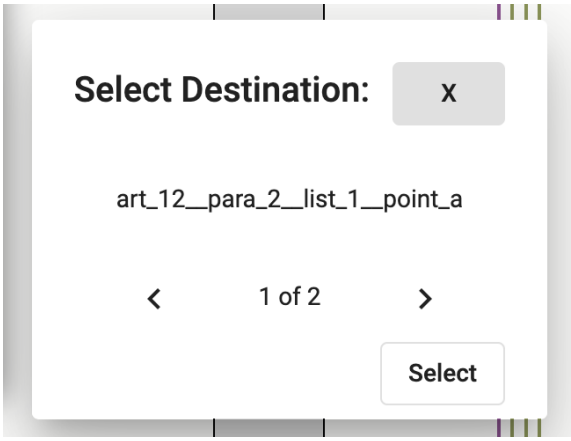


Figura 4.4: Dialog di selezione nel caso di destinazioni multiple

La possibilità di trovare più destinazioni è data dalla differenza dei riferimenti delle porzioni di testo: alcuni sono assoluti nel documento mentre altri relativi alla posizione in cui si trovano. Un esempio lampante è costituito dagli articoli, i quali presentano una numerazione assoluta in tutto il documento. Nel momento in cui si ricerca nel testo un determinato articolo questo sarà unico e trovabile a prescindere dalla sua effettiva posizione (in un capitolo piuttosto che in un altro e così via). Differentemente vanno trattati i punti, i quali nel documento modificante non contengono informazioni sulla lista in cui si trovano: se nella stessa posizione sono presenti più liste con medesimi punti non è possibile determinare algoritmicamente quale sia la lista originariamente intesa, pertanto il sistema si rifà alla scelta dell'utente. Purtroppo, in documenti di tale complessità ed eterogeneità di autori, non sono rari gli errori. Obiettivo di questa nuova implementazione è stata anche quella di rendere l'utente partecipe di eventuali errori rilevati, in modo anche da evitare che vengano applicate modifiche in maniera scorretta. In questo punto del processo infatti, se una modifica non presenta una sorgente o una destinazione ciò verrà reso noto all'utente tramite *dialog* (Figura 4.5).

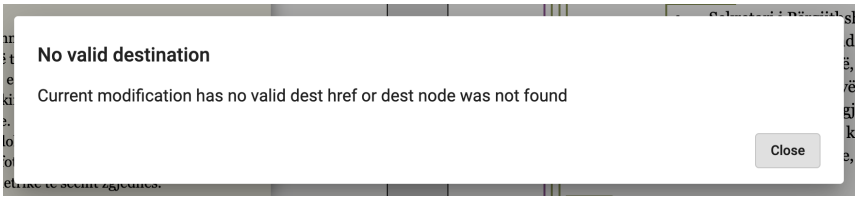


Figura 4.5: Dialog per destinazione non valida

Inoltre, tale modifica non potrà essere applicata se l'utente proverà a farlo (Figura 4.6).

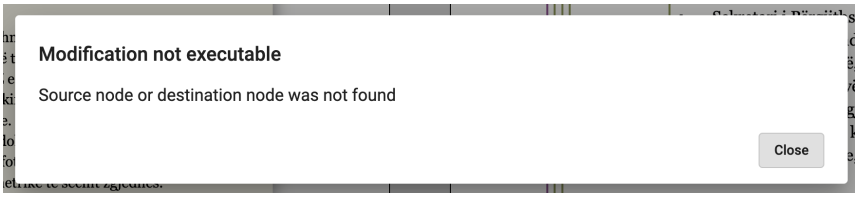


Figura 4.6: Dialog per modifica non eseguibile

Nel momento in cui l'utente decide di voler applicare una modifica, potrà farlo dopo averla selezionata tramite le frecce e premendo il pulsante *Apply modification*. Un *dialog* chiede all'utente di confermare la scelta e, in caso di risposta affermativa, la modifica è applicata sul documento modificato. Il testo rimosso viene nascosto e il testo nuovo sottolineato in verde. La modifica viene visualizzata adesso in verde anche sul documento modificante, perchè segnata come applicata e pertanto non sarà possibile applicarla nuovamente.

Simplex mette a disposizione un menù delle preferenze dal quale l'utente può scegliere una serie di opzioni: cambiare la lingua, scegliere quali pannelli avere nella propria interfaccia e diverse modalità di visualizzazione. Nel caso del consolidamento, sono state aggiunte due nuove funzionalità attivabili o disattivabili da tale menù: la visualizzazione delle abrogazioni e la rinumerazione automatica.

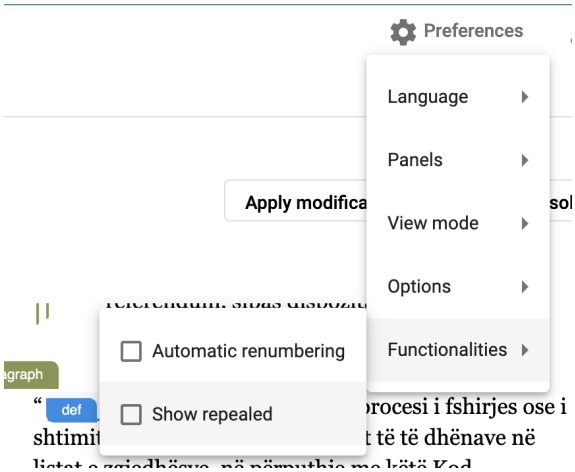


Figura 4.7: Menù delle preferenze

Nel caso delle abrogazioni l'utente può scegliere se visualizzare o meno le strutture abrogate, come è possibile osservare nelle Figure 4.8 e 4.9.

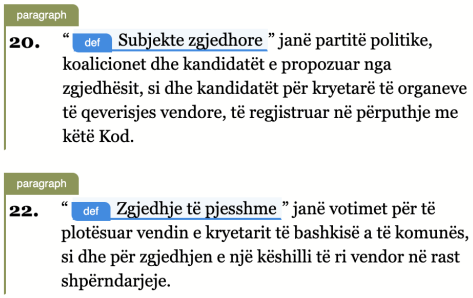


Figura 4.8: Esempio di abrogazione con visualizzazione disattivata

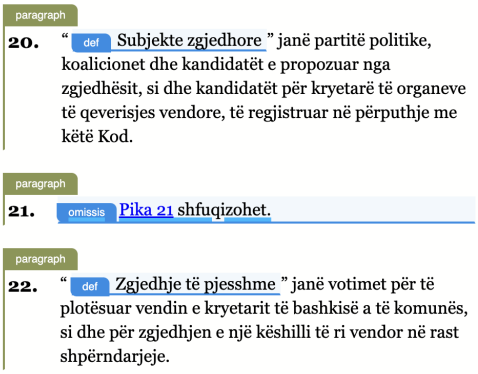


Figura 4.9: Esempio di abrogazione con visualizzazione attivata

Inoltre, nel caso in cui le abrogazioni non siano mostrate, sempre attraverso le prefenze, è possibile attivare la rinumerazione automatica, con la quale tutte le porzioni di testo successive sono rinumerate concordemente. Per esempio: la rimozione dell'articolo 21 farà sì che l'articolo 22 diventi il 21, il 23 il 22 e così via. Volendo riprendere l'esempio della figure precedenti, con la rinumerazione automatica si ot-

tiene adesso il seguente risultato: il precedente paragrafo 22 è diventato adesso il paragrafo 21 (Figura 4.10).

