ALMA MATER STUDIORUM • UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

EditTrails: un sistema per la generazione e visualizzazione di modifiche a documenti strutturati

Relatore: Chiar.mo
Prof. Fabio Vitali

Presentata da:
Giorgia Ghermandi

II Sessione

Anno Accademico 2018-2019

Indice

Indice delle illustrazioni

Capitolo 1

Introduzione

Lo scopo del progetto è realizzare un sistema per tracciare e visualizzare le modifiche apportate ad un documento, in modo da rendere comprensibile il loro significato all'utente, attraverso una classificazione delle operazioni effettuate e un'elaborazione delle informazioni relative ad esse.

I problemi da risolvere per ottenere un sistema di questo tipo riguardano tre aspetti: rilevare le modifiche, ricavare informazioni su di esse, e implementare delle modalità di visualizzazione che permettano all'utente di accedere facilmente a queste informazioni.

È necessario prima di tutto stabilire quale sia la struttura ideale per rappresentare le modifiche. Il modo più semplice sarebbe infatti quello di descrivere ciascuna di esse come una serie di operazioni atomiche di aggiunta e cancellazione di contenuto, senza nessuna elaborazione ulteriore. Questo però porta ad una visualizzazione difficile da interpretare per l'utente e povera di informazioni, specialmente se l'azione originaria è più complessa di una sola cancellazione o aggiunta di testo.

Un esempio di una modifica di questo tipo è lo spostamento di una frase in una zona diversa del documento. Questa operazione si compone di due parti: prima il contenuto interessato viene rimosso dal testo, poi viene aggiunto in una nuova posizione.

Testo originario:	*Testo modificato*:
Esempio di modifica di un documento. Questa frase sarà spostata. Verranno generate due modifiche.	Esempio di modifica di un documento. ~~Questa frase sarà spostata~~. Verranno generate due modifiche. **Questa frase sarà spostata**.

Fig. 1 – un esempio di spostamento (MOVE)

Nell'esempio precedente, il testo cancellato è barrato, mentre quello aggiunto è in grassetto. Visualizzando queste due modifiche senza nessun tipo di elaborazione, il risultato sarebbe simile a questo:

Rimosso: Questa frase sarà spostata.

Aggiunto: Questa frase sarà spostata.

Fig. 2 – Esempio di visualizzazione delle modifiche relative ad uno spostamento

Il legame tra le due modifiche, evidente a chi ha effettuato l'operazione di spostamento, non viene espresso in nessun modo. L'utente potrebbe comunque essere in grado di individuarlo grazie alla vicinanza delle due descrizioni e al fatto che il loro contenuto sia identico, ma riuscirci diventerebbe più difficile con l'allungarsi della lista delle modifiche durante l'evoluzione del documento. In ogni caso, è una visualizzazione ridondante e difficile da leggere.

Per quanto riguarda il problema dell'individuazione delle modifiche, esistono già algoritmi di diffing che si occupano di rilevare i cambiamenti avvenuti tra due versioni dello stesso testo. Tuttavia, il confronto che effettuano non può tenere in considerazione l'ordine

temporale in cui sono avvenuti, e questo può portare ad una perdita di informazioni riguardo alle operazioni effettivamente svolte dall'utente.
Riprendendo l'esempio precedente, si potrebbe modificare ulteriormente il documento eliminando la prima frase, in questo modo:

Testo originario:	*Testo modificato*:
Esempio di modifica di un documento. Questa frase sarà spostata. Verranno generate due modifiche.	~~Esempio di modifica di un documento. Questa frase sarà spostata~~. Verranno generate due modifiche. **Questa frase sarà spostata**.

Fig. 3 – Esempio di cancellazione e spostamento adiacenti

La lista delle modifiche individuata tramite algoritmo di diffing, a causa della vicinanza delle due cancellazioni, assomiglierebbe a questa:

Rimosso: Esempio di modifica di un documento. Questa frase sarà spostata.

Aggiunto: Questa frase sarà spostata.

Fig. 4 – Esempio di visualizzazione di cancellazione e spostamento adiacenti

Risulta difficile per l'utente capire che le operazioni effettuate sono una cancellazione e uno spostamento, e non una cancellazione e un'aggiunta di testo. È quindi necessario tenere traccia anche dell'ordine temporale in cui avvengono le modifiche, oltre che della loro vicinanza all'interno del testo.

Una volta trovata una soluzione che permetta di rilevare i cambiamenti nel giusto ordine ed elaborare correttamente le informazioni che li riguardano, è necessario chiedersi quale sia il miglior modo di visualizzare queste informazioni.

Gli strumenti esaminati nella prima parte del capitolo uno offrono una grande varietà di metodi di visualizzazione delle modifiche. È evidente che una soluzione univoca non esista, ma che in generale è preferibile offrire sia delle visualizzazioni dettagliate, sia una visione d'insieme che permetta di trovare la modifica cercata senza dover scorrere l'intero documento.

Ho analizzato alcune tecnologie già esistenti che si occupano di uno o più aspetti riguardanti l'individuazione, analisi e visualizzazione delle modifiche sui documenti. Un loro difetto comune è quello di raccogliere informazioni incomplete o che riguardano solo un determinato aspetto del cambiamento trovato.

Ad esempio, nel caso dei sistemi per la stesura collaborativa di documenti analizzati, lo scopo principale della visualizzazione delle modifiche è quello di mostrare il contributo dei vari autori, e non la natura specifica dei cambiamenti effettuati o il loro ordine temporale. Di conseguenza, non vengono rilevate informazioni come la data e l'orario in cui sono state applicate le modifiche, fondamentali per determinare il loro ordine.

In molti degli editor di testo esaminati non è presente una visione d'insieme delle modifiche, ma solo la visualizzazione, applicata direttamente al testo, di quelle che separano la versione precedente da quella corrente. Quelli che invece prevedono la compilazione graduale di una lista delle operazioni effettuate sul testo, non la conservano in modo permanente.

Anche nel caso di strumenti che riescono a ottenere un'analisi più approfondita, questa è rivolta ad un determinato tipo di documenti, o considera solo alcune categorie di modifiche. Inoltre, tutte le tecnologie elencate prestano maggiore attenzione all'analisi di versioni diverse del documento che a quella delle singole modifiche che le differenziano. Offrono quindi visualizzazioni complessive abbastanza efficaci, ma che non permettono all'utente di

ricevere dettagli su una sola modifica. Queste visualizzazioni sono spesso ricavate attraverso algoritmi di diffing, che possono portare alle visualizzazioni incomplete analizzate in precedenza.

Inoltre, proprio come negli esempi già visti, la maggior parte di questi strumenti non cerca una correlazione tra le operazioni atomiche che ha individuato, impedendo di identificare correttamente l'azione originaria a cui corrispondono.

L'analisi che ha portato a queste conclusioni è contenuta nella prima parte del capitolo due, mentre la seconda parte è dedicata alle tecnologie che ho utilizzato per realizzare il progetto. Ciascuna di esse si occupa di un ambito specifico del problema.

La prima è l'editor di testo SSE [CDV18], creato da ricercatori dell'Università di Bologna in collaborazione con Alstom Ferroviaria. Oltre a facilitare la visualizzazione delle versioni passate dei documenti, permette di rilevare le modifiche effettuate dall'utente appena queste vengono completate. Le informazioni ricavate in questo modo sono di basso livello, ma conservano l'ordine temporale delle modifiche, permettendo di evitare possibili imprecisioni dovute all'utilizzo di algoritmi di diffing.

Per ottenere una correlazione tra le modifiche e ricavare maggiori informazioni su di esse, viene applicata una ricombinazione secondo il modello a tre livelli ideato da Vitali [DSV19]. Il livello più basso esprime le operazioni atomiche avvenute sul documento, mentre quelli successivi aggiungono informazioni su di esse e sul loro collegamento. Per applicare il modello ad una lista di modifiche, ho usato la libreria 3Diff di Spinaci [SPI19].

Infine, per la rappresentazione delle modifiche, ho usato la libreria Differ implementata da Dondi [DON19] che permette di generare tre tipi diversi di visualizzazione: Word style, Wiki style, e Git style.

Ho utilizzato questi strumenti per realizzare il progetto, in modo da offrire una possibile soluzione che si occupi di tutti gli aspetti del problema. A partire dalle operazioni atomiche ricavate grazie agli eventi dell'editor e raggruppate dall'algoritmo, crea la visualizzazione della lista delle modifiche. Al contrario di quello che succede per gli editor esaminati nel

capitolo uno, questa lista descrive l'intera evoluzione del documento. Inoltre, mantiene la struttura a tre livelli, permettendo all'utente di espanderli e gestire così il grado di dettaglio in cui visualizzare ciascuna modifica.

Le informazioni fornite in ciascun livello sono determinate a partire dal contenuto delle operazioni e dalla descrizione del tipo di azione effettuata. Mentre il primo livello visibile presenta delle informazioni generali per capire in cosa consista la modifica, i due successivi entrano maggiormente nel dettaglio, fino a presentare le operazioni atomiche stesse. È inoltre possibile ottenere informazioni aggiuntive cliccando su un'apposita icona accanto ad ogni modifica. Infine, è possibile visualizzare l'area del documento in cui è avvenuta una specifica modifica cliccando su uno dei tre livelli che compongono la sua visualizzazione.

Queste funzionalità del progetto verranno descritte nel capitolo due. Il capitolo tre, invece, tratterà dell'architettura di EditTrails, descrivendo separatamente le parti client-side e server-side, e soffermandosi sui dettagli del procedimento seguito per generare le modifiche e le visualizzazioni.

Capitolo 2

Visualizzazione delle modifiche testuali

Esiste la necessità di trovare metodi efficaci per rilevare le modifiche, ricavare informazioni su di esse, e implementare visualizzazioni che permettano all'utente di accedere a queste informazioni. Esistono già strumenti che si concentrano su uno o più di questi aspetti. In questo capitolo ho esaminato quelli che ho trovato più interessanti dal punto di vista delle funzionalità offerte.

La prima parte è dedicata ad una serie di ricerche effettuate nell'ambito della stesura collaborativa di documenti. Il loro scopo è facilitare il rilevamento e la visualizzazione delle modifiche effettuate dai vari autori, permettendo così di quantificare meglio il contributo di ciascuno e risolvere eventuali conflitti tra le diverse versioni prodotte.

Ho poi esaminato alcuni strumenti di questo tipo dedicati a formati e tipologie specifiche di documenti. Nonostante l'ambito di cui si occupano sia più limitato rispetto a quello che tenta di coprire il progetto, offrono modalità interessanti per la visualizzazione o la classificazione delle modifiche trovate. Alcuni permettono di raggruppare e classificare le operazioni atomiche per descrivere meglio le modifiche avvenute.

Mi sono concentrata soprattutto sulle modalità di visualizzazione delle modifiche. La maggior parte degli strumenti esaminati offre una visione d'insieme, oltre ad evidenziare sul testo le operazioni atomiche. Alcuni permettono all'utente di personalizzare questa rappresentazione in base a determinati parametri, come ad esempio il livello di granularità o la posizione all'interno del documento.

Infine, ho effettuato una panoramica dei principali editor di testo che offrono funzionalità di tracciamento e visualizzazione delle modifiche, riassumendo i pregi e difetti di ciascuna.
L'ultima parte del capitolo è dedicata alla descrizione delle tecnologie che ho utilizzato per implementare il progetto, e mostra come ciascuna di esse risponda ad uno degli ambiti già esaminati del problema.

2.1 Stesura collaborativa di documenti

Gli strumenti relativi alla stesura collaborativa dei documenti spesso offrono funzionalità che permettono di tracciare le modifiche e risolvere i conflitti che possono insorgere tra quelle effettuate da utenti diversi. Sfruttando le informazioni raccolte su di esse, è possibile creare visualizzazioni che aiutino l'utente ad avere una visione d'insieme del processo di modifica del documento.
In ordine di rilevanza, ho analizzato l'editor di testo asincrono creato da Papadopoulou, Ignat, Oster e Norrie, il loro sistema per la visualizzazione dei cambiamenti apportati ad una pagina web, l'interfaccia di Alsubhi e Munson, la rappresentazione delle versioni di Wikipedia, e l'Edit History Analyzer di Fong e Biuk-Aghai. Ho evidenziato come questi strumenti riescano a fornire sia una modalità di visualizzazione complessiva che una rappresentazione relativa al testo, ma non forniscano informazioni dettagliate sulle singole modifiche.
Papadopoulou, Ignat, Oster e Norrie [PIO06] hanno osservato che aumentare la consapevolezza di ciascun co-autore delle modifiche effettuate dagli altri lo aiuta a comprendere meglio come il documento stia evolvendo e migliora la cooperazione. Per questo motivo hanno implementato un editor di testo asincrono che traccia le modifiche avvenute e permette all'utente di selezionare quali visualizzare.

