Alma Mater Studiorum Università di Bologna

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di laurea in Informatica

TEST SISTEMATICI PER TRADUTTORI DI FORMATO

Tesi di laurea in Tecnologie Web

RELATORE

Chiar.mo Prof.

Fabio VITALI

CORRELATORE

Dott. Angelo DI IORIO

Presentata da:

Michele COLASANTE

I Sessione

ANNO ACCADEMICO 2009 / 10

INDICE

CAPITOLO 1

INTRODUZIONE

Sostenere il processo di verifica nello sviluppo di un software è una fase molto importante: viene realizzata per descrivere in dettaglio le funzionalità previste dai requisiti di un applicativo.
Diverse discipline studiano processi e metodologie finalizzate alla corretta progettazione di un sistema software. In molti casi però, la correttezza delle soluzioni realizzate viene determinata dopo un lungo processo di verifica, nel quale si propongono scenari di testing, caratterizzati da finalità diverse,volte a garantire l'affidabilità del software.

Lo scopo di questa tesi, dunque, è realizzare un ambiente di testing per la verifica sistematica di traduttori per formati documentali. Infatti, data la vasta gamma di applicativi e di formati di cui disponiamo oggi, siamo sempre in cerca di soluzioni per utilizzare lo stesso documento sui più lettori digitali.
Il formato di un documento è legato al software che lo produce. Questo legame si traduce in un'incompatibilità, quando un utente vuole elaborare un documento mediante uno strumento che non supporta tale formato.
Quando impossibilitati, adottiamo delle soluzioni di traduzione per adattare il formato specifico all'applicazione che intendiamo utilizzare. Queste soluzioni, nella maggioranza dei casi, sono già presenti nei software di comune utilizzo (Office, Open Office): ci permettono di salvare un file con un estensione diversa, oppure esportare il documento in un formato diverso. Il sostegno per la compatibilità non è sempre garantito, ovvero, gli strumenti che utilizziamo non prevedono tutti i formati possibili che vorremmo utilizzare. In questi casi, tali soluzioni di conversione vengono sviluppate come software indipendenti e non integrati in uno strumento di lavoro già esistente.
I team di sviluppo, che progettano queste idee, cercano di sostenere gli utenti nella loro gestione dei formati, con l'obiettivo di migliorare e facilitare l'interazione fra sistemi differenti nonché lo scambio e il riutilizzo delle informazioni, anche in sistemi non omogenei.

Gli scenari proposti in questa tesi studiano problematiche relative alla fase di testing per traduttori di formato, con l'obiettivo di realizzare un ambiente di verifica. OpenXMLODFConverter è un progetto interno della facoltà di Bologna: scritto in Java, allo stato attuale supporta un solo tipo di

conversione, ovvero dal formato word, basato su OpenXML di Office, al formato di testo di OpenOffice, basato su OpenDocument.
I formati descritti per il convertitore, anche essendo standard, introducono molte differenze; alcune di queste svantaggiano gli utenti che li utilizzano, altre si propagano agli sviluppatori che realizzano sistemi di conversione.
Le differenze introdotte dai formati limitano le soluzioni adottate dai programmatori per la traduzione delle funzionalità, in quanto sono sempre in cerca di tecniche per massimizzare la qualità del risultato, avendo sempre più l'esigenza di garantire determinate prestazioni. Il supporto di adeguati ambienti di testing avvantaggia la ricerca qualitativa.

Per analizzare le funzionalità offerte dal convertitore, ho realizzato un insieme di test eseguiti da un applicativo incaricato di eseguire i test.
La fase di progettazione è stata caratterizzata da un attento studio sulle tecniche e metodologie, proposte dall'ingegneria del software, per la realizzazione di test di verifica. Infatti, sono stati riproposti diversi test, tra i quali: test funzionali caratterizzati dall'analisi delle funzionalità, test di performance che descrivono le prestazioni in termini di spazio e utilizzo di risorse e test di errore, atti alla verifica delle anomalie, che si ripercuotono sulla generazione dei risultati.
La fase descritta più dettagliatamente è stata quella della verifica funzionale, infatti, la test suite è stato il punto cruciale della ricerca, realizzata grazie allo studio effettuato sugli scenari già esistenti che svolgono lo stesso ruolo.