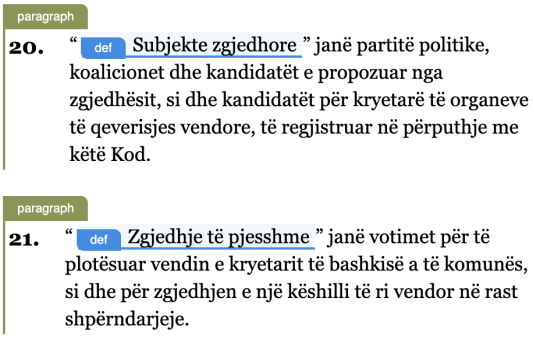


Figura 4.10: Esempio di abrogazione con rinumerazione automatica attivata

La rinumerazione è attivabile anche nei casi di inserimento.
L'utente può così decidere in totale autonomia quali modifiche applicare, quali no e in quale ordine; inoltre potrà visualizzare quelle già effettuate utilizzando le stesse frecce di scorrimento. A questo punto l'utente può decidere di annullare in toto il processo: in questo caso nessuna modifica sarà salvata e nessun documento modificato. Se, invece, soddisfatto del risultato potrà terminare il processo con il salvataggio delle modifiche. Sarà quindi creata una nuova *expression* del documento riportante nei metadati le nuove informazioni: le *passiveModifications* generate, la data di messa in vigore, la data in cui è stato eseguito il consolidamento e l'autore. L'*expression* è così salvata nel database e potrà essere consultata successivamente, anche sull'editor principale.

4.2 Inserimento

4.2.1 Inserimento di una quotedText

Il comportamento standard per l'inserimento di una *quotedText* consiste nell'aggiungere il suo contenuto, ossia una o più stringhe, in coda al contenuto della destinazione trovata.

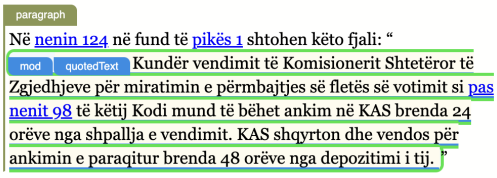


Figura 4.11: Esempio di inserimento di una quotedText (documento modificante)

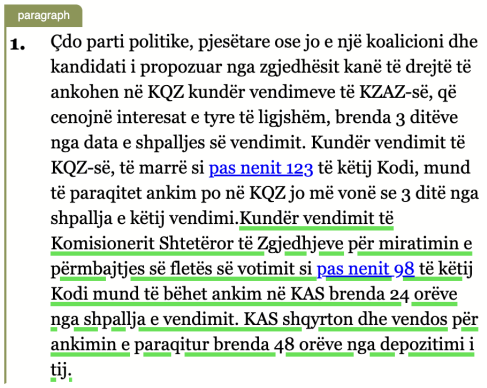


Figura 4.12: Esempio di inserimento di una quotedText (documento modificato)

4.2.2 Inserimento di una quotedStructure

L'inserimento di una *quotedStructure* consiste nell'aggiungere il suo contenuto, il quale potrebbe essere una struttura come un articolo o un paragrafo, dopo la destionazione trovata, allo stesso livello gerarchico.

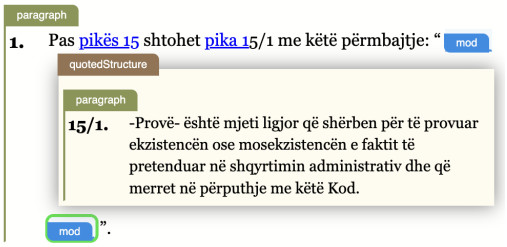


Figura 4.13: Esempio di inserimento di una quotedStructure (documento modificante)

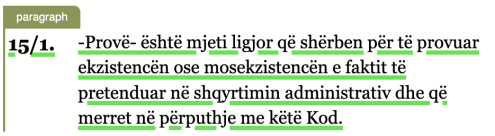


Figura 4.14: Esempio di inserimento di una quotedStructure (documento modificato)

In riferimento all'esempio riportato nelle Figure 4.13 e 4.14, se l'utente attiva la rinumerazione automatica il paragrafo inserito diventerebbe il paragrafo 16 anzichè il 15/1.

4.3 Sostituzione

4.3.1 Sostituzione di una quotedText

Nel caso in cui si voglia sostituire una *quotedText* si elimina il contenuto della destinazione trovata e si inserisce il contenuto della *quotedText*.

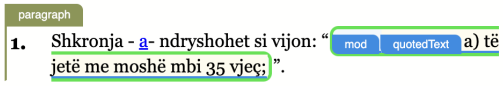


Figura 4.15: Esempio di sostistuzione di una quotedText (documento modificante)

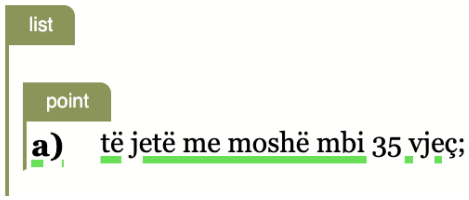


Figura 4.16: Esempio di sostistuzione di una quotedText (documento modificato)

4.3.2 Sostituzione di una quotedStructure

Nel caso, invece, di una sostituzione di una *quotedStructure* viene eliminato il contenuto della destinazione trovata e inserito il contenuto della *quotedStructure*.

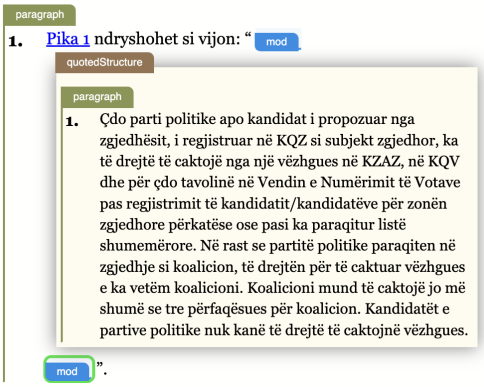


Figura 4.17: Esempio di sostistuzione di una quotedStructure (documento modificante)

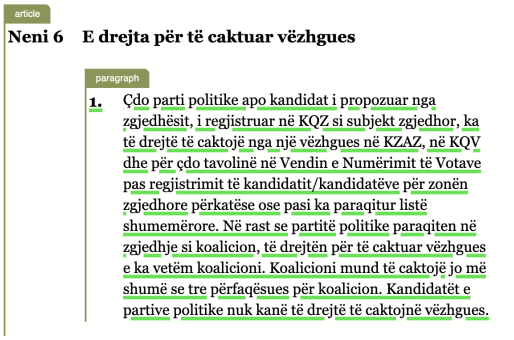


Figura 4.18: Esempio di sostistuzione di una quotedStructure (documento modificato)

4.4 Abrogazione

Infine, per eseguire un’abrograzione si procede all’eliminazione del contenuto della destinazione trovata e all’inserimento di un nodo <omissis>, al cui interno si inserisce la disposizione modificativa in linguaggio naturale. Nell’esempio nelle Figure 4.19 e 4.20 è riportato il testo in lingua albanese: “Pika 21 shfuqizohet”, in italiano “Il paragrafo 21 è abrogato”.