Il documento viene rappresentato attraverso un modello gerarchico ad albero, dove ciascun livello corrisponde a un grado di granularità. Anche se è possibile definirne un numero maggiore, quelli utilizzati nell'editor sono cinque: documento, paragrafo, frase, parola e carattere. La visualizzazione dipende dai livelli scelti dall'utente. Ad esempio, quello relativo alle parole deve essere selezionato per mostrare correzioni dello spelling dei termini.

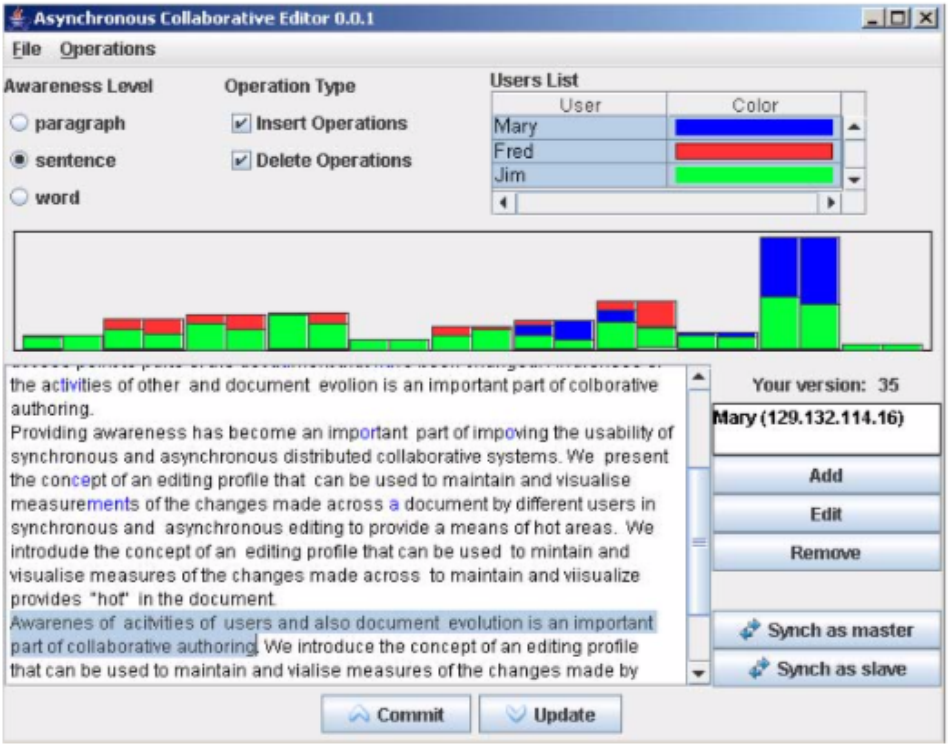


Fig. 3. Awareness enhanced GUI

Fig. 5 – Schermata di visualizzazione delle modifiche, figura di [PIO06]

Le modifiche vengono visualizzate sia all'interno del testo, sia attraverso un istogramma riassuntivo. In questo caso, ciascuna barra del grafico rappresenta una sezione del documento. L'altezza è determinata dalla quantità di cambiamenti avvenuti nella porzione

di testo corrispondente rispetto al totale. Le fasce orizzontali quantificano il contributo di ciascun utente, a cui viene assegnato un colore identificativo.
È inoltre possibile selezionare quali operazioni atomiche visualizzare. Non viene dunque effettuata un'analisi più approfondita del tipo di modifica effettuata, anche se la divisione in livelli applica una prima distinzione tra di esse.
L'istogramma riesce nell'obiettivo di informare l'utente della distribuzione delle modifiche all'interno del documento e del contributo degli altri autori, ma non permette di ottenere informazioni su una singola modifica.
Gli stessi ricercatori hanno inoltre implementato un sistema [PIO08] che permette ai co-autori di siti web di tenere traccia di modifiche avvenute sia in pagine specifiche, che in quelle ad esse collegate. Lo scopo è evitare che la modifica di una pagina renda imprecise le informazioni semantiche contenute nelle altre, nel caso di frammenti ripetuti o che dipendono gli uni dagli altri.
Un processo in background controlla periodicamente le pagine da monitorare e le confronta tramite algoritmo di diffing alla loro versione precedente. Le modifiche così trovate vengono salvate come una lista di attributi, che comprendono anche l'operazione meccanica, il livello del nodo interessato dal cambiamento e la posizione. Inoltre, viene misurato l'effetto che ha avuto il cambiamento sulla pagina in relazione ad una serie di parametri, come ad esempio il numero di caratteri e frasi inseriti o cancellati dalla modifica. La scelta dei parametri è influenzata dalle preferenze espresse dall'utente.
Se sono presenti link, le modifiche relative alle pagine collegate vengono recuperate allo stesso modo. Sulla base delle informazioni raccolte, viene costruita una visualizzazione composta da tre parti. Quella centrale presenta il testo, in cui vengono evidenziati i caratteri modificati, mentre le altre due sono barre laterali suddivise in sezioni più piccole. In quella a destra, rappresentano i paragrafi presenti in tutto il documento, mentre a sinistra solo le frasi visibili. Le zone rosse indicano i cambiamenti avvenuti nella zona del testo corrispondente, e la loro grandezza dipende da quante parole sono state modificate.

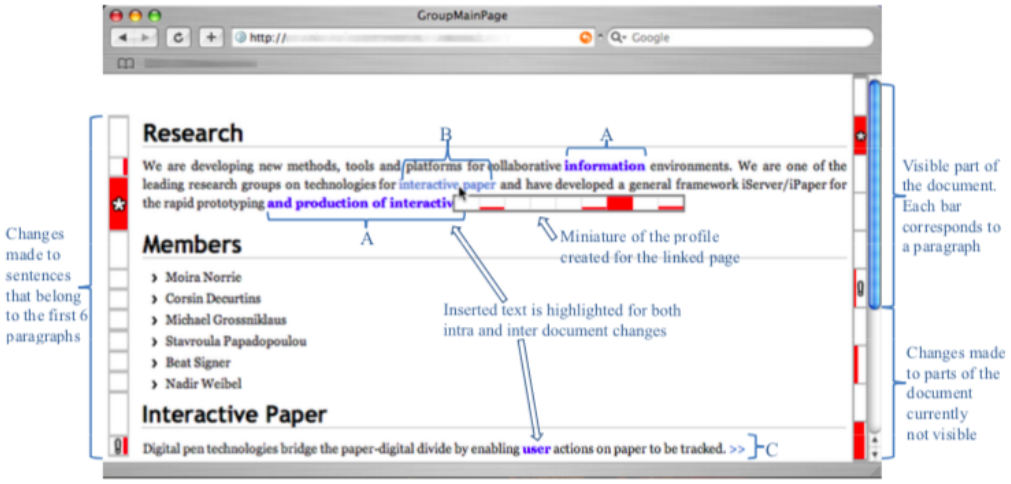


Fig. 3. Visualisation of intra– and inter–document changes made to the main page

Fig. 6 – Schermata di visualizzazione delle modifiche, figura di [PIO08]

Le sezioni in cui è presente un collegamento ad una pagina che è stata modificata sono indicate con un asterisco. Inoltre, posizionando il cursore su un link, appare una barra orizzontale simile a quelle laterali, relativa però ai cambiamenti avvenuti nella pagina collegata.

Nonostante le due visualizzazioni riassuntive permettano di capire chiaramente quali zone della pagina siano state modificate di più rispetto all'ultima versione, non offrono nessuna informazione riassuntiva su di esse. È necessario cercarle manualmente nel testo, operazione che può richiedere molto tempo.

Lo studio effettuato da Alsubhi e Munson [AM16] si concentra sulla creazione di un'interfaccia per la visualizzazione ed il merging delle modifiche effettuate su un documento dai vari utenti che collaborano alla sua stesura. Si propongono in particolare di offrire un sistema di filtraggio delle modifiche in base alla categoria di appartenenza. Definiscono quattro tipi di operazioni: cambi di font, correzioni di spelling, modifiche alla

grammatica e al contenuto. Non implementano un modo per effettuare questa classificazione, ma ipotizzano che esista già uno strumento per ottenerla allo scopo di mostrare le funzionalità dell'interfaccia.

Per generare la visualizzazione, vengono caricati quattro documenti: la versione originaria del testo, due versioni modificate in modo indipendente tra cui effettuare il merging, e la lista delle modifiche avvenute in entrambe, già classificate nelle quattro categorie.

La schermata è divisa verticalmente in due pannelli. Quella sinistra mostra il testo ottenuto dal merging delle versioni modificate. Vengono evidenziate le modifiche conflittuali, che l'utente può selezionare per ottenere maggiori informazioni e scegliere quale verrà conservata. Il pannello di destra presenta le tre versioni del documento fornite. Quando si seleziona una modifica nel testo unificato, il contenuto corrispondente nel testo originale e nella versione a cui appartiene vengono evidenziati. Inoltre, è possibile attivare o disattivare la visualizzazione dei quattro tipi di modifiche.

Questa classificazione delle operazioni facilita la comprensione dei cambiamenti avvenuti, ma come per il sistema precedente non esiste una visualizzazione riassuntiva delle modifiche che permetta di individuarle con precisione senza scorrere i documenti interessati.

Per valutare l'efficacia dell'interfaccia, gli autori hanno sottoposto dieci volontari ad un test in cui avrebbero dovuto utilizzarla per svolgere una lista di compiti su due insiemi di documenti. Ciascun insieme era composto dalla versione originaria, a cui erano stati applicati degli errori di vario genere, da altre due versioni che li risolvevano in modo diverso, e da una lista che riassumeva le modifiche che differenziavano ciascuna versione dall'originale. I compiti consistevano nell'individuare uno specifico tipo di modifica basato sulle quattro classificazioni introdotte, per poi accettare o rifiutare i cambiamenti individuati.

I risultati del test hanno portato gli autori a concludere che la classificazione dei cambiamenti migliora la capacità degli utenti di comprendere la natura delle modifiche effettuate, e facilita di conseguenza le operazioni di merging.

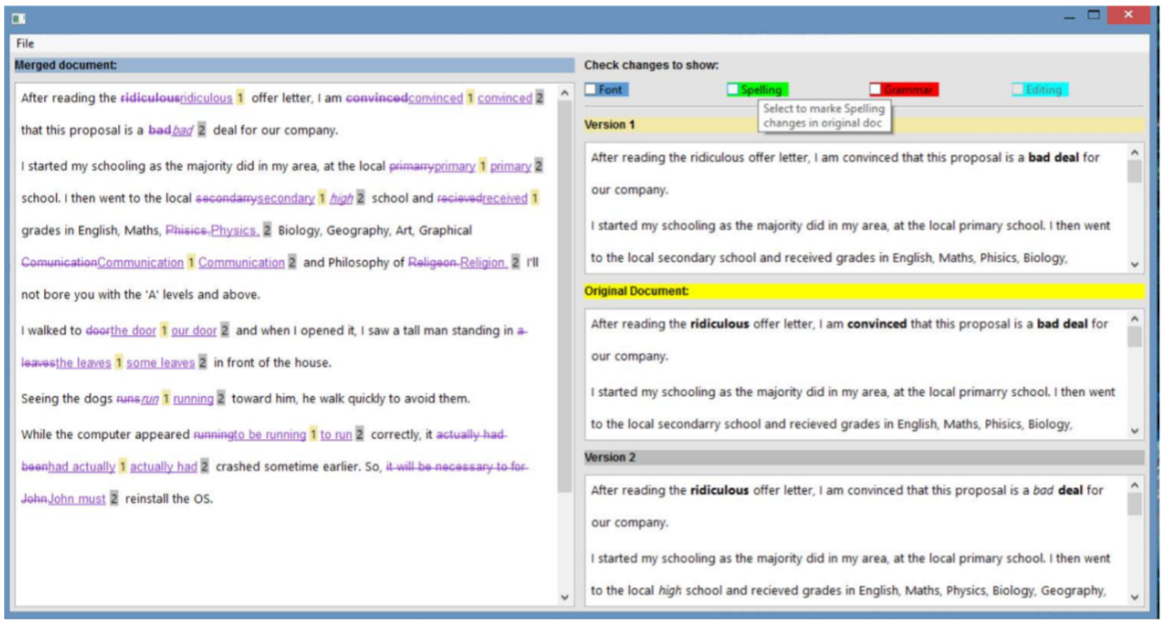


Figure 1 The main screen of the proposed interface.