Successivamente, lo studio di ricerca ha portato a realizzare un meccanismo che automatizzasse la fase di verifica. Così, abbiamo realizzato un tester con lo scopo di velocizzare il processo di testing, supportando la supervisione dei risultati e delle statistiche attraverso un sistema di documentazione.
Il tester offre due modalità di utilizzo. La prima, attraverso un'interfaccia grafica che semplifica il passaggio di parametri e la creazione della documentazione. La seconda, in modalità da terminale, ripropone le stesse funzionalità della precedente, offrendo il vantaggio di poter essere eseguita attraverso script o interpreti batch.
Il tester sfrutta un plugin di Office che si propone come metro di giudizio, in quanto svolge lo stesso ruolo di OpenXMLODFConverter.
Non avendo test precedenti sul convertitore, si è ritenuto opportuno utilizzare software che presentassero funzionalità simili, per apportare paragoni sulla valutazione dei risultati. Grazie alla modalità batch, offerta dall'add-ins di Office, il tester riesce ad eseguire entrambi gli applicativi, con lo scopo di rilasciare, al programmatore che effettua i test, una base di partenza per la valutazione dei risultati attesi. Lo svantaggio introdotto dal plugin di Office è stato l'utilizzo di moduli "dll", che non hanno permesso di adattare il tester in un ambiente operativo diverso da quello di Windows.

Il tester utilizza un sistema di report per annotare le valutazioni riscontrate durante la verifica contribuendo, così, alla documentazione finale.
La documentazione generata dal tester offre un valido strumento di confronto nel tempo. Infatti, grazie ai risultati prodotti, lo sviluppatore che attua le verifiche potrà averne la visione complessiva, anche dopo modifiche apportate alla struttura del software. Così facendo, potrà attuare verifiche in funzione delle versioni della release, focalizzando velocemente i risultati passati con quelli attuali.
La documentazione descrive dettagliatamente le caratteristiche dei dati ottenuti nel processo di conversione, riportando statistiche di successo e di insuccesso. Anche la fase di sviluppo del tester è stata influenzata dallo studio sulle metodologie proposte dall' ingegneria del software per ambienti di testing.
Nel progetto di tesi verranno descritte tali tecniche, in riferimento alle soluzioni adottate nell'implementazione del tester.

CAPITOLO 2

FORMATI DOCUMENTALI

Oggi i formati digitali offrono un valido strumento di supporto per lo scambio d'informazioni.
I benefici derivanti dall'utilizzo di questi strumenti sono la diminuzione dei costi e dei tempi per la creazione e la gestione delle idee. Nonostante ciò, il passaggio dalla gestione cartacea a quella digitale ha introdotto diverse difficoltà. Non potendo i supporti digitali comprendere tutti i formati documentali, siamo costretti a dover cercare delle soluzioni per potere utilizzare quest'ultimo su più applicazioni.
Gli aspetti tecnici su cui ci soffermeremo in questo capitolo sono legati agli standard per documenti di ufficio e alle soluzioni di traduzione e conversione (utilizzati per la giusta formattazione del documento da parte dell'applicativo che li interpreta). Verranno analizzati i modelli dei fogli di testo di Office e OpenOffice, in riferimento alle relazione e alle diversità che introducono, con lo scopo di proporre uno scenario di testing per "OpenXMLODFConverter" (progetto interno alla facoltà di Bologna), paragonandolo poi con applicativi già esistenti che svolgono lo stesso ruolo.

2.1. OpenXML e OpenDocument

I formati documentali analizzati durante il processo di testing sono stati "docx" della suite Office e "odt" di Open Office. L'estensione "docx" appartiene ai documenti di testo della suite Office basati sul formato OpenXML. Sfruttando tecnologia ZIP per la compressione dei dati e XML per la rappresentazione interna, OpenXML, sviluppato da Microsoft, viene considerato un formato aperto in quanto standardizzato nel 15 agosto del 2008 dall' ISO con la sigla ISO/IEC DIS 29500. Lo standard ha definito due differenti set di specifiche: Transitional ovvero il set contenente funzionalità definite "deprecate" ed incluse per retrocompatibilità. Al contrario Strict è il set contenente le nuove funzionalità, approvato allo scopo di diventare lo standard effettivo **[IS10]**. In Open Office i documenti di testo hanno l'estensione "odt" e sono basati sul formato OpenDocument (anch'esso adotta ZIP per la compressione dei dati e XML per la rappresentazione interna). OpenDocument, sviluppato dal consorzio industriale OASIS, è un formato aperto standardizzato dall'ISO nel 1 maggio 2006 rilasciato da con la sigla UNI CEI ISO/IEC **[IS06]**. Seppur i due documenti vengano utilizzati

equivalentemente, gli standard su cui sono basati presentano una differente struttura.