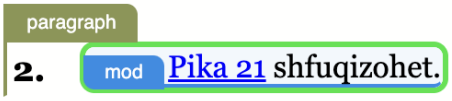


Figura 4.19: Esempio di abrograzione con visualizzazione (documento modificante)

Figura 4.20: Esempio di abrograzione con visualizzazione (documento modificato)

Capitolo 5

Architettura del ConsolidationViewer

5.1 Simplex

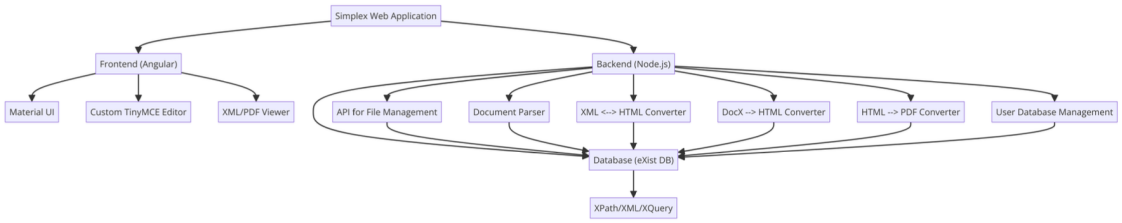


Figura 5.1: Architettura di Simplex [D'A24]

Il ConsolidationViewer fa parte del software Simplex, un web editor per documenti legislativi in formato Akoma Ntoso teorizzato dal professor Fabio Vitali e dal dott. Andrea D'Arpa e finanziato dall'Università di Bologna. Simplex consiste in un'applicazione web il cui client è affidato al framework Javascript Angular, mentre il lato server è realizzato con NodeJs e un insieme di database [Po24]. (Figura 5.1)

5.1.1 Simplex Server

Il backend di Simplex conta due entità fondamentali: il database e l'applicazione server. Il database principale per la conservazione dei documenti Akoma Ntoso è *eXistdb*, un DBMS (*DataBase Management System*) open-source per la gestione e interrogazione di documenti XML. Appartiene alla categoria dei database NoSQL orientati ai documenti, ossia quei database in cui l'unità principale di archiviazione e gestione è il documento [eX18]. Il database degli utenti è invece un database MongoDB, nel quale vengono memorizzati dati in formato JSON sugli utenti per la gestione degli accessi e dei permessi sui documenti. Poichè gli utenti finali del software saranno enti governativi, con altissimi standard per quanto riguarda la sicurezza, il modulo può essere facilmente sostituito con altre tecnologie, come per esempio il protocollo di autorizzazione SPID di Lepida [D'A24]. Nell'applicazione server vera e propria è possibile trovare il modulo per la gestione dei documenti, il quale ha il compito di salvare e recuperare i documenti da eXist. Altro modulo che riveste un ruolo fondamentale è il modulo per la conversione dei documenti, il quale offre le seguenti conversioni:

- da XML a XHTML, una particolare estensione di HTML che l'editor TinyMCE utilizza per lavorare sul testo, in modo da convertire i documenti Akoma Ntoso in HTML usabili negli editor TinyMCE;
- da XHTML a XML, in modo da riconvertire i documenti manipolati da TinyMCE in documenti Akoma Ntoso corretti;
- da Docx a XHTML, per l'importazione di documenti provenienti da altri *word processors* come Microsoft Word;
- da HTML a PDF, per esportare i documenti in formato .pdf.

5.1.2 Simplex Client

Il framework scelto per la realizzazione del frontend di Simplex è, come accennato, Angular. Angular è un framework Javascript open-source, sviluppato e mantenuto da Google. Il suo obiettivo è quello di semplificare sia lo sviluppo che il testing di applicazioni web single-page, ossia composte da una singola pagina web dinamica. Angular utilizza HTML e TypeScript, un'estensione di Javascript il cui tratto distintivo è un sistema di annotazione dei tipi che permette di controllarli in fase di compilazione. Il framework si basa sul concetto di componente, che consiste in un blocco di codice autonomo e che può essere opportunamente riutilizzato. Ogni componente definisce il proprio template grafico attraverso un apposito file HTML, seguito dal suo corrispondente file CSS, e implementa la logica sottostante nel file TypeScript. La suddivisione dell'intero applicativo in componenti è ormai un espediente molto utilizzato in diversi framework Javascript, poichè facilita la manutezione, gestione e implementazione di nuove funzionalità. Angular inoltre fornisce supporto alle architetture Model-View-Controller e Model-View-ViewModel, quest'ultima adottata in Simplex. Altro concetto fondamentale presente in Angular è il concetto di *service*, il quale è definito come qualsiasi valore, funzione o feature di cui l'applicazione ha bisogno. Un service è solitamente una classe con uno scopo ben definito. Il loro utilizzo permette di separare le feature relative alla view dei componenti da altri tipi di processi, in modo da rendere i componenti più chiari ed efficienti [Go24].

Per quanto riguarda invece l'interfaccia utente essa è realizzata tramite la libreria Angular Material, la quale segue i principi di *User Experience* (UX) forniti da Google. La libreria permette agli elementi dell'interfaccia di adattarsi dinamicamente e allo stesso tempo di avere un aspetto piacevole pur rimanendo intutivi nell'utilizzo. Ultimo elemento del client di Simplex, ma non per questo meno importante, è l'integrazione della libreria TinyMCE per la realizzazione dei diversi editor presenti

nell'applicazione. TinyMCE è un editor online che opera su testo formattato; nel nostro caso, i documenti sono in formato XHTML. Esso si definisce un editor WYSIWYG ("What You See Is What You Get"), in quanto permette agli utenti di visualizzare a schermo il contenuto dei documenti nella loro formattazione finale, senza che essi si debbano preoccupare di come questo risultato sia ottenuto [Co24]. In questo modo l'utente può modificare i documenti in modo simile ad un classico editor di testo, poichè Tiny gestisce autonomamente i cambiamenti nella struttura, senza che siano richieste nozioni di HTML o CSS. L'uso di TinyMCE, unitamente al servizio di conversione da Akoma Ntoso a XHTML e viceversa, forniscono un'interfaccia ideale per la modifica dei documenti legislativi, sia dal punto di vista dell'utente ma anche per quanto riguarda lo sviluppo di nuove funzionalità che prevedono la manipolazione dei documenti, come il consolidamento.
L'interfaccia di Simplex è interamente configurabile tramite un file di configurazione in formato JSON. Il file di configurazione nasce dall'esigenza di superare la principale limitazione di Lime, ossia la creazione di diverse versioni a seconda delle singole esigenze del cliente. In questo modo, solo il file di configurazione sarà modificato, lasciando il resto del progetto inalterato, evitando la duplicazione di bug e favorendo la manutenibilità del sistema. Il file di configurazione possiede le seguenti informazioni:

- *identification*: informazioni che identificano la specifica versione dell'applicazione;
- *localFiles*: eventuali file locali necessari all'applicazione per funzionare correttamente;
- *server*: gli URI di tutti i servizi server-side;
- *layout*: l'intera organizzazione gerarchica dei componenti visibili a schermo, comprende quindi contenitori, view e types posizionati nel layout della window,

il nodo padre della gerarchia;

- *components*: il contenitore di tutte le sottostrutture che sono definite al di fuori degli elementi a cui appartengono.

5.2 ConsolidationViewer

Il ConsolidationViewer, contrariamente a quanto suggerisce il suo nome, gestisce sì l'interfaccia visiva, ma implementa anche la logica sottostante del consolidamento. Si occupa dell'interazione con l'utente attraverso i *dialog*, ma provvede anche all'esecuzione materiale delle modifiche sul documento, alla modifica dei metadati al termine del processo, nonché al salvataggio dei nuovi documenti. Di seguito è riportato lo schema architetturale del client di Simplex, con particolare riferimento ai componenti introdotti dal progetto per la gestione del consolidamento.