Fig. 7 – Interfaccia per il merging dei documenti, figura di [AM16]

Ho poi esaminato il sistema di registrazione delle modifiche utilizzato da Wikipedia [WIK19]. Essendo basata su un sistema di scrittura collaborativa, deve affrontare gli stessi problemi già trattati riguardanti la registrazione e visualizzazione delle modifiche. In particolare, per ogni pagina offre una lista riassuntiva delle sue versioni, elemento che mancava in tutti gli strumenti esaminati fino ad ora.

Per ciascuna versione, vengono presentate informazioni come la data e lo username dell'autore. Per calcolare le modifiche che separano due versioni è necessario confrontarle. In questo caso, i paragrafi modificati vengono visualizzati uno accanto all'altro, con il contenuto della modifica evidenziato e l'operazione meccanica corrispondente indicata da un simbolo.

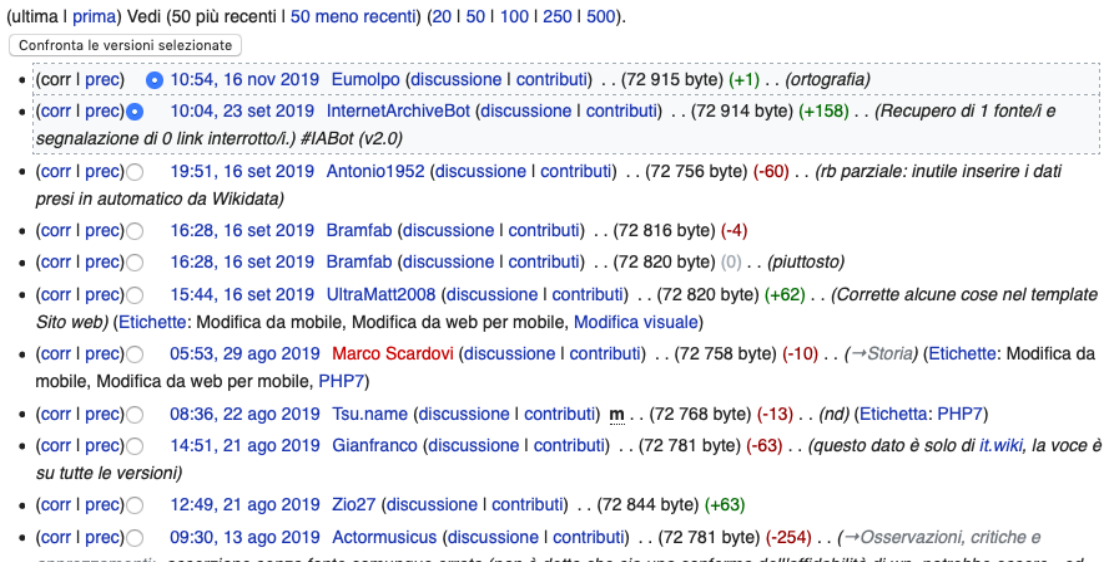


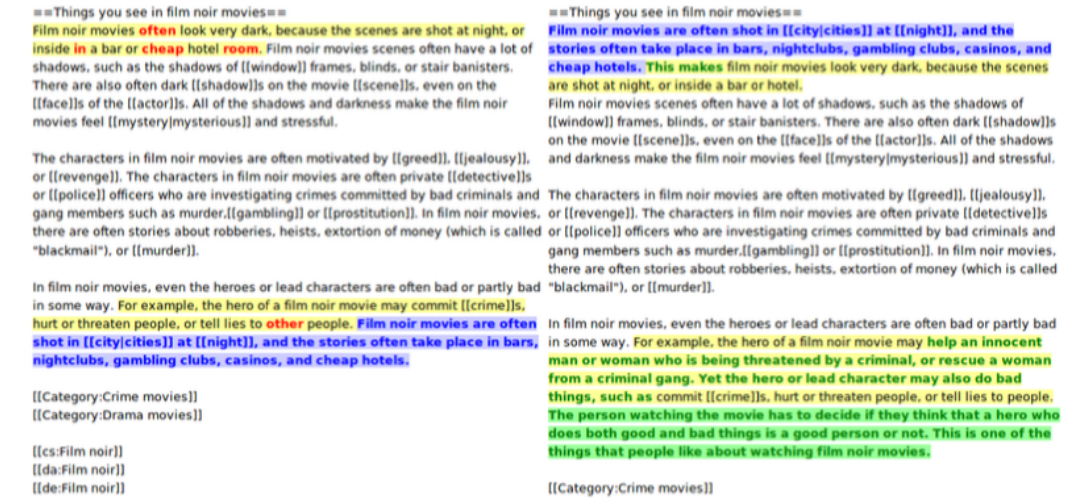
Fig.8 – Lista delle versioni di una pagina di Wikipedia

Da sole, queste due modalità di visualizzazione delle versioni rendono difficile cercare modifiche specifiche. Inoltre, non viene effettuato nessun tipo di analisi ulteriore sui tipi di operazioni effettuate dagli utenti.

Fig.9 – Visualizzazione delle modifiche di una pagina di Wikipedia

Fong e Biuk-Aghai [FB10] hanno ideato l'Edit History Analyzer allo scopo di offrire una versione migliorata del confronto tra le versioni passate delle pagine di Wikipedia. In particolare, volevano che rendesse più comprensibile il tipo di operazione corrispondente a ciascuna modifica.

Per ottenere una classificazione più dettagliata dei cambiamenti, hanno progettato il funzionamento di un componente, chiamato Text Difference Engine, in grado di trovare le modifiche tramite algoritmi di diffing. A ciascun elemento della lista risultante sarebbe poi stata assegnata un'operazione di base tra le quattro definite: cancellazione, inserimento, spostamento e sostituzione. L'Action Categorizer si occupa poi di esaminare ulteriormente le modifiche in base all'operazione e al contenuto, allo scopo di identificare l'azione di livello più alto corrispondente. Questa viene selezionata sulla base di regole di riconoscimento predefinite.



==Things you see in film noir movies==
Film noir movies often look very dark, because the scenes are shot at night, or inside in a bar or cheap hotel room. Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may commit [[crime]]s, hurt or threaten people, or tell lies to other people. Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels.

[[Category:Crime movies]]
[[Category:Drama movies]]

[[cs:Film noir]]
[[da:Film noir]]
[[de:Film noir]]

==Things you see in film noir movies==
Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels. This makes film noir movies look very dark, because the scenes are shot at night, or inside a bar or hotel.
Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may help an innocent man or woman who is being threatened by a criminal, or rescue a woman from a criminal gang. Yet the hero or lead character may also do bad things, such as commit [[crime]]s, hurt or threaten people, or tell lies to people. The person watching the movie has to decide if they think that a hero who does both good and bad things is a good person or not. This is one of the things that people like about watching film noir movies.

[[Category:Crime movies]]

Figure 5. Difference page of article "Film noir" produced by our edit history analyzer (background colours: yellow – sentence changed, blue – sentence moved, green – sentence added; text colours: red – word deleted; blue – word moved, green – word inserted)

Fig.10 – Visualizzazione creata dall'Edit History Analyzer, figura di [FB10]

La visualizzazione che hanno realizzato mostra le due versioni del testo affiancate, con le quattro operazioni di base visualizzate in modo diverso: rosso per le cancellazioni, verde

per gli inserimenti, blu per il contenuto spostato e giallo per quello cambiato. Per le parole modificate viene evidenziato solo il testo, per le frasi anche lo sfondo.
Questa visualizzazione si rivela più efficace di quella normalmente usata da Wikipedia. È però dipendente da quel modello, e non applicabile ad altri ambiti.

2.2 Rilevare e visualizzare modifiche

Anche al di fuori della scrittura collaborativa di documenti, esistono molti ambiti che beneficiano di un sistema di confronto tra le versioni e visualizzazione delle modifiche rilevate. Spesso i documenti utilizzati hanno un formato specifico, ma alcuni spunti sono comunque generalizzabili per essere applicati ad altri tipi di contenuto.
Gli strumenti che ho esaminato, in ordine di rilevanza, sono: FuzzyTree, il sistema di Kehrer, Kelter e Taentzer, e due software per il merging di file di codice, ovvero WinMerge e DiffMerge. Ciascuno di essi offre funzionalità che risulterebbero utili anche in un ambito più generale, come la ricombinazione delle operazioni di base e varie modalità di visualizzazione, parzialmente adattabili in base alle esigenze dell'utente. Queste però sono state implementate per formati specifici di documenti, e non sono direttamente applicabili ad ambiti più generali senza un'ulteriore elaborazione.
FuzzyTree è un sistema ideato da Chawathe [CHA06], per visualizzare i cambiamenti avvenuti tra versioni diverse di documenti tecnici relativi all'ambito medico, allo scopo di aumentare la quantità di informazioni disponibili all'utente e rendere più semplice la ricerca di modifiche all'interno del testo.
Queste vengono visualizzate tramite una lista strutturata ad albero, dove ogni voce può essere espansa per accedere alle informazioni che contiene. I figli di un nodo possono non essere mostrati completamente, in relazione al loro grado di importanza e al livello di dettaglio selezionato dall'utente.

Nonostante questo modo di visualizzare le modifiche sia più informativo e permetta di navigare facilmente i cambiamenti avvenuti all'interno del documento, può essere applicato solo a documenti XML strutturati in un modo specifico. Alcune delle operazioni stesse, essendo relative alla modifica di label interne al markup, non sono di facile lettura per chi non conosce la struttura utilizzata.

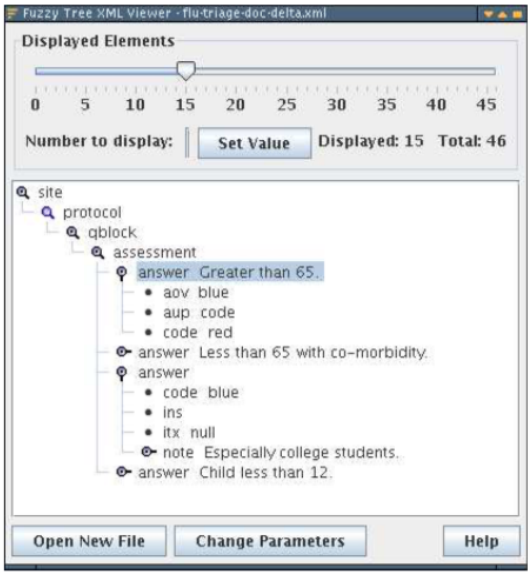


Figure 6. *A screen-shot of the* FuzzyTree *variable-resolution browser operating on the document of Figure 5. The selective display algorithm exposes the modified data (aov, aup, ins, etc.) while hiding other, less important, data.*

Fig.11 – Visualizzazione delle modifiche in FuzzyTree, figura di [CHA06]

Kehrer, Kelter e Taentzer [KKT11] hanno elaborato un sistema per presentare le modifiche applicate ai modelli utilizzati nel software engineering. In particolare, hanno usato l'Eclipse Modeling Framework (EMF) per rappresentare i modelli come grafi modificabili dall'utente.

Nonostante l'ambito a cui è applicato sia ancora più specifico del precedente, e non consideri ad esempio i semplici documenti testuali, ho trovato interessante il modo in cui applica la ricombinazione delle operazioni atomiche. Ogni operazione dell'utente viene espressa tramite una serie di cancellazioni e inserimenti. Questa sequenza può essere anche molto lunga, in quanto alcune delle azioni permesse sui modelli, come la cancellazione di un riferimento, sono composte da più operazioni atomiche rispetto a quelle che compongono le modifiche applicabili ad un documento testuale, come la cancellazione di una frase. È quindi molto più importante ricombinare le modifiche di livello basso in quelle di livello più alto. Queste ultime vengono identificate a partire dalle regole che definiscono le operazioni permesse sul modello, e la conversione avviene in modo automatico grazie alla struttura schematica, quindi facile da generare, delle regole di riconoscimento. Le sequenze di operazioni atomiche vengono confrontate con queste regole per trovare corrispondenze.

Un altro ambito in cui è utile avere un sistema di visualizzazione efficace delle modifiche è quello dei software per il confronto e il merging di file di codice.

Un esempio di questo tipo di software è Winmerge [WIN19]. Infatti, permette di effettuare un confronto tra le versioni che si vogliono unire, realizzato a livello di riga. Nella visualizzazione, le linee modificate hanno uno sfondo di colore diverso: oro per quelle cambiate e aggiunte, arancione per quelle spostate, grigio per quelle eliminate. Nel caso sia attiva l'opzione corrispondente, vengono evidenziate anche differenze a livello di parola o carattere. È inoltre possibile creare dei filtri personalizzati per ignorare determinati tipi di cambiamenti.