2.2. Struttura Package WordProcessingML

Il WordprocessingML package, del formato word basato su specifiche Office OpenXML, è composto da un'insieme di parti (alcune richieste obbligatoriamente altre facoltative) **[MV07a]**.
Il package WordProcessingML utilizza due tipi di tecnologie: ZIP e XML.
Gli elementi che compongono il pacchetto ZIP sono:

- ***Core Properties***: Contiene i metadati del documento.
- ***Main Document***: Rappresenta il body del documento.
- ***Glossary Document***: Include dei frammenti del documento (che non sono visibili all'utente ma che lo potrebbero esserlo per il Main Document).
- ***Bibliography***: Contiene i dati bibliografici.
- ***Comments***: Specifica le informazioni di tutti i commenti presenti nel documento.
- ***EndNotes***: Mantiene tutte le note di chiusura del documento.
- ***FootNotes***: Include tutte le note al piè della pagina.
- ***Header***: Rappresenta l'intestazione (mostrata all'inizio di una o più pagine).
- ***Footer***: Contiene la rappresentazione del piè di pagina (mostrato alla fine di una o più pagine).
- ***Numbering Definitions***: Include le proprietà per ogni lista numerata usata nel documento.
- ***Style Definitions***: Contiene le definizioni di un set di stili usati negli elementi del documento.
- ***Theme***: Definisce le proprietà di formattazione e di effetti visivi degli elementi quali Headings, table ecc.
- ***Fonts table and fonts***: Contiene le informazioni dei font usati nel documento.
- ***Document settings***: Definisce le proprietà del documento.
- ***Web Settings***: Definisce le proprietà specifiche per il web nel documento.
- ***Printer Setting***: Contiene le informazioni per l'inizializzazione e la creazione di un "ambiente" per una stampante o per un dispositivo video.

2.2.1. Struttura documento WordProcessingML

Un documento WordProcessingML è composto da una raccolta di "racconti" **[N06]**, ognuno dei quali è un elemento che compone il package precedentemente descritto. L'unico vincolo imposto per la creazione del package è il documento principale.

Essendo la struttura del racconto ad albero, un tipico percorso dalla radice alla foglia dovrebbe comprendere questi elementi XML **[N06]** :

<w:document> è il nodo radice del documento.

<w:body> definisce il corpo e può contenere al suo interno uno o più paragrafi.

<w:p> definisce l'elemento paragrafo, può contenere al suo interno una o più corse; può anche includere le proprietà paragrafo specificate dall'elemento <w:pPr>, che a sua volta può contenere proprietà di default (come, ad esempio, l'elemento <w:rPr> che definisce proprietà di formattazione relative agli oggetti <w:r>).

<w:r> Questo tipo di elemento è un concetto fondamentale per il formato OpenXML. Può includere diversi tipi di contenuti come il dominio del testo ma anche le sue proprietà. Realizza un porzione di testo contiguo con caratteristiche condivise. Non contenendo markup aggiuntivo di testo, differisce in modo significativo da altri formati che permettono, ad esempio, la nidificazione arbitraria di proprietà (come HTML) **[R06]**.

<w:t> è la sezione text che racchiude una quantità arbitraria di testo senza formattazione, interruzioni di linea, tabelle, grafici o altro materiale non testuale. La formattazione del testo è ereditata dalle proprietà d' esecuzione e di paragrafo.

<w:tbl> sintatticamente rappresenta la tabella. Può contenere al suo interno gli elementi che definiscono le righe <w:tr> e/o gli elementi che esplicitano le celle <w:tc>

Diversi linguaggi di markup sono supportati da OpenXML per la descrizione di forme, operazioni matematiche e grafica vettoriale. Tra questi linguaggi troviamo:

- ***DrawingML***: usato per rappresentare le forme e altri oggetti grafici all'interno del documento.
- ***VML***: un formato per la grafica vettoriale che è incluso per compatibilità e sarà sostituito con DrawingML.
- Altri linguaggi di markup condivisi dai formati della suite di Office sono ***