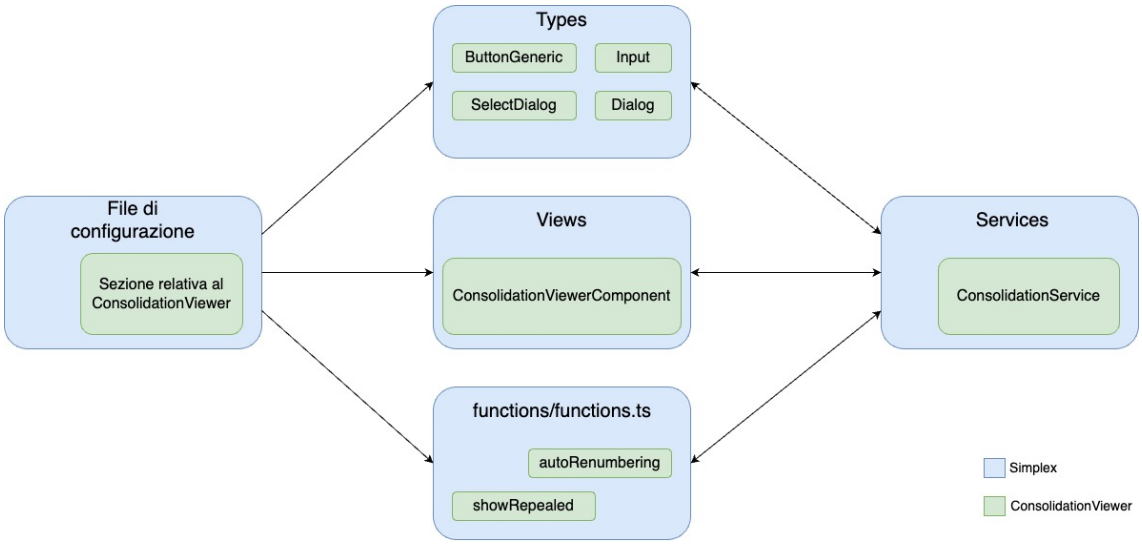


Figura 5.2: Architettura del client di Simplex e ConsolidationViewer

5.2.1 File di configurazione

Nel file di configurazione (*/assets/config.json*), all'interno della proprietà *layout*, è stata definita la porzione di interfaccia relativa al ConsolidationViewer, riportata di seguito nella sua struttura essenziale.

```
{
    "type": "tabElement",
    "label": "CONSOLIDATION_VIEWER",
    "action": "toggleConsolidationViewer",
    "evaluate": "evaluateConsolidationViewer",
    "height": "95%",
    "items": [
    {
        "type": "horizontalContainer",
        "view": "CONSOLIDATION_BUTTONS",
        "items": [ {lista degli elementi grafici}, ...,
            {
                "view": "consolidationViewer",
                "description": "double editor"
            }
        ]
    }]
}
```

Il nodo padre di questa gerarchia è costituito da un *tabElement*, ossia un elemento che serve a popolare il padre *tabContainer* di *mat-tab*. In questo modo l'intera interfaccia dedicata al ConsolidationViewer è racchiusa nella propria tab nell'interfaccia complessiva.

I metodi *action* ed *evaluate* fanno riferimento a delle apposite funzioni definite nel file *functions/functions.ts*, il cui scopo principale è quello di fare funzionare il passaggio da una tab all'altra dell'interfaccia. All'interno della *tabElement* è inserito un *horizontalContainer*, il quale può contenere altri elementi disponendoli per colonne; il funzionamento è pari al "display: flex" in CSS. All'interno dell'*horizontalContainer* sono inseriti tutti gli elementi dell'interfaccia: un insieme di *input* per mostrare a schermo gli URI dei documenti, una serie di pulsanti, una *label* per la descrizione delle modifiche e infine il *consolidationViewer* vero e proprio. L'interfaccia in questo modo è altamente personalizzabile, in quanto è possibile rimuovere o aggiungere un nuovo elemento facilmente, senza la necessità di stravolgere ciò che è stato già creato, se non con piccoli aggiustamenti.
Generalmente ogni elemento inserito nel layout fa riferimento e renderizza un'istanza di un componente, solitamente categorizzato come *type* o *view*. I *type* sono piccoli componenti, tendenzialmente riutilizzabili in diversi punti del progetto e per scopi non sempre identici. Le *view* sono componenti generalmente più grandi, con scopi ben precisi e solitamente utilizzati una volta sola. Per la definizione dell'interfaccia del ConsolidationViewer sono stati definiti dei nuovi *type*:

- *input*: i quali renderizzano un tag input con la propria label associata, sono istanze dell'InputComponent;
- *button-generic*: i quali renderizzano un pulsante per cui è possibile definire autonomamente la callback per l'evento onclick, in contrapposizione ai bottoni già presenti nel progetto, sono istanze del ButtonGenericComponent;

Seguono adesso degli esempi di *input* e *button-generic* inseriti nel file di configurazione finale:

```
{
    "type": "input",
```

```
    "label": "MODIFYING_DOCUMENT",
    "disabled": "true",
    "flexGrow": "1",
    "paddingLeft": "0.5%"
},
{
    "type": "button-generic",
    "label": "SELECT_MODIFYING_DOCUMENT",
    "paddingLeft": "0.5%",
    "style": "consolidation"
}
```

La proprietà *label* viene utilizzata successivamente come identificativo per ogni elemento. Tutte le proprietà di stile hanno dei nomi che rispecchiano le corrispettive proprietà CSS, così anche i valori inseriti. Infine, la proprietà *action* del *button-generic* se specificata fa riferimento a una funzione in *functions/functions.ts*. I rispettivi componenti saranno esaminati in dettaglio nella sottosezione 5.2.4, intitolata "Componenti aggiuntivi". La nuova *view* realizzata è invece la *consolidationViewer*, la quale renderizza un'istanza del ConsolidationViewerComponent, sul quale ora si soffermerà la trattazione.

5.2.2 ConsolidationViewerComponent

Il ConsolidationViewerComponent è il componente che si occupa della gestione degli editor e dei documenti selezionati. È composto da un file HTML e un file CSS che definiscono l'interfaccia visibile e un file TypeScript, posizionati nella cartella */components/views/consolidation-viewer*. Il componente riceve in input la lista degli elementi elencati nel file di configurazione e, durante l'inizializzazione, imposta i

riferimenti e definisce tutti i comportamenti, quali le *clickFunction* dei pulsanti. Per la selezione dei documenti si è deciso di riutilizzare la funzione classica di apertura, già usata nell'editor principale, in quanto fornisce un *dialog* per la selezione del documento all'interno del database *eXist*, permettendo di ottenere sia il documento in formato HTML che i metadati parsati. Il ConsolidationViewerComponent richiama così il FileManagerComponent, che si occupa di gestire il *dialog* mostrante lo stato del database. Una volta effettuata una scelta, i metadati sono parsati e memorizzati nel corrispondente MetadataService, mentre il contenuto del documento in formato HTML è memorizzato in un'apposita struttura dati all'interno del componente stesso.