Winmerge fornisce diversi strumenti per facilitare l'individuazione delle modifiche e navigare all'interno delle due versioni confrontate. Ad esempio, selezionando una riga modificata è possibile confrontarla con quella corrispondente in modo isolato rispetto al resto del file. Inoltre, appaiono delle icone che permettono di scorrere la lista delle modifiche, evitando così di dover visualizzare anche le porzioni di codice rimaste uguali.

Accanto all'area dove appare il risultato del confronto è presente un pannello che permette di visualizzare una versione riassuntiva della lista delle differenze. È composto da due colonne bianche, corrispondenti alle versioni confrontate, attraversate da linee colorate. Lo spessore, colore e posizione di queste linee rispecchiano le modifiche trovate nella visualizzazione principale. L'area visualizzata ha uno sfondo più scuro e, nel caso in cui l'utente abbia selezionato una modifica, la linea corrispondente viene evidenziata. Nel caso siano presenti degli spostamenti di testo, le linee che corrispondono allo stesso contenuto vengono collegate da un tratto nero. Questa visualizzazione permette di avere una visione d'insieme delle differenze trovate, particolarmente utile in caso di documenti lunghi.

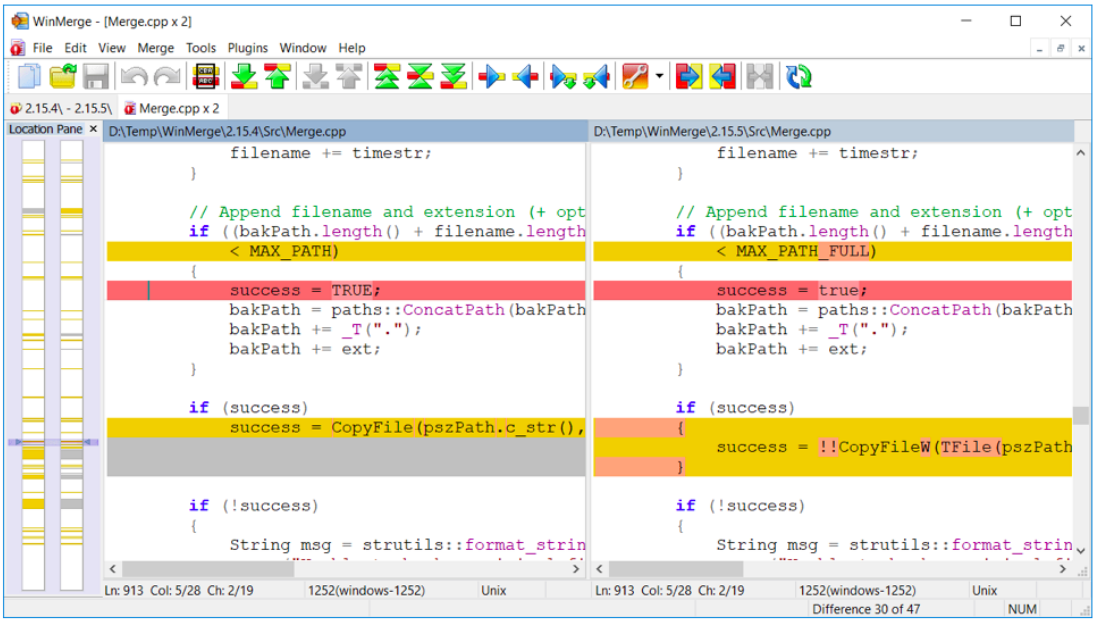


Fig.12 – Schermata di confronto delle versioni in Winmerge, figura di [WI19]

Un altro software con funzionalità simili ma modalità di visualizzazione diverse è Diffmerge [DI19]. Permette sia di visionare il confronto tra le versioni in sola lettura, sia eventualmente di modificare quella più recente. In questo caso, i file vengono nuovamente confrontati dopo ogni modifica, e i cambiamenti appaiono direttamente sul testo. Si può

scegliere tra un allineamento orizzontale o verticale dei documenti, ed è possibile utilizzare dei marker per forzare l'allineamento di righe specifiche. Inoltre, si può decidere di visualizzare tutto il testo, solo le righe modificate, oppure solo le righe modificate insieme ad un contesto di tre righe attorno a ciascuna.

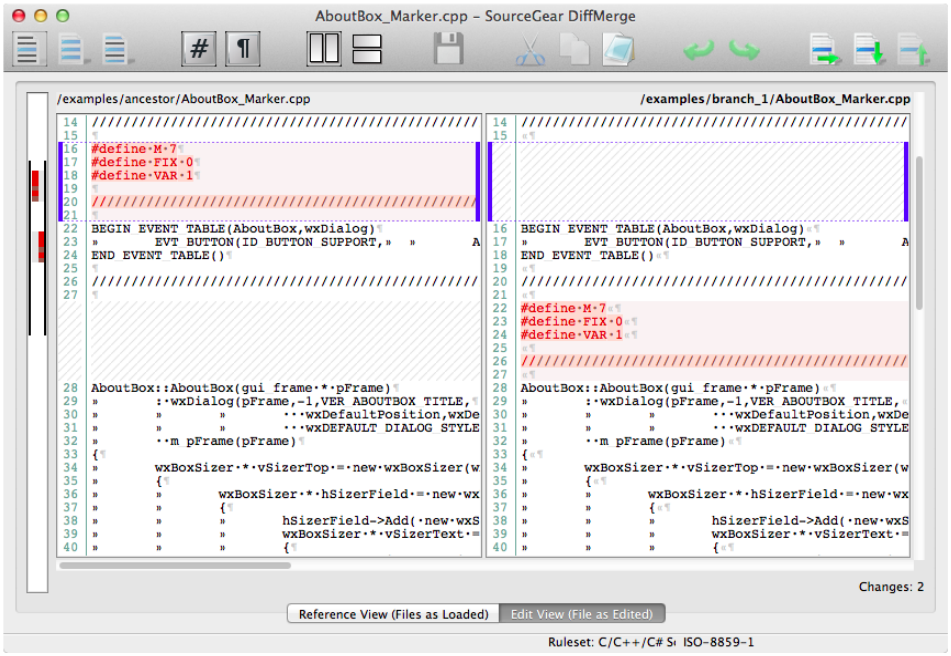


Fig.13 – Schermata di confronto delle versioni in DiffMerge, figura di [DI19]

Accanto all'area principale per il confronto è presente un pannello che rappresenta il file in scala, dove le linee colorate di rosso corrispondono alle modifiche. Può essere utilizzato per scorrere alla posizione voluta del documento. Il contenuto modificato viene evidenziato in rosso anche nel pannello principale, ma non è ulteriormente specificato il tipo di operazione avvenuta. Come per lo strumento precedente, è possibile definire quali tipi di righe e cambiamenti devono essere ignorati durante il confronto. Questi appaiono comunque nella visualizzazione, in un colore più chiaro. Infine, si può specificare se visualizzare le modifiche solo a livello di linea, dove viene evidenziata unicamente la riga modificata, o anche di carattere. In questo caso, sottolinea solo la parte di contenuto della riga diverso. In

generale, le informazioni offerte sono poche e la visualizzazione non aiuta a comprendere in che modo i documenti siano effettivamente cambiati.

2.3 Editor di testo

Anche i principali editor di testo presentano sistemi per rilevare i cambiamenti avvenuti su un documento. Un'analisi approfondita delle loro funzionalità di tracciamento e visualizzazione delle modifiche è stata effettuata da Dondi [DON19].

Esaminando Microsoft Word, Apache Open Office, e Apple Pages, osserviamo che le funzionalità di tracciamento delle modifiche non agiscono in modo automatico, ma devono essere attivate espressamente dall'utente. Inoltre, una volta approvate le modifiche effettuate, i dati che le caratterizzano scompaiono e non è più possibile recuperarli.

In tutti e tre i casi, le modifiche vengono evidenziate sul testo, distinguendo solo le operazioni atomiche. Word e Pages presentano inoltre una colonna riassuntiva delle modifiche, posizionata accanto al testo. Anche in quel caso, vengono registrati solo i cambiamenti non ancora approvati dall'utente.

Un altro editor esaminato da Dondi è Google Docs. Al contrario dei precedenti, esso conserva tutti i cambiamenti apportati al documento, e ad ogni gruppo di modifiche cronologicamente vicine corrisponde una versione. Oltre alla visualizzazione sul testo delle operazioni atomiche, permette di visualizzare una lista riassuntiva delle versioni, ma non delle singole modifiche.

Majer e Rönnau [MR15] hanno osservato che, mentre è possibile scaricare i documenti da Google Drive in formato HTML, non si può ottenere una copia delle modifiche, che devono quindi essere trovate nuovamente dopo il download. Hanno quindi ideato DynTrack, un sistema che permette di scaricare automaticamente le versioni dei documenti e compararle tra loro.

Hanno inoltre implementato due modalità di visualizzazione delle modifiche. La prima le rappresenta in forma di testo, con cancellazioni evidenziate in rosso e aggiunte in verde. La seconda, invece, le presenta in una struttura ad albero.

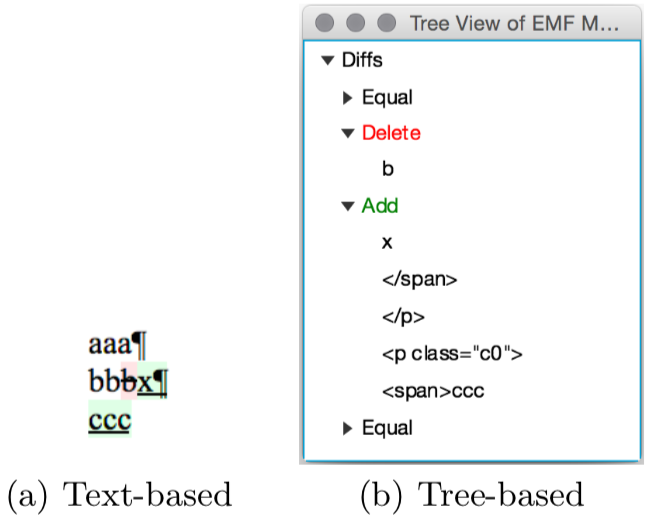


Figure 3: Visualization of Changes

Fig.14 – Visualizzazione delle modifiche in DynTrack, figura di [MR15]

Infine, trattano la possibilità di implementare delle funzioni di filtraggio e aggregazione che facilitino ulteriormente la visualizzazione delle modifiche. Il filtraggio renderebbe possibile selezionare quali visualizzare, sulla base ad esempio della posizione nel documento o al fatto che riguardino o meno il contenuto. L'aggregazione servirebbe ad unire modifiche atomiche in quelle di livello più alto, basandosi sulla posizione o sul loro tipo di operazione.

2.4 Componenti e librerie utilizzate

Come detto nel primo capitolo, per ottenere una rappresentazione accurata delle modifiche è necessario recuperarle nell'ordine e nella forma in cui vengono effettuate. Questo risulta impossibile nel caso in cui vengano confrontate la versione originale e una già modificata del documento, in quanto i cambiamenti verrebbero rilevati in base alla loro posizione nel testo e non ci sarebbe modo di ricostruirne l'ordine temporale. Inoltre, risulterebbe impossibile distinguere modifiche adiacenti che condividono lo stesso tipo di operazione atomica.

Per poter rilevare i cambiamenti mentre vengono applicati, è necessario utilizzare uno strumento che permetta sia di modificare il documento, sia di rilevare che le operazioni siano avvenute, oltre a poter recuperare informazioni basilari su di esse quali la posizione e il contenuto che le riguarda.

A questo scopo, ho usato l'editor di testo Web-based SSE (Smart Structured Editor) [CDV18], creato da ricercatori dell'Università di Bologna con la collaborazione di Alstom Ferroviaria per facilitare la stesura di documenti tecnici. In particolare, risponde all'esigenza di Alstom di gestire la documentazione dei progetti di cui si occupa, che può anche essere composta da migliaia di documenti. SSE facilita questo compito utilizzando un linguaggio di markup appositamente definito, la possibilità di creare e utilizzare template, ovvero frammenti di contenuto che possono essere condivisi da più di un documento, e un sistema per il congelamento dei documenti. Quest'ultimo permette di salvare una versione del testo, che riflette il suo stato al momento del congelamento e non potrà più essere modificata.

All'interno dell'editor, la lista delle versioni del documento viene visualizzata dentro all'apposito tab History. È possibile cliccare su ciascuna di esse per visualizzare il contenuto corrispondente all'interno della schermata principale dell'editor.