Il *setup* degli editor avviene nel momento in cui entrambi i documenti hanno superato i controlli preliminari, come descritto nella sezione 4.1 intitolata "Funzionamento del ConsolidationViewer", per garantire che tutte le condizioni necessarie siano soddisfatte. Il componente si occupa di creare due nuove istanze di TinyMCE e, attraverso l'utilizzo del costrutto *Promise.all*, attende che entrambi siano inizializzati correttamente per impostarne il contenuto e renderli visibili all'utente.

Per poter scorrere efficacemente tra le diverse modifiche è stata realizzata una funzione di *scroll* sulla finestra dell'editor dato un certo *Element*, ossia un'istanza della classe base per tutti gli elementi del DOM, riutilizzando la funzione di *scroll* fornita da TinyMCE. Ogni volta che l'utente utilizza le frecce di scorrimento sull'interfaccia viene richiamata la funzione *scrollOnCurrentModification*. Questa controlla se i nodi sorgente e destinazione siano già stati calcolati, nel caso in cui l'utente abbia già visualizzato una volta la modifica, in modo da allentare lo sforzo computazionale. Al momento, la ricerca della sorgente è implementata direttamente nel componente con un semplice *querySelector* tramite identificatore. Non si esclude che in futuro si potrà affinare la ricerca utilizzando un'euristica in grado di catturare casi più complicati. Invece, la ricerca delle destinazioni, essendo più complessa è stata im-

plementata nel ConsolidationService, pertanto per una trattazione più approfondita si rimanda il lettore alla sottosezione successiva, intitolata "ConsolidationService". Nel caso in cui il service restituisca più destinazioni, allora è aperto il *dialog* di selezione, altrimenti è restituito direttamente il nodo trovato. Se la ricerca dei nodi nei rispettivi documenti non è andata a buon fine si è deciso di avvisare l'utente tramite *dialog* prima di procedere allo scorrimento degli editor sui punti di interesse. Le funzioni di scorrimento singole si occupano anche dell'aggiunzione e della rimozione dello stile, in modo che vengano opportunamente evidenziate all'utente solo le parti di testo coinvolte dalla modifica corrente.
Un'altra azione regolata dal componente è il salvataggio del documento modificato una volta che il processo è stato concluso. Secondo lo standard Akoma Ntoso il consolidamento produce una nuova *expression* del documento modificato seguendo il modello FRBR. Quindi, piuttosto che la modifica del documento, si utilizza l'operazione già definita del "Salva come...", in quanto la modifica dei metadati del documento produce un URI diverso. In questo modo il documento sarà salvato come un nuovo documento e si differenzierà dal precedente. Entrando più nel dettaglio, si è deciso di modificare i dati relativi a data e autore sia dell'*expression* che della *manifestation* nel seguente modo:

- per quanto riguarda l'*expression*:
  - la data sarà la data del *work* del documento modificante;
  - l'autore sarà pari a quello del *work* del documento modificante;
- per quanto riguarda la *manifestation*:
  - la nuova data scelta è la data del giorno corrente;
  - l'autore è l'username dell'utente che sta effettuando il processo di consolidamento, ottenuto grazie allo UserService;

La modifica dei metadati avviene richiamando una funzione creata appositamente nel MetadataService, in modo da poter generare i nuovi URI in base alle nuove informazioni. Al termine del processo inoltre, è essenziale l'aggiunzione delle nuove *passiveModifications* derivanti dalle modifiche applicate; per ognuna di esse è generata una modifica che dovrà essere inserita nei metadati del documento modificato. Infine, è bene ricordare che le istanze degli editor possono gravare sul carico computazionale; pertanto, nel momento in cui il processo è finito o annullato, una funzione di reset si occupa di inizializzare il contenuto e di distruggere tali istanze.

5.2.3 ConsolidationService

Il ConsolidationService si occupa della gestione delle modifiche attive e di tenere traccia della modifica corrente, per poter eventualmente eseguire le azioni di modifica su di essa. Il relativo file TypeScript si trova in */services*.
Le modifiche attive sono memorizzate in un vettore, dopo averle ricevute dal ConsolidationViewerComponent in seguito al *parsing* dei metadati del documento modificante. Inoltre, è definita una struttura dati per mantenere le informazioni relative alla modifica corrente tra cui l'indice nel vettore, la descrizione e i nodi trovati. Grazie a tali strutture dati, il ConsolidationService si occupa di gestire lo scorrimento sull'array di modifiche attive, comunicando le nuove informazioni al ConsolidationViewerComponent per permettere lo scorrimento sull'interfaccia visibile.
Il service si occupa anche, come accennato precedentemente, della ricerca delle destinazioni nel documento modificato. Attualmente la ricerca assume che solo i punti abbiano un riferimento relativo, pertanto se il riferimento non contiene punti, la ricerca è implementata tramite un *querySelector*. Invece, per la gestione dei punti, è determinante l'utilizzo delle espressioni regolari per il controllo dell'eventuale esistenza di più liste con stessi punti. Nel caso in cui siano restituiti più nodi, si è scelto di delegare la decisione del nodo desiderato all'utente, pertanto sarà compito

del ConsolidationViewerComponent comunicare la scelta al ConsolidationService.
Cuore del service è sicuramente la funzione di applicazione delle modifiche, la quale, a seconda del tipo di modifica, esegue una diversa funzione a scelta tra: *insertion*, *repeal* e *substitution*. Inoltre, segna la modifica come eseguita, permettendo l'aggiornamento dello stile concordemente. Le singole funzioni di modifica sono state definite separatamente, in modo da aumentare la modularità e la leggibilità del codice. Inoltre, si è voluto tenere conto anche della possibilità, in futuro, di implementare nuovi tipi di modifiche.
Inserimenti e abrogazioni supportano la rinumerazione automatica quando attivata dall'utente nel menù delle preferenze. Simplex mette a disposizione un *service*, il M*oduleStateService*, il quale permette di visualizzare o modificare le preferenze dell'utente memorizzate nel *localStorage*. Nel caso della rinumerazione automatica la preferenza è espressa tramite un valore booleano, il quale è controllato ogni volta che un'abrogazione o un inserimento è applicato. Nel caso in cui la preferenza sia attiva, si fa uso del già implementato *numberingService* per l'aggiornamento della numerazione.
Altra funzionalità delegata al ConsolidationService è la generazione di una descrizione in linguaggio naturale a partire da una modifica. A partire dai riferimenti di sorgente e destinazione e dal tipo della modifica, è creata una stringa attraverso l'uso delle espressioni regolari, in modo anche da espandere le abbreviazioni; per esempio, gli articoli sono solitamente indicati con la stringa "art" oppure i paragrafi abbreviati con "para". Nelle seguenti figure (Figure 5.3 e 5.4) è possibile osservare un esempio di modifica, assieme ai dati relativi a tipo, sorgente e destinazione, e la descrizione corrispondente generata dal ConsolidationService.

```
<textualMod eId="pmod_1" type="insertion">
    <source href="#art_1__para_1__mod_1"/>
    <destination href="/akn/al/act/ligj/2008-12-29/10019/!main/#art_2__para_15"/>
    <new href="#art_1__para_1__mod_1__qstr_1"/>
</textualMod>
```

Figura 5.3: Esempio di modifica e suoi dati

article 1, paragraph 1: insertion at article 2, paragraph 15

Figura 5.4: Descrizione corrispondente generata

Il service, gestendo l'array delle modifiche attive, si occupa anche del suo ordinamento. Quando le modifiche attive sono inserite nei metadati dei documenti non sempre sono nell'ordine di comparsa, pertanto è sembrato naturale eseguire l'ordinamento per migliorare l'esperienza utente. La funzione di ordinamento estrae i numeri delle strutture testuali dai riferimenti delle sorgenti ed esegue il *sort* dalla più significativa, ossia quella più a sinistra, a quella meno significativa. A parità di cifra si considera quella successiva.
Per concludere, il service offre una serie di funzioni che espongono lo stato attuale dell'array di modifiche e della modifica corrente. Ne sono esempio la funzione che restituisce le informazioni sulla modifica corrente o quella che restituisce un valore booleano a seconda che la modifica corrente sia stata eseguita o meno.

5.2.4 Componenti aggiuntivi

Per la realizzazione dell'intero sistema sono stati implementati altri piccoli, ma fondamentali, componenti. Il ButtonGenericComponent e l'InputComponent, come menzionato precedentemente, sono stati creati per la gestione dell'interfaccia.
Il primo rappresenta un'estensione del già presente ButtonComponent; il template è costituito solo da un tag *button* la cui etichetta corrisponde alla proprietà *label* specificata nel file di configurazione oppure a una sua traduzione. Il ButtonGeneric

riceve in input la cosiddetta *clickFunction*, una funzione che verrà eseguita dopo un click sul pulsante. Questo meccanismo permette di definire la callback al di fuori del componente stesso, solitamente nel padre che lo richiama nel proprio template HTML. Se la *clickFunction* non è definita, esso ha lo stesso comportamento del ButtonComponent normale.

L'InputComponent, a livello di elementi HTML, è costituito da un form minimale, contenente un elemento input e la sua *label*. La stringa di testo da inserire nella *label* è definita nel file di configurazione, mentre l'*inputValue* può essere impostato in maniera analoga alla *clickFunction* dei bottoni.

Per migliorare l'interazione del sistema con l'utente, invece, sono stati implementati dei *dialog*.

Primo tra questi è il DialogComponent, il quale renderizza un *mat-dialog* essenziale. Il componente è stato progettato in modo che fosse completamente personalizzabile dal componente chiamante. Le unità sempre presenti sono il titolo, il paragrafo e un pulsante per la chiusura del *dialog* stesso. Il DialogComponent riceve in input anche un valore booleano, il quale esprime se il *dialog* deve essere di sola visualizzazione o se è necessario che l'utente prenda una decisione binaria, mostrando così un secondo pulsante, anch'esso personalizzabile. Alla sua chiusura, il *dialog* comunica al chiamante una stringa, dalla quale si può desumere la scelta effettuata ed eseguire delle azioni piuttosto che altre.

I precedenti componenti hanno carattere generale e sono stati progettati anche con l'intenzione di poterli riutilizzare in futuro e in altre sezioni del progetto complessivo. Tuttavia, è stato necessario implementare un *dialog* apposito per la scelta delle destinazioni multiple: il SelectDialog. Quando aperto, il *dialog* riceve una lista di nodi e la funzione di scorrimento sugli editor. Attraverso le frecce l'utente può visualizzare i diversi nodi sul testo e sceglierne uno attraverso un pulsante. Alla chiusura, il SelectDialog restituisce al chiamante il nodo desiderato dall'utente.

5.2.5 Visualizzazione delle abrogazioni

Ultima interessante funzionalità è la visualizzazione delle abrogazioni, la quale può essere attivata o disattivata dall'utente in qualsiasi momento dal menù delle preferenze. Si è scelto di permettere all'utente di modificare la preferenza anche dopo aver eseguito delle abrogazioni, in modo da non rendere la preferenza iniziale vincolante. Per questo motivo, le abrogazioni sono eseguite sempre attraverso le stesse operazioni, descritte nella sezione 4.4, intitolata "Abrogazione". La struttura risultante, contenente il tag *omissis*, sarà sempre presente, ma nascosta o visibile in base al relativo valore booleano memorizzato nelle preferenze. Tale struttura avrà inoltre la proprietà *status* con valore *editorial*, al fine di segnalare che la sua presenza è una scelta di natura editoriale. In questo modo, in futuro sarà possibile rimuovere facilmente dai documenti gli elementi di tipo editoriale, non sempre necessari. La gestione della feature di visualizzazione avviene nel *functions/functions.ts* in un'apposita funzione, la quale ottiene dal *consolidationService* le modifiche di tipo abrogativo già eseguite e modifica la proprietà "*display*" in accordo al nuovo valore booleano selezionato.

Capitolo 6

Valutazione

In questa sezione intendo riportare una valutazione complessiva del sistema realizzato e di come questo risolva, almeno in parte, i problemi delle precedenti implementazioni. La valutazione è stata effettuata attraverso numerosi test durante e alla fine dello sviluppo, volti alla verifica della correttezza delle operazioni ma anche alla loro efficacia, dal punto di vista dell'esperienza utente. A tal proposito, i test sono stati condotti con l'ausilio di un gruppo di esperti nello sviluppo e nell'uso di editor di documenti in formato Akoma Ntoso. L'esperienza e i riscontri degli esperti hanno permesso di poter migliorare il sistema, correggendo errori e avendo la possibilità di considerare casi d'uso non pensati precedentemente. Fondamentale è stato anche l'utilizzo di documenti già marcati reali, riflettenti la complessità ed eterogeneità dei documenti di quelli che saranno gli utenti finali del software.

L'analisi è stata effettuata principalmente attraverso la realizzazione del processo di consolidamento su coppie di documenti provenienti dal contesto professionale, da diverse legislazioni. Il test in primis ha richiesto che gli esperti selezionassero i documenti scelti dal database e procedessero alla libera navigazione all'interno dei documenti, anche attraverso le frecce di scorrimento. La creazione di uno stile ad hoc per l'evidenziazione delle modifiche, in particolare delle porzioni di testo coinvolte

dalla modifica in entrambi i documenti, ha permesso una facile individuazione della modifica in oggetto, unitamente alla rivisitata descrizione in linguaggio naturale della modifica. Difatti essa, riportando sia i riferimenti nel documento modificante che nel modificato, ha permesso di verificare la correttezza delle evidenziazioni e delle funzioni di scorrimento sviluppate. Inoltre, la decisione di sviluppare due stili differenti per mettere in evidenza quando una modifica deve essere ancora applicata o meno, è risultata in una maggiore chiarezza da parte degli utenti dello stato corrente del processo.

Nell'esplorazione delle varie modifiche espresse, i partecipanti si sono confrontati anche con la presenza di modifiche con riferimenti non corretti. In questo caso, l'individuazione di tali modifiche non corrette è stata immediata, grazie all'introduzione di appositi *dialog* riportanti in breve il tipo di errore riscontrato. In questo modo, i valutatori sono stati avvisati dell'errore prima di un eventuale fallimento della funzione di scorrimento sugli editor. La gestione di riferimenti non corretti è indirettamente gestita anche dal nuovo algoritmo di ordinamento delle modifiche, il quale, al fine di porre le modifiche in ordine di comparsa nel documento modificante, ha fatto in modo di lasciare per ultime eventuali modifiche con *source* non corrette. Così, gli esperti hanno evidenziato un miglioramento nella fluidità durante l'esplorazione dei documenti e, contestualmente, una maggiore consapevolezza dell'applicabilità di una modifica, sebbene sia stato comunque introdotto un controllo che impedisce l'applicazione di modifiche prive dei requisiti necessari, con conseguente avviso all'utente tramite *dialog*.