Il congelamento di una versione è un'azione che deve essere effettuata dall'utente. Prima di EditTrails, non era ancora presente un sistema per rilevare le modifiche che fosse automatico, e quindi indipendente dalle decisioni dell'utente.

Per realizzare la lista delle modifiche, ho utilizzato principalmente gli eventi dell'editor legati alla modifica del documento, che permettono di individuare soltanto operazioni atomiche.

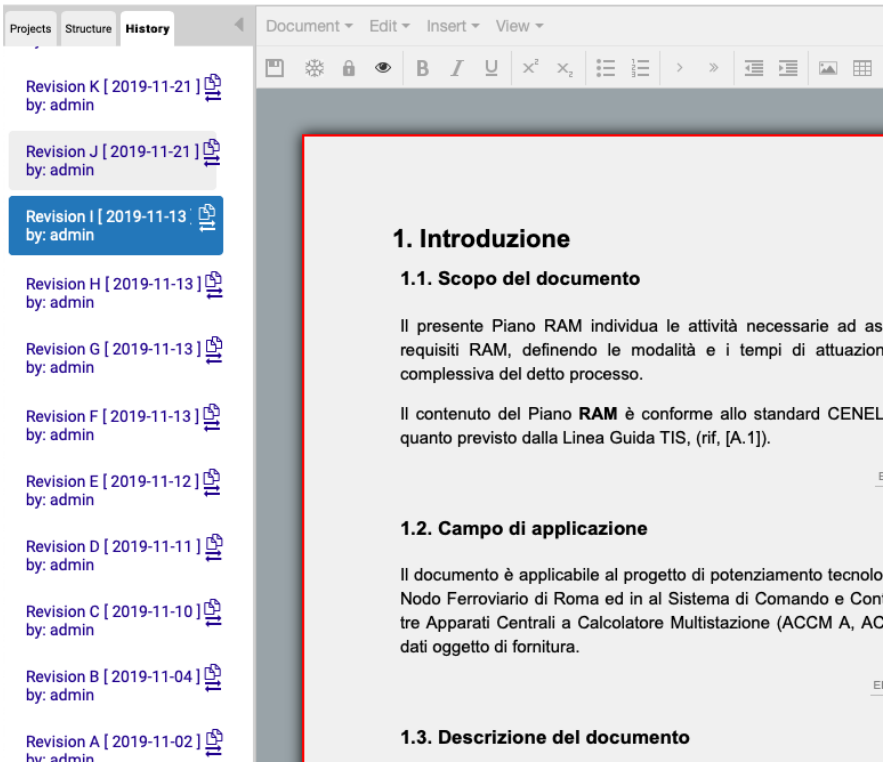


Fig. 15 - Tab History con una versione congelata attiva

Come visto, queste informazioni da sole non sono sufficienti a costruire una visualizzazione delle modifiche esaustiva. Devono quindi essere ulteriormente elaborate in una struttura che permetta di comprenderne meglio il significato.

Questa struttura viene ricavata attraverso un raggruppamento delle modifiche, effettuato grazie alla libreria 3Diff di Spinaci [SPI19] sulla base del modello a tre livelli di Vitali [DSV19], che descriverò nel capitolo tre.

3Diff permette sia di confrontare due testi selezionando uno degli algoritmi di diffing offerti, e ottenere così una lista di modifiche opportunamente raggruppate, sia di elaborare una lista di modifiche già esistente. Ai fini del progetto userò quest'ultima funzionalità, in quanto la lista di modifiche da raggruppare verrà recuperata a partire dagli eventi dell'editor. La lista ricavata grazie alla libreria verrà utilizzata per generare la visualizzazione delle modifiche all'interno del tab History di SSE.

Oltre alla rappresentazione delle modifiche sotto forma di lista, è possibile attivare tre tipi di visualizzazione, implementati dalla libreria Differ di Dondi [DON19]. Ciascuno di essi presenta caratteristiche diverse dagli altri, in modo da permettere all'utente di scegliere il tipo di rappresentazione che più si avvicina alle sue esigenze.

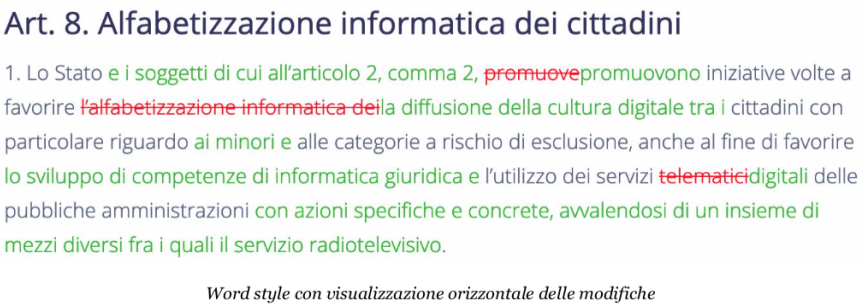


Fig.16 – Visualizzazione Word style, figura di [DON19]

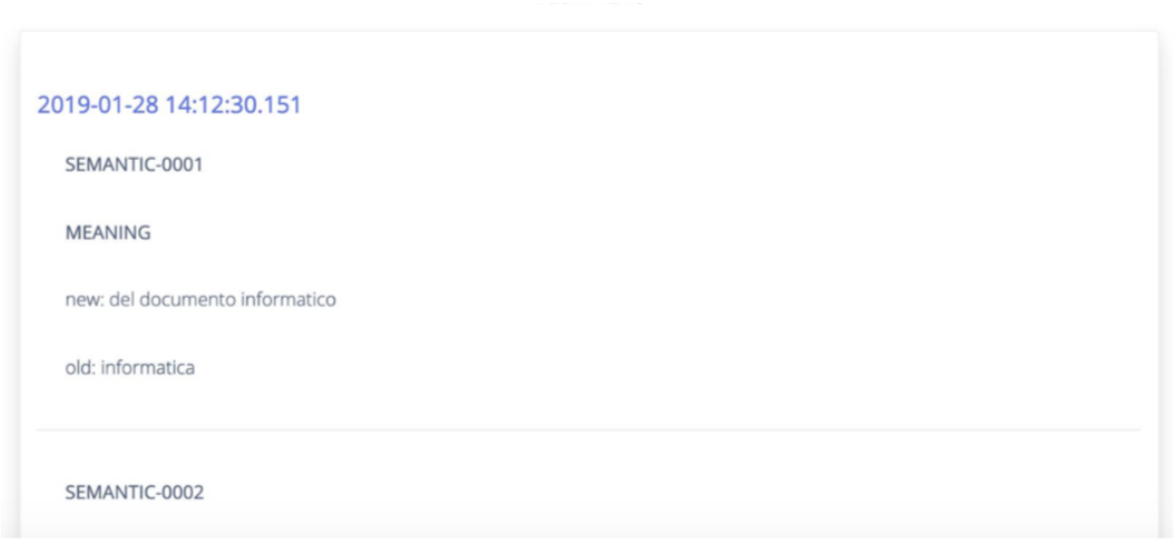
La prima modalità è Word style: presenta il testo in versione integrale, su cui viene evidenziato il contenuto di ciascuna modifica in base all'operazione corrispondente. In particolare, il testo aggiunto è verde, mentre quello cancellato appare barrato e in rosso. Non vengono aggiunti ulteriori dettagli riguardanti il tipo di operazione. Questa visualizzazione è utile per avere una visione d'insieme delle modifiche applicate al testo e

per conoscere il loro ordine all'interno di esso, indipendentemente da quello temporale in cui sono avvenute.

Fig. 17 – Visualizzazione Wiki style, figura di [DON19]

La seconda visualizzazione è Wiki style. A differenza della precedente, esprime l'operazione più complessa di cui fanno parte gli inserimenti e le cancellazioni. Non permette di visualizzare tutto il testo, ma solo il contesto delle modifiche avvenute, rappresentato dal paragrafo di cui fa parte il loro contenuto. Queste porzioni di testo sono divise in due colonne, una dedicata alle cancellazioni, l'altra agli inserimenti. Questo facilita il confronto tra i paragrafi relativi alle operazioni atomiche che fanno parte della stessa modifica.

L'ultima modalità di visualizzazione è Git style. Segue l'ordine cronologico delle modifiche ed esplicita la data e l'ora in cui ciascuna è avvenuta. Viene inoltre espressa l'operazione di alto livello corrispondente, e il contesto della modifica prima e dopo la sua applicazione.



Git style

Fig.18 – Visualizzazione Git style, figura di [DON19]

Capitolo 3

EditTrails

L'analisi svolta nel capitolo due ha mostrato come ancora esista il bisogno di creare un sistema che si occupi sia di rilevare le modifiche, sia di elaborare e visualizzare le informazioni relative ad esse. Infatti, mentre è stata implementata una grande varietà di metodi per rappresentare i cambiamenti di un testo, i primi due aspetti del problema sono spesso trascurati dagli strumenti già esistenti. Ho quindi implementato un sistema in grado di occuparsi anche della generazione ed elaborazione delle modifiche.

3.1 Modello a tre livelli

Come ho accennato nel capitolo due, per ottenere una descrizione più accurata di ciascuna modifica, ho utilizzato la libreria 3Diff, che raggruppa le modifiche nella struttura del modello a tre livelli [DSV19]. Ciascuno dei livelli di cui è composta offre una caratterizzazione sempre più dettagliata delle modifiche. Infatti, il primo è simile all'output di un algoritmo di diffing, mentre quelli successivi sono ottenuti a partire da un raggruppamento delle modifiche del livello precedente. Ciascun livello esprime un proprio insieme di operazioni, che descriverò di seguito.

Il primo è il livello meccanico. Presenta due operazioni: inserimento e cancellazione di stringhe di caratteri, indicate rispettivamente come INS e DEL. Le modifiche a questo

livello descrivono solo l'operazione avvenuta, senza analizzare ulteriormente il contenuto. Questo significa che, ad esempio, un inserimento che riguarda solo il testo e un altro che contiene elementi relativi al markup vengono rappresentati entrambi come operazioni di tipo INS.

```
{
  "id": "mech-00028",
  "op": "DEL",
  "pos": 80,
  "content": "<p id='x45' class='t'>text."
},
{
  "id": "mech-00029",
  "op": "INS",
  "pos": 80,
  "content": "<p class='t' id='x45'>new."
}
```

Fig.19 - esempi di modifiche meccaniche, figura di [DSV19]

Il secondo è il livello strutturale. Ogni modifica a questo livello è composta da una serie di operazioni meccaniche che fanno parte della stessa azione, anche nel caso in cui vengano effettuate in più fasi o riguardino posizione diverse del documento. Sono divise in due tipi: le operazioni sul testo, che vengono svolte all'interno dello stesso nodo di testo, e le operazioni sul markup, che riguardano più nodi e sono sempre considerate atomiche, anche nel caso riguardino più operazioni meccaniche.
Le operazioni sul testo sono ulteriormente distinguibili in tre categorie: quelle avvenute all'interno di una parola, come le modifiche allo spelling, quelle che riguardano l'intera parola, e quelle che coinvolgono più parole.
Le operazioni strutturali che riguardano esclusivamente il testo sono le seguenti:

- TEXTINSERT: è l'inserimento di una stringa formata da più parole.