Nell'applicazione di una modifica è stato deciso di richiedere all'utente una conferma esplicita della sua volontà, vista la complessità dei documenti e del processo in sé, ma considerata anche l'attuale impossibilità di annullare una singola modifica. A tal proposito sono stati introdotti dei nuovi pulsanti sull'interfaccia che potessero ripristinarla sia nella fase di selezione dei documenti, sia a processo iniziato. In que-

sto modo, i partecipanti non sono stati costretti a ricaricare la pagina web al fine di ricominciare il processo da zero ed evitare il salvataggio di documenti non desiderati. Il confronto con il gruppo di esperti è stato fondamentale per portare alla luce casi d'uso e feature non inizialmente pensati. Un importante risultato di questi feedback è stato l'implementazione di un controllo sulle date dei documenti selezionati, volto a garantire che il documento modificato sia antecedente a quello modificante. In caso contrario, è stato deciso di chiedere all'utente se desidera invertire l'ordine dei documenti. Senza il contributo dei valutatori, questa idea non sarebbe mai emersa. Un'altra funzionalità nata dal confronto con gli esperti è stata la visualizzazione delle abrogazioni, ossia la possibilità da parte dell'utente di visualizzare o meno le strutture abrogate. Tale funzionalità permette di dare rappresentazione ai diversi contesti in cui il consolidamento è usato. Ad esempio, per l'Unione europea, che dispone di un complesso sistema di emendamenti in cui è possibile avere più documenti modificanti per lo stesso modificato, è fondamentale monitorare le strutture abrogate, cercando inoltre di evitare la rinumerazione. Per questo motivo, la rinumerazione automatica è stata implementata, ma con la possibilità di lasciare la scelta all'utente, in modo che il sistema possa adattarsi al meglio agli stili e alle metodologie delle diverse legislazioni.

Un risultato significativo delle sessioni di test e di valutazione è costituito dalla più complessa ricerca di eventuali destinazioni multiple quando sono presenti dei punti nei riferimenti. Questa nuova implementazione ha aperto la strada a una ricerca più precisa che includa maggiormente l'utente, scegliendo di affidare a quest'ultimo la scelta della destinazione desiderata. Sebbene tale ricerca sia stata implementata solo per i punti delle liste, costituisce il punto di partenza per lo sviluppo di un algoritmo di ricerca sempre più preciso e sempre più adattabile ai diversi scenari che si possono incontrare in tali documenti.

Le sessioni di test hanno previsto, inoltre, dei momenti volti alla verifica della corret-

tezza dei nuovi documenti Akoma Ntoso. Con il salvataggio alla fine del processo, è generata una nuova *expression* del documento modificato, quindi è stato necessario che il nuovo documento presentasse i corretti metadati risultanti e le nuove porzioni di testo. A tal proposito, la generazione di una nuova *expression*, con conseguente nuovo URI, ha reso possibile il salvataggio del nuovo documento senza sovrascrivere il documento modificato precedente, operazione che, invece, avveniva in Lime. Per concludere, durante lo sviluppo è stata dedicata particolare attenzione al costo computazionale, adottando accorgimenti per ridurre il carico ove possibile. Una delle componenti più "onerose" del sistema è rappresentata dagli editor TinyMCE; di conseguenza, sono state curate con precisione sia la loro inizializzazione che la loro distruzione al termine del processo. Un'ottimizzazione significativa è stata evitare l'istanza degli editor nel caso in cui i documenti selezionati non superassero i controlli preliminari, ad esempio se il documento modificante non contiene modifiche attive. Così facendo, gli editor vengono attivati solo quando il processo può essere effettivamente eseguito e vengono eliminati una volta completato o annullato.

Le feature descritte in questo capitolo rappresentano una novità rispetto agli strumenti precedenti per la realizzazione del consolidamento e, anche e soprattutto grazie al supporto continuo degli esperti, si è ottenuto un processo più fluido, più chiaro e meno macchinoso; l'utente ha piena libertà nel determinare il modo in cui eseguire il processo, sia per quanto riguarda la possibilità di applicare determinate modifiche anziché altre, sia per quanto concerne le diverse funzionalità attivabili in base alle proprie esigenze. Inoltre, è stato possibile considerare anche funzionalità non ancora implementate, ma che potrebbero esserlo nel prossimo futuro. La scelta di separare la gestione del consolidamento in sé dalla gestione delle modifiche, integrando quest'ultima in un apposito *service*, ha aumentato non solo la leggibilità del codice a favore dei futuri sviluppatori, ma anche la manutenibilità del sistema. In questo modo, è possibile accedere alle operazioni di modifica o allo stato attuale del vetto-

re delle modifiche attive anche nelle altre parti del progetto. Questo ha permesso l'introduzione della feature relativa alla visualizzazione delle abrogazioni e prepara il terreno per la realizzazione di nuove funzionalità simili. Per concludere, di seguito sono riportate due tabelle riepilogative che confrontano le funzionalità implementate in Lime e nel ConsolidationViewer.

Feature	**Lime**	**ConsolidationViewer**
Selezione documenti dal DB	Sì	Sì
Frecce di scorrimento	Sì	Sì
Evidenziazione parti di testo coinvolte	Sì, ma non sempre visibile	Sì
Inserimenti	Sì	Sì
Sostituzioni	Sì	Sì
Abrogazioni	Sì	Sì
Supporto nodi wrapUp	Sì	Sì
Annullamento del processo	No	Sì
Controllo presenza modifiche attive	No	Sì
Scambio documenti per date errate	No	Sì
Visualizzazione diversa per modifiche applicate	No	Sì
Modifiche in ordine di comparsa	No	Sì
Descrizione modifica con riferimenti completi	Solo riferimenti modificato	Sì
Rinumerazione automatica	No	Sì
Visualizzazione abrogazioni	No	Sì
Libertà ordine di applicazione	No	Sì

Tabella 6.1: Tabella del confronto tra Lime e il ConsolidationViewer

Feature	**Lime**	**ConsolidationViewer**
Algoritmo per determinazione di liste	No	Sì
Possibilità di scegliere la destinazione tra diverse	No	Sì
Aggiornamento metadati	No	Sì
Salvataggio che non sovrascrive i documenti	No	Sì

Tabella 6.2: Tabella del confronto tra Lime e il ConsolidationViewer

Capitolo 7

Conclusioni e sviluppi futuri

In questa tesi è stato dimostrato e descritto come il ConsolidationViewer in Simplex sia un buono strumento per la realizzazione del consolidamento diretto deliberativo e di come questo risolva parte dei problemi presenti nelle precedenti implementazioni. Il ConsolidationViewer offre supporto alle operazioni di inserimento, sostituzione e abrogazione, sia di sole stringhe di testo che di intere strutture testuali aventi il proprio markup. I nuovi pulsanti di annullamento permettono all'utente di reimpostare l'interfaccia in caso di errori senza essere costretti a ricaricare la pagina web. Le modifiche sono adesso evidenziate nei due testi paralleli e presentano uno stile che indichi chiaramente se sono già state applicate o meno. Inoltre, esse sono ordinate in ordine di comparsa nel documento modificante grazie all'introduzione di un apposito algoritmo. Sull'interfaccia è presente una nuova descrizione della modifica in linguaggio naturale. Sono state aggiunte nuove funzionalità attivabili dall'utente in base al proprio stile e alla legislazione di provenienza: la visualizzazione delle strutture abrogate e la rinumerazione automatica dopo inserimenti e abrogazioni. Per quanto riguarda la ricerca delle destinazioni, è stata implementata una ricerca più precisa capace di controllare l'eventuale esistenza di più liste con stessi punti. Inoltre, è supportata la possibilità di avere più destinazioni e si delega all'utente

la scelta di quella desiderata. Il nuovo sistema pone una maggiore attenzione all'esperienza utente, attraverso l'utilizzo di *dialog* che segnalino eventuali errori nei metadati relativi alle modifiche che impediscano la loro applicazione ed errori relativi alla selezione dei documenti, come la non selezione, un documento modificante antecedente al relativo modificato o un documento modificante senza modifiche attive. Infine, il salvataggio del nuovo documento a fine processo con i nuovi metadati corrispondenti, ossia *passiveModifications*, date e autori di *expression* e *manifestation*, senza la sovrascrizione del documento precedente.