- TEXTDELETE: è il contrario dell'operazione precedente. Corrisponde ad una cancellazione di più parole.
- TEXTREPLACE: è la sostituzione di una stringa formata da più parole con un'altra.
- PUNCTUATION: è un sottotipo di TEXTREPLACE che riguarda sia le modifiche alla punteggiatura, sia quelle che avvengono come conseguenza di un cambiamento di punteggiatura. Quest'ultimo tipo di modifica si ha ad esempio quando la prima lettera di una parola diventa maiuscola in seguito all'aggiunta di un punto.
- WORDREPLACE: un sottotipo di TEXTREPLACE dove una parola viene sostituita con un'altra.
- WORDCHANGE: un sottotipo di WORDREPLACE in cui la parola non viene completamente sostituita, ma solo modificata.


```
{
  "id": "struct-00017",
  "op": "TEXTREPLACE",
  "by": "Fabio Vitali",
  "timestamp": "2019-03-10T07:26:44",
  "items": [{
    "id": "mech-00031",
    "op": "DEL",
    "pos": 94,
    "content": "Initial text."
  },{
    "id": "mech-00033",
    "op": "INS",
    "pos": 94,
    "content": "New words."
  }]
}
```


Fig. 20 - esempio di modifica strutturale, figura di [DSV19]

Le operazioni che riguardano il markup o un misto di markup e testo sono le seguenti:

- NOOP: una modifica che non ha nessun effetto sul documento, indipendentemente dal fatto che sia avvenuta dentro al markup o al testo. Un esempio è l'inversione dell'ordine, ma non del contenuto, degli attributi di un nodo.
- INSERT: è l'inserimento di uno o più nodi in un documento. Fanno parte della modifica anche eventuali aggiunte di testo prima o dopo ai nodi.
- DELETE: è la cancellazione di uno o più nodi e dell'eventuale testo che li segue o li precede.
- REPLACE: la sostituzione di un nodo con un altro, senza che venga modificato il contenuto.
- MOVE: è lo spostamento di uno o più nodi in una posizione diversa del documento. Le operazioni di cancellazione e inserimento che lo compongono possono non essere consecutive nel tempo, purché il loro contenuto sia identico e la posizione diversa.
- WRAP: l'inserimento di una porzione di documento all'interno di un nuovo nodo, che diventa il figlio di quello corrente. Le operazioni che lo compongono sono due inserimenti che riguardano esclusivamente il markup. Un esempio è l'aggiunta di uno stile, come il grassetto, ad una sezione di testo.
- UNWRAP: è l'operazione inversa della precedente. Consiste nella rimozione di un nodo, ed è formato da due cancellazioni di markup. Il contenuto del nodo eliminato rimane nella stessa posizione, ma ad un livello superiore.
- JOIN: è l'unione di due nodi fratelli dello stesso tipo. Il nodo risultante prende le caratteristiche del primo dei due uniti.

- SPLIT: è l'operazione contraria di quella precedente, ovvero la separazione di un nodo in due dello stesso tipo, ciascuno con le caratteristiche del nodo originario.

Infine, il terzo livello è quello semantico. Definisce operazioni che un utente può comprendere immediatamente in termini di significato e impatto sul documento,e che cercano di spiegare sia il tipo di azione avvenuta, sia il motivo per cui è stata effettuata. Identificare le operazioni che fanno parte di questo livello è un compito complesso che dipende molto dall'interpretazione personale, quindi la lista di modifiche è per sua stessa natura incompleta.
Le operazioni per ora definite sono le seguenti:

- MEANING: riguarda tutte le operazioni che cambiano il significato del testo. Viene inoltre assegnata a tutte le modifiche che non possono essere ricondotte a nessuna delle altre operazioni semantiche.
- FIX: la correzione di una parte di documento. Tutte le operazioni mirate a risolvere un errore fanno parte di questa categoria, indipendentemente dal fatto che riguardino il testo o il markup.
- STYLE: sono operazioni che modificano l'aspetto o il contenuto di un documento, senza cambiarne il significato. Un esempio può essere la sostituzione di una parola con un sinonimo.
- EDITWAR: è formata da una sequenza di operazioni strutturali che riguardano la stessa porzione di testo e riportano il contenuto del documento ad uno stato uguale o simile a quello precedente. Ciascuna delle modifiche che compongono una edit war annulla, del tutto o in gran parte, il contributo di quella precedente. Non è necessario che siano consecutive nel tempo, né che siano state effettuate dallo stesso autore.

- EDIT WAKE: è composta da una modifica principale e da una serie di modifiche di rilevanza minore che servono a risolvere eventuali errori grammaticali o di struttura causati dalla prima modifica.


```
{
  "id": "sem-00002",
  "op": "FIX",
  "old": "Some words is better than others",
  "new": "Some words are better than others",
  "items": {
    "id": "struct-00021",
    "op": "WORDREPLACE",
    "by": "Fabio Vitali",
    "timestamp": "2019-10-10T07:25:23",
    "items": [{
      "id" : "mech-00083",
      "op":"DEL",
      "pos": 215,
      "content": "is"
    },{
      "id" : "mech-00084",
      "op":"INS",
      "pos": 215,
      "content": "are"
    }]
  }
}
```


Fig. 21- esempio di modifica semantica, figura di [DSV19]

Questo modello offre una classificazione delle modifiche che ne descrive in maggior dettaglio il significato, ed esprime informazioni aggiuntive che verranno sfruttate per generare le varie visualizzazioni offerte dal progetto.

3.2 Lista delle modifiche

Come si è visto nel capitolo due, l'editor di testo SSE permette all'utente di congelare una versione del documento visualizzato. L'elenco delle versioni congelate è contenuta

all'interno del tab History, dove è possibile selezionarle per visualizzare il loro contenuto. Il congelamento non è un'operazione automatica, ma avviene in seguito ad una scelta dell'utente. Inoltre, viene preservato solo lo stato del testo, senza nessuna indicazione riguardante le operazioni specifiche che hanno portato alla creazione di quella versione.

La lista delle modifiche aggiunge al sistema di congelamento già presente la possibilità di visualizzare ogni modifica apportata al documento. Appare all'interno del tab History, in modo che la selezione di una versione già congelata permetta di visionare anche le modifiche che la caratterizzano. Queste appaiono in ordine cronologico, con quella più recente in cima.

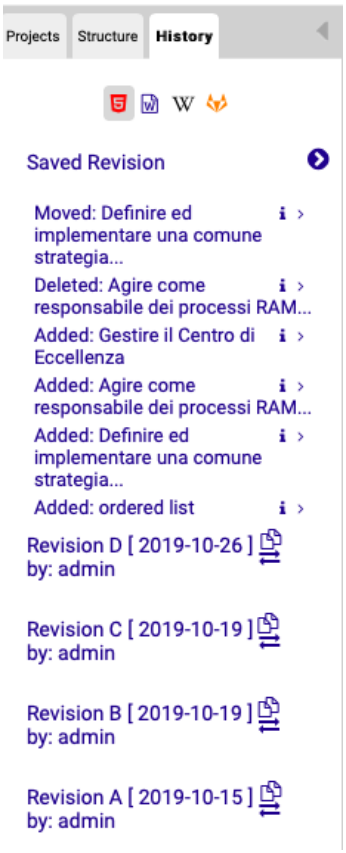


Fig. 22 - Lista delle modifiche visualizzata all'interno di SSE

Lo scopo della lista è sia quello di rappresentare i cambiamenti effettuati, sia di permettere all'utente di accedere ad informazioni aggiuntive su ciascuno, al livello di dettaglio interessato. Infatti, la visualizzazione di ciascuna modifica è composta da una struttura espandibile a tre livelli, ciascuno contenente informazioni diverse dai precedenti.
Il primo livello mostra il tipo generico di operazione effettuata. Per il testo normale, esistono quattro possibili categorie: cancellazione, inserimento, spostamento e sostituzione. Oltre all'operazione corrispondente, viene presentato anche il testo modificato. Nel caso sia troppo lungo è riportata solo la prima parte, mentre il contenuto integrale viene descritto dal terzo livello. Nel caso di operazioni che riguardano effetti applicati al testo o elementi particolari come tabelle o immagini, al posto del contenuto è riportato il tipo di effetto o struttura modificati. Questa visualizzazione permette all'utente di intuire la natura della modifica, senza appesantire la visualizzazione con ulteriori dettagli, che sono invece visibili nei livelli successivi.

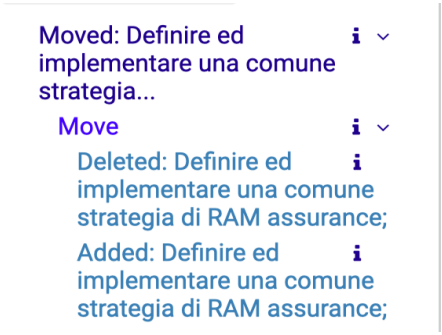


Fig.23 – Esempio di modifica completamente espansa

Il secondo livello esprime l'operazione specifica corrispondente alla modifica. Infine, il terzo presenta separatamente e in modo integrale il contenuto e l'operazione delle modifiche meccaniche che compongono il raggruppamento.
Accanto ad ogni livello è inoltre presente un'icona che permette di visualizzare attraverso un popup ulteriori informazioni riguardanti la modifica, quali ad esempio l'autore e la data in cui è stata effettuata.

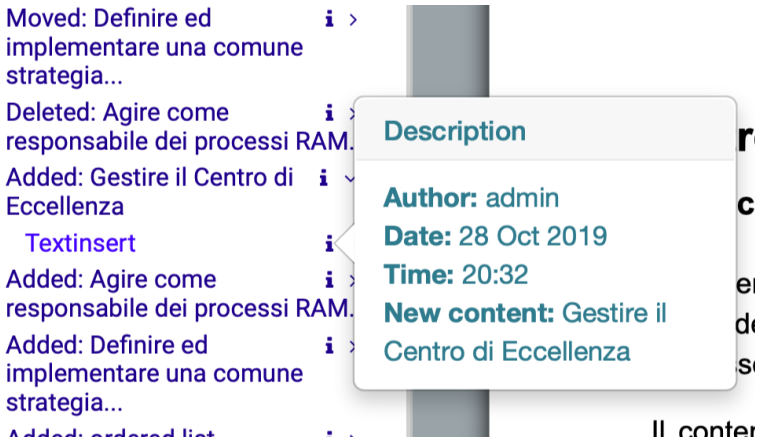


Fig.24 - Esempio di popup

Cliccando su una modifica, il documento scorre fino a rendere visibile l'area in cui è avvenuta. Questo permette all'utente di visualizzare il suo effetto sul testo, senza dover scorrere manualmente le parti non interessate.

3.3 Visualizzazioni

Come già visto nel capitolo tre, è possibile visualizzare le modifiche riguardanti la versione selezionata del documento attraverso tre stili diversi implementati dalla libreria Differ. Per accedere a ciascuno di essi viene fornito un widget nella parte alta dell'area dedicata alla lista delle modifiche, in aggiunta a un quarto che permette di tornare alla versione normale del testo, che è anche l'unica modificabile.

Fig.25 - Widget per attivare le visualizzazioni, con la versione normale attiva

Le visualizzazioni vengono caricate dentro l'area normalmente dedicata alla modifica del documento. Ho cercato di conservare il più possibile le caratteristiche originarie di

ciascuna, con qualche modifica. In particolare, il Wiki style e il Git style presentano l'intero paragrafo in cui si colloca il contenuto della modifica, che viene evidenziato all'interno di esso in rosso nel caso sia stato cancellato, in verde se rappresenta un'aggiunta.
Nel caso si selezioni una versione passata, viene conservata la modalità di visualizzazione attiva.

Capitolo 4

Architettura di EditTrails

In questo capitolo descriverò l'architettura del progetto. Nella prima sezione tratterò la parte client-side, soffermandomi sul procedimento usato per generare la lista delle modifiche, dal momento in cui vengono rilevati i cambiamenti all'interno dell'editor fino alla creazione della struttura a tre livelli che verrà visualizzata dall'utente. Descriverò anche come vengono applicate le visualizzazioni implementate da Dondi alle modifiche trovate.
Infine, nella seconda sezione descriverò i servizi server-side utilizzati per gestire il raggruppamento e il salvataggio delle modifiche.

4.1 Architettura client-side

La parte client-side dell'applicazione è gestita dal file tabhistory.js, che si occupa di generare le operazioni meccaniche, creare la lista delle modifiche e gestire le tre modalità di visualizzazione. Accennerò ai servizi server-side utilizzati, che saranno descritti meglio nella sezione dedicata.
La funzione generateEdits si occupa di individuare i cambiamenti a livello meccanico. Viene chiamata dall'event handler legato al cambiamento di contenuto dell'editor. Riceve in input l'oggetto relativo all'evento ed effettua una serie di controlli per verificare che contenga tutte le informazioni necessarie a generare le modifiche. Questo serve ad evitare

che vengano considerate come cambiamenti del testo anche operazioni che non lo sono, come il caricamento di un documento all'interno dell'editor o il salvataggio del suo contenuto.

In particolare, gli attributi dell'evento utilizzati per generare le modifiche sono i seguenti:

- originalEvent: è a sua volta un oggetto, e contiene informazioni utili ad identificare il tipo di operazione effettuata. In particolare, nel caso l'utente abbia attivato uno dei comandi dell'editor, ad esempio cliccando su uno degli elementi della barra degli strumenti, il suo attributo 'type' indica il tipo di azione effettuata.

- Level.bookmark e lastLevel.beforeBookmark: oggetti che contengono uno o più array numerici. Questi indicano il percorso da seguire all'interno della struttura del documento per trovare le posizioni di inizio e a volte di fine delle modifiche meccaniche effettuate. La presenza o meno di determinati array all'interno di questi oggetti permette di capire quali operazioni meccaniche compongano la modifica effettuata dall'utente. Ad esempio, selezionando una porzione di testo e cancellandola, si otterrebbe l'oggetto in figura 26, dove l'attributo start di lastLevel.beforeBookmark indica la posizione di inizio del contenuto cancellato e l'attributo end quella di fine. L'operazione meccanica risultante sarà appunto una cancellazione.