Il sistema realizzato pone le basi per un processo sempre più intuitivo, con l'obiettivo di incentivare un numero crescente di utenti ad automatizzare il consolidamento, anziché eseguirlo manualmente. Lo standard Akoma Ntoso, progettato per modellare le peculiarità delle diverse legislazioni, offre una vasta gamma di casi d'uso, con l'impiego di specifici tag in alternativa ad altri e differenti modalità di strutturazione dei documenti e del relativo markup. Di conseguenza, gli strumenti sviluppati per la gestione di tali documenti devono essere in grado di riflettere questa complessità. Il sistema descritto in questa tesi affronta solo una parte di queste casistiche, ma potrà essere ampliato e perfezionato nel prossimo futuro.

Una funzionalità concepita ma non implementata, poiché ritenuta meno prioritaria rispetto ad altre, è l'introduzione nell'interfaccia di un pulsante per l'annullamento della singola modifica. Questo eviterebbe all'utente di ricominciare il processo dall'inizio in caso di errori, oltre a poterlo utilizzare per ottenere un'anteprima dell'applicazione della modifica. Altre possibili estensioni includono l'implementazione degli altri tipi di modifiche testuali: *join*, *split* e *renumbering*. In Simplex esistono già operazioni simili per alcuni di questi: *downgrade* e *upgrade* nell'editor principale per *join* e *split*, mentre il *renumbering* trova la sua controparte nel *numberingService*. Pertanto, l'implementazione di questi nuovi tipi di modifiche non solo è tecnicamente possibile, ma rappresenterebbe un'importante novità rispetto al passato.

Lo sviluppo del ConsolidationViewer è stato realizzato partendo dall'assunzione che i documenti modificanti contenessero una sola sorgente, un'unica destinazione ed eventualmente un tag *new*. Tra le possibili estensioni future, vi è la possibilità di considerare più sorgenti o destinazioni, nonché l'introduzione del supporto per il tag *old*, che indica il testo precedente da sostituire o eliminare. Sebbene il tag *old* non sia utilizzato da tutte le legislazioni, risulta comunque essere abbastanza diffuso.
In un futuro prossimo, l'interazione con altri software potrebbe consentire di ricevere i documenti tramite API anziché limitarsi alla selezione dal database. Sarà quindi necessario prendere in considerazione questa possibilità durante la fase di selezione. Infine, una possibile implementazione futura potrebbe essere l'introduzione della funzionalità che permetta di riprendere il processo di consolidamento, superando il binarismo "finito-annullato". In questo modo l'utente potrebbe "mettere in pausa" il processo e riprenderlo successivamente, senza dover riapplicare le modifiche già eseguite. Una possibile soluzione potrebbe consistere nella comparazione tra le modifiche attive del modificante e le modifiche passive già presenti nel documento modificato. Se le modifiche passive risultano inferiori in numero rispetto a quelle attive, sarà evidente che non tutte le modifiche sono state applicate, consentendo all'utente di riprendere il processo da dove si è interrotto. Nonostante il sistema offra margini di estensione e miglioramento, il risultato raggiunto in questa prima versione soddisfa pienamente gli obiettivi prefissati, ponendo solide fondamenta per sviluppi futuri.

Riferimenti bibliografici

[Bo06] Bowman, Geoffrey (2006) The art of legislative drafting. Amicus Curiae, 2006 (64). pp. 2-9. https://sas-space.sas.ac.uk/136/1/BowmanGeoffreyIssue064.pdf

[D'A24] Andrea D'Arpa. SIMPLEX Streamlining Legal Document Marking Software. Tesi di laurea Magistrale CdS informatica, Università di Bologna, 2024. data di discussione 14 marzo 2024.

[Po24] Mattia Poggioli. Gestione di eventi utente in un editor di documenti legali basato su Akoma Ntoso. Tesi di laurea Triennale CdS informatica, Università di Bologna, 2024. https://amslaurea.unibo.it/30983/

[Pa23] Monica Palmirani. Modelling the legislative lifecycle. Slide del corso di Informatica Giuridica, CIRSFID – University of Bologna, Law Faculty. Anno accademico 2023/2024.

[Eur17] EUR-Lex. Consolidamento. 2017. https://eur-lex.europa.eu/legal-content/IT/TXT/?uri=LEGISSUM:consolidation

[VP18] Fabio Vitali, Monica Palmirani. Akoma Ntoso Version 1.0 Part 1: XML Vocabulary. 08 Maggio 2018. OASIS Standard. https://docs.oasis-open.org/legaldocml/akn-core/v1.0/cos01/part1-vocabulary/akn-core-v1.0-cos01-part1-vocabulary.html

[CA08] Giuseppe Ciavarini Azzi (2008) Consolidamento dei testi legislativi. https://www.dizie.eu/dizionario/consolidamento-dei-testi-legislativi/

[eX18] eXist Solutions (2018). eXist-db Homepage. https://exist-db.org/exist/apps/homepage/index.html

[Go24] Google (2024). Dependency injection in Angular. https://angular.dev/guide/di/creating-injectable-service

[Co24] Coco Poley. What is WYSIWYG? Definition, Meaning and Key Features. 08 Novembre 2024. TinyMCE. https://www.tiny.cloud/blog/wysiwyg/

[La23] Tamás Lattmann. The EU legislative procedure. 16 Ottobre 2023. https://c4ep.eu/the-eu-legislative-procedure-trying-to-strike-a-delicate-balance-between-sovereignty-of-member-states-interests-of-the-whole-community-while-giving-weight-to-the-representatives-of-the-eur/

[CD24] Camera dei Deputati. La Costituzione della Repubblica Italiana. Ultima visita: 1 Dicembre 2024. https://legislature.camera.it/cost_reg_funz/671/672/documentotesto.asp

[DS24] Department of the Senate, Australia. Passage of legislation. Ultima visita: 2 Dicembre 2024. https://senate.gov.au/passage-of-legislation/index.html

[AG24] Australian Government, Federal Register of Legislation. Ultima visita: 2 dicembre 2024 https://www.legislation.gov.au/help-and-resources/understanding-legislation/glossary

[MM96] Maurizio Malo. I testi unici. 1996 https://www.osservatoriosullefonti.it/archivi/archivio-volumi-osservatorio/osservatorio-1996/8-05-maurizio-malo/file

[OLRC24] Office of the Law Revision Counsel. Positive Law Codification. Ultima visita: 3 dicembre 2024 https://uscode.house.gov/codification/legislation.shtml

[SR01] Senato della Repubblica. Deliberazione 25 ottobre 2001. 2001 https://www.senato.it/leg/norme/regolamento/delibere/urn_nir_senato.repubblica_deliberazione_2001-10-25;nir-2001_ottobre_25_deliberazione@originale/delibera.html