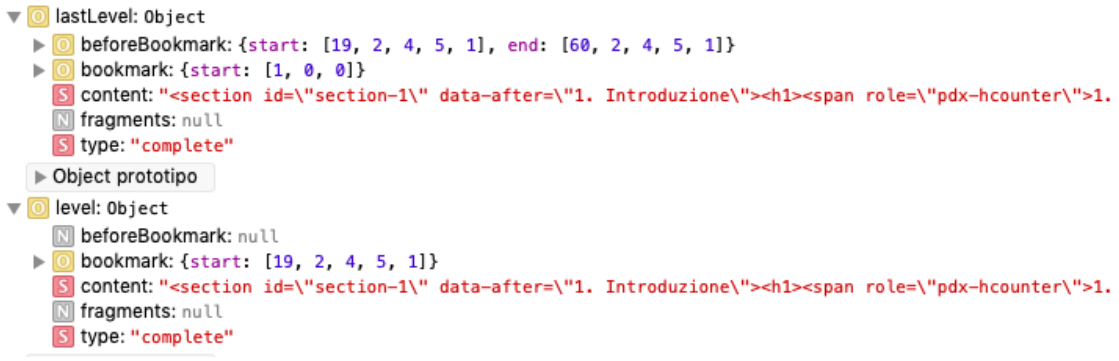


Fig.26 – struttura di level e lastLevel nel caso di una cancellazione con selezione

Nel caso l'oggetto relativo all'evento contenga tutte le informazioni necessarie, generateEdits analizza le caratteristiche degli attributi presenti all'interno di level e lastLevel, e in base ad esse chiama la funzione che gestisce la tipologia di modifica corrispondente. Di seguito descriverò brevemente queste funzioni e il tipo di operazioni di cui si occupano. Ciascuna di esse prende in input l'evento, il contenuto del documento precedente alla modifica e quello presente nell'editor, e restituisce un array di modifiche meccaniche.

- checkCommand: viene chiamata nel caso in cui l'attributo 'type' di originalEvent non sia vuoto. Gestisce il caso in cui sia stato incollato del testo e le operazioni di aggiunta e cancellazione di liste e cancellazione e modifica di tabelle. Si occupa anche degli stili applicati al testo.
- checkSections: viene chiamata in caso di aggiunta di una sezione o sottosezione.
- checkSelection: gestisce la cancellazione o sostituzione di contenuto che era stato precedentemente selezionato.
- checkDeletedImage: gestisce la cancellazione di un'immagine.
- checkAddedFigure: si occupa dell'aggiunta di un'immagine o una tabella.
- checkChanges: individua le operazioni di cancellazione, eliminazione o sostituzione di contenuto che non era stato selezionato.
- checkLabels: l'unica di questa lista a non venire chiamata da generateEdits, si attiva quando la didascalia di una tabella o immagine viene modificata.

Per ricavare la posizione corrispondente a questi array, generateEdits utilizza la funzione getPosition, che prende in input l'array da convertire e il testo a cui fa riferimento e restituisce un valore numerico. Questo corrisponde al numero di caratteri del documento

che precedono il punto in cui è stata effettuata la modifica. Il contenuto di ciascuna operazione meccanica viene ricavato a partire dalla posizione così ottenuta e da controlli specifici di ogni funzione, che dipendono dalla struttura dell'evento.
La lista di modifiche meccaniche generate in questo modo viene data in input alla funzione getEdits, che effettua una chiamata al servizio GetEdits del server per ottenere un raggruppamento generato da 3Diff. Le modifiche semantiche così ottenute vengono aggiunte all'array unsavedEdits, che contiene le modifiche non salvate.
Viene poi chiamata la funzione getEditsVisualization, che si occupa di generare la struttura HTML che sarà appesa all'interno del tab di SSE dedicato alla lista delle versioni del documento, permettendo così all'utente di visualizzare le modifiche.
I tre livelli in cui è divisa la visualizzazione di ciascuna modifica sono generati a partire dalle informazioni presenti nel raggruppamento.
Per le modifiche che riguardano il testo del documento, il primo livello visibile contiene il tipo di operazione effettuata e i primi quaranta caratteri del contenuto cambiato. Il tipo di operazione viene individuato a partire da quella strutturale, e può appartenere a quattro categorie:

- Aggiunta, indicata da “Added”: per operazioni strutturali relative a inserimento di contenuto, con una sola modifica meccanica di tipo INS.
- Cancellazione, indicata da “Deleted”: nel caso riguardi la rimozione di testo e abbia una sola modifica meccanica di tipo DEL.
- Sostituzione, indicata da “Changed”: se l'operazione esprime un cambiamento di contenuto e ha due modifiche meccaniche, una INS e una DEL, con la stessa posizione iniziale.
- Spostamento, indicato da “Moved”: se l'operazione strutturale è di tipo MOVE.

Nel caso in cui il contenuto comprenda anche elementi di markup, questi vengono esaminati per individuare l'aggiunta, cancellazione o modifica di tabelle, immagini, liste e stili applicati al testo. Tutte queste modifiche vengono segnalate in modo specifico per il tipo di elemento a cui fanno riferimento. Ad esempio, l'aggiunta di una tabella sarà descritta dalla dicitura: “Added: table”.

L'operazione strutturale specifica viene visualizzata nel secondo livello. Infine, il terzo presenta il contenuto e l'operazione delle modifiche meccaniche. A differenza del primo livello, dove i contenuti troppo lunghi vengono troncati per aiutare la leggibilità della lista, il terzo li riporta in modo integrale.

La gestione delle informazioni rimanenti relative a ciascun elemento della lista è affidata alla funzione findInfo, che prende in input una modifica semantica e restituisce la struttura HTML del popup che la descrive.

Nel markup relativo ad ogni livello è contenuto anche l'identificativo della modifica meccanica, strutturale o semantica corrispondente. Quando l'utente clicca su una delle descrizioni dei livelli, l'handler dell'evento correlato recupera l'identificativo della modifica corrispondente e controlla se appartenga ad un'operazione salvata o non salvata. Nel primo caso, effettua una chiamata al servizio GetEditsList del server per ottenere un array contenente la lista di modifiche della versione visualizzata. Altrimenti, recupera il contenuto dell'array delle modifiche non salvate. In entrambi i casi, la lista così ottenuta viene tagliata per ottenere solo le modifiche avvenute dopo quella selezionata.

Viene poi chiamata la funzione getEditElement, che prende in input questo array, l'identificativo della modifica cliccata e la sua posizione. Prima di effettuare la ricerca dell'elemento corrispondente, estrae le modifiche meccaniche dall'array ricevuto in input, grazie alla funzione extractMechanicalEdits. Confronta poi la posizione di ciascuno degli elementi del risultato con quella della modifica cliccata. Nel caso la posizione della modifica meccanica preceda quella della modifica selezionata, la posizione di quest'ultima viene decrementata o incrementata di una quantità pari alla lunghezza del contenuto della

modifica meccanica. In particolare, viene sottratta se l'operazione meccanica esaminata è di tipo INS, aggiunta se di tipo DEL. In questo modo, la posizione risultante terrà conto dei cambiamenti del testo avvenuti dopo la modifica selezionata.

Infine, utilizzando la posizione appena calcolata e il contenuto dell'editor, la funzione scrollToEdit individua l'elemento cercato, fa scorrere la pagina fino a renderlo visibile e lo evidenzia brevemente di grigio.

Quando si accede ad un documento, la visualizzazione attiva è quella normale, che mostra solamente il suo contenuto. È possibile cliccare sui widget delle visualizzazioni presentati nel capitolo tre per cambiare modalità.

In questo caso, si effettua prima una chiamata al servizio GetVisualizationData del server per recuperare la lista di modifiche corrispondente alla versione del documento visualizzata. Poi, per elaborare la visualizzazione richiesta, viene chiamata la funzione corrispondente della libreria Differ. Quelle disponibili sono loadWordStyle, loadGitStyle e loadWikiStyle, che generano rispettivamente le visualizzazioni Word style, Git style e Wiki style. La prima richiede in input sia le modifiche che il contenuto del documento, mentre le altre due solo le modifiche.

La visualizzazione viene caricata all'interno dell'editor. Nel caso l'utente selezioni una versione diversa del documento, la modalità di visualizzazione rimane quella attiva, e il suo contenuto viene aggiornato con le informazioni relative alla versione scelta. Per tornare alla visualizzazione normale, è sufficiente cliccare sulla prima icona. In questo caso, verrà caricato il contenuto del documento all'interno dell'editor, senza nessun tipo di elaborazione ulteriore.

4.2 Architettura server-side

La parte server-side è gestita da due file: history.js, che si occupa delle richieste, e il modulo historymodule.js, che implementa le funzioni utilizzate da history.js per modificare il contenuto dei file relativi alle modifiche.
I servizi di history.js sono:

- GetEdits: riceve una lista di modifiche meccaniche e le raggruppa grazie alla funzione layering del modulo.
- GetEditsList: riceve il path che identifica il documento attivo e restituisce la lista completa delle modifiche effettuate sul documento attraverso tutte le sue versioni. Questa lista è salvata nel file history.json specifico di ogni documento, e viene aggiornata dopo ogni congelamento con le modifiche della versione appena congelata.
- GetVisualizationData: restituisce la lista di modifiche relative unicamente alla versione del file indicata.
- SaveEdits: riceve le modifiche non salvate e le appende al file edits.json del documento e della versione specificata tramite la funzione append del modulo.
- SaveHistory: salva la lista di modifiche e l'identificativo dell'ultima versione creata nel file history.json.

Il file historymodule.js, invece, contiene le seguente funzioni:

- layering: riceve in input una lista di modifiche meccaniche, la versione del documento a cui sono applicate e quella successiva, e applica ad esse la ricombinazione implementata da 3Diff.

- save: prende in input una lista di modifiche e l'identificativo del documento in cui deve essere salvata. Nel caso in cui questo esista già, il suo contenuto viene sovrascritto, altrimenti viene creato da zero.
- show: prende in input una lista di modifiche e due indici, e restituisce la porzione di lista compresa tra di essi, dove possibile. Altrimenti, restituisce tutta la lista.
- append: riceve una lista di modifiche e l'identificativo di un file. Effettua una ricombinazione tra le modifiche già presenti nel file e quelle fornite, poi appende queste ultime al file indicato. Gli indici delle modifiche aggiunte vengono modificate grazie alla funzione changeIds.

Capitolo 5

Valutazione

Ho implementato EditTrails allo scopo di ottenere un sistema che permettesse di rilevare le modifiche apportate ad un documento e accedere alle informazioni che le riguardano tramite opportune visualizzazioni. Come visto nel capitolo tre, queste visualizzazioni si compongono di una lista delle modifiche ricavata grazie al raggruppamento nel modello a tre livelli, e dalle visualizzazioni ottenute tramite la libreria Differ.

Per mostrare l'efficacia di queste rappresentazioni, ho scelto di confrontarle con quelle offerte da uno degli strumenti analizzati nel secondo capitolo, in particolare un editor di testo. Infatti, anche le visualizzazioni di EditTrails sono generate nel contesto di un editor, ovvero SSE, e condivide quindi alcune caratteristiche con questo tipo di strumenti. In particolare, mostra le modifiche dopo che queste vengono applicate al documento, rilevandole in modo automatico, e offre almeno due modalità diverse di visualizzazione delle modifiche, di cui una riassuntiva sotto forma di lista.

Tra gli editor analizzati ho scelto di usare Apple Pages, ma avrei potuto ottenere risultati simili anche utilizzandone uno diverso. Infatti, le visualizzazioni che offrono sono abbastanza simili fra loro e presentano gli stessi difetti, ovvero la mancanza di un'analisi più approfondita delle modifiche e di informazioni dettagliate su di esse.

Ho effettuato il confronto tra le visualizzazioni di Apple Pages e quelle di EditTrails utilizzando uno dei documenti tecnici forniti da Alstom. Dopo aver caricato il documento all'interno di entrambi gli editor, ho applicato manualmente la stessa sequenza di modifiche

su ciascuna delle due copie. L'analisi che segue è incentrata su un sottoinsieme di queste modifiche formato da una cancellazione di testo, un inserimento, uno spostamento, una sostituzione, e l'aggiunta dell'effetto grassetto ad una parola. Ho scelto queste modifiche perché sono di complessità diversa e riguardano una o più operazioni a basso livello. Inoltre, il cambiamento di stile permette di valutare come vengono gestite le modifiche che coinvolgono esclusivamente il markup.

In Pages, la visualizzazione delle modifiche è divisa in due parti: una colonna riassuntiva che espone una panoramica delle modifiche avvenute, e una rappresentazione sul testo, dove i caratteri modificati sono evidenziati in giallo.

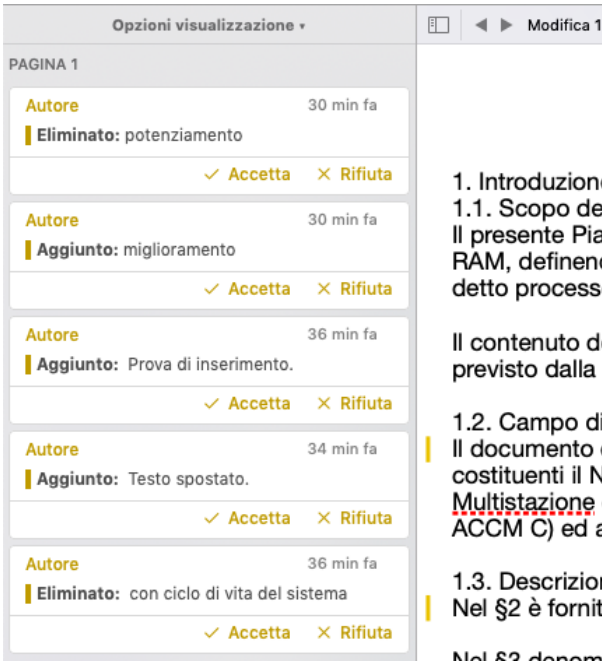


Fig.27 – Lista di modifiche visualizzata in Apple Pages

Nella figura 27 viene rappresentata la lista di modifiche di Pages, mentre nella figura 28 quella di EditTrails, con i livelli completamente espansi.

La visualizzazione proposta da Pages segue esclusivamente l'ordine delle modifiche all'interno del testo, nonostante l'orario relativo a ciascuna di esse venga comunque

segnalato. Inoltre, rileva le modifiche a livello meccanico, ma non esprime le relazioni tra di esse, né le analizza per ricavare informazioni ad alto livello, a differenza di quanto avviene in EditTrails. In particolare, nella lista esaminata la parola “potenziamento” viene cambiata in “miglioramento”. Pages presenta queste due modifiche come distinte fra loro, mentre EditTrails riesce a individuare la sostituzione di testo di cui fanno parte.

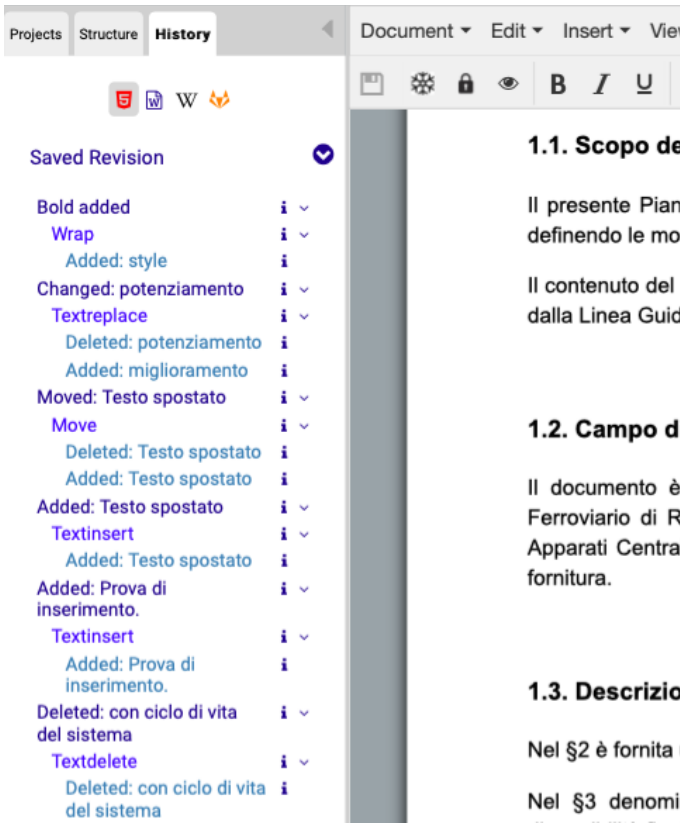


Fig.28 – Lista di modifiche visualizzata nel tab History

Un'altra differenza sostanziale risiede in come vengono ricombinate le modifiche. Nell'esempio, è stata aggiunta la stringa “Testo spostato”, che poi in seguito è stata effettivamente rimossa dalla posizione in cui si trovava e incollata in un punto differente del testo. Mentre EditTrails ha unito le due modifiche relative allo spostamento, Pages ha

sostituito il primo inserimento con il secondo, senza segnalare in alcun modo la cancellazione di testo.

Le modifiche che riguardano solo il markup, come appunto l'aggiunta di un effetto, non vengono evidenziate in alcun modo da Pages, mentre in EditTrails sono opportunamente segnalate.

Infine, Pages non registra tutte le modifiche applicate ad un documento, ma solo quelle che l'utente deve ancora esaminare, e non le conserva dopo che sono state approvate o rifiutate, al contrario di EditTrails che registra la storia completa del documento.

In aggiunta alla lista, Pages offre solo una modalità di visualizzazione delle modifiche, mentre EditTrails permette di selezionarne tre. Tra di esse, Word style è quella che si avvicina di più alla visualizzazione di Pages, aggiungendo una distinzione più chiara grazie all'utilizzo di diversi colori. Nonostante questo, nessuna delle visualizzazioni di EditTrails permette di rappresentare le modifiche sul testo mentre questo viene modificato, caratteristica invece presente in Pages.

In generale, le modalità di visualizzazione offerte da EditTrails appaiono più dettagliate rispetto a quelle trovate negli strumenti esaminati fino a questo momento. Condivide però con essi il difetto di non riuscire ad offrire una soluzione definitiva al problema del sovrappopolamento della lista. Infatti, nel caso di documenti lunghi a cui è stata applicata una grande quantità di modifiche, le dimensioni della lista potrebbero aumentare in modo tale da rendere la visualizzazione più confusa e le informazioni più difficili da accedere per gli utenti. Nel caso di EditTrails, questo effetto potrebbe essere arginato assicurandosi che le versioni dei documenti vengano congelate prima che la lista di modifiche aumenti eccessivamente di dimensioni. Purtroppo, questa responsabilità cadrebbe sull'utente, dal momento che il congelamento non è un'operazione automatica. Occorrerebbe quindi trovare un altro tipo di soluzione.

Conclusioni

Nel primo capitolo ho decritto lo scopo del progetto e gli aspetti del problema che si pone di risolvere, ovvero quello di rilevare e visualizzare le modifiche apportate ad un documento in modo da rendere le informazioni che le riguardano facilmente accessibili all'utente.
Nel secondo, invece, ho esaminato una serie di strumenti che utilizzavano vari metodi di rilevazione e visualizzazione delle modifiche, tra cui editor di testo e sistemi di scrittura collaborativa di documenti, evidenziando come non forniscano rappresentazioni esaustive dal punto di vista delle informazioni visualizzate. Ho inoltre trattato le tecnologie utilizzate per implementare il progetto, del quale costituiscono una parte fondamentale.
Nel terzo capitolo ho descritto nel dettaglio il modello a tre livelli ed esposto le funzionalità di EditTrails, focalizzandomi sulle informazioni visualizzate nella lista delle modifiche e sulle varie modalità di visualizzazione proposte.
La loro implementazione è stata descritta nel quarto capitolo, che tratta dell'architettura del progetto, entrando nel dettaglio su come vengono generate ed elaborate le modifiche.
Infine, nel quinto capitolo ho esaminato i pregi e i difetti del progetto rispetto agli strumenti già analizzati, prendendo come esempio l'editor di testo Apple Pages.
Il risultato di questo percorso è un sistema che riesce a gestire sia la generazione delle modifiche, sia l'elaborazione delle informazioni relative ad esse e la loro visualizzazione. A differenza degli altri strumenti analizzati, recupera le operazioni meccaniche mentre queste vengono applicate al documento, evitando le imprecisioni che invece potrebbero risultare dall'utilizzo di un algoritmo di diffing. Inoltre, usando l'algoritmo 3Diff effettua un raggruppamento delle operazioni meccaniche che permette di avere una descrizione

completa di ogni singola modifica. L'utente può scegliere quali informazioni visualizzare grazie alla divisione in livelli della lista e ai popup, che presentano dettagli aggiuntivi. Infine, offre la possibilità di generare le tre visualizzazioni implementate dalla libreria Differ.

Come evidenziato nel quarto capitolo, in caso di grandi quantità di modifiche la visualizzazione in forma di lista perde in parte la sua efficacia.

Per risolvere questo problema, potrebbe essere utile implementare un sistema di filtraggio delle modifiche simile a quello trovato in alcuni degli strumenti esaminati nel capitolo uno, che permetterebbe di visualizzare solo cambiamenti che hanno le caratteristiche cercate, o raggruppati secondo determinati criteri. Ad esempio, l'utente potrebbe scegliere di visualizzare solo le operazioni avvenute in un determinato periodo di tempo, o quelle effettuate da uno specifico autore. Un'altra possibilità sarebbe quella di poterle ordinare in base alla loro posizione nel testo, come già avviene nella visualizzazione Word style.

Si potrebbe inoltre elaborare ulteriormente la descrizione del primo livello visibile delle modifiche, in modo che esprima in modo più specifico l'operazione corrispondente.

Bibliografia

[AM16] Alsubhi A., Munson E., *Design and Usability Testing of a User Interface for Three-Way Document Merging*, Dchanges'16, Settembre 13 2016, Vienna, Austria, http://dx.doi.org/10.1145/2993585.2993590

[CDV18] Caponi A., Di Iorio A., Vitali F., Alberti P., Scatà M., *Exploiting patterns and templates for technical documentation,* DocEng ’18, Agosto 28–31, 2018, Halifax, Canada, https://doi.org/10.1145/3209280.3209537

[CHA06] Chawathe S.S., *Tracking Changes in Healthcare Documents*, CBMS'06, 22-23 Giugno 2006, Salt Lake City, UT, USA, https://doi.org/10.1109/CBMS.2006.161

[DIF19], Sourcegear Software Services, *Diffmerge*, ultima visita il: 17/11/19, https://sourcegear.com/diffmerge/

[DON19], Dondi S., *Differ – Elaborazione e rappresentazione di modifiche a documenti strutturati*, 12 Marzo 2019, https://amslaurea.unibo.it/id/eprint/17913

[DSV19] Di Iorio A., Spinaci G., Vitali F., *Multi-layered edits for meaningful interpretation of textual differences*, Proceedings of the ACM Symposium on Document Engineering, 23-26 Settembre 2019, Berlino, Germania https://doi.org/10.1145/3342558.3345406

[FB10] Fong P.K., Biuk-Aghai R.P., *What Did They Do? Deriving High-Level Edit Histories in Wikis* ,WikiSym '10, July 7-9, 2010, Gdańsk, Polonia, https://doi.org/10.1145/1832772.1832775

[KKT11], Kehrer T., Kelter U., Taentzer G., *A Rule-Based Approach to the Semantic Lifting of Model Differences in the Context of Model Versioning*, ASE 2011, Lawrence, KS, USA, https://doi.org/10.1109/ASE.2011.6100050

[MR15] Maier S., Rönnau S. *A REST-based Document Model For Collaborative Editing of Documents*, DChanges 2015, 8/09/2015, Lausanne, Svizzera, http://dx.doi.org/10.1145/2881631.2881636

[PIO06] Papadopoulou S., Ignat CL., Oster G., Norrie M.C., *Increasing Awareness in Collaborative Authoring through Edit profiling*, IEEE, 17-20/11/2005 http://dx.doi.org/10.1109/COLCOM.2006.361864

[PIO08] Papadopoulou S., Ignat CL., Oster G., Norrie M.C., *Intra/Inter-document Change Awareness for Co-authoring of Web Sites*, WISE 2008 pp. 90-115, https://doi.org/10.1007/978-3-540-85481-4_9

[SPI19], Spinaci G., Documentazione di 3Diff, Github, 2019, https://github.com/three-level-model-diff/3Diff

[WIK19] Wikimedia Foundation, *Aiuto:Cronologia*, Wikipedia, ultima modifica il: 27/10/2019, https://it.wikipedia.org/wiki/Aiuto:Cronologia

[WIN19], Winmerge, ultima visita il: 17/11/19, https://winmerge.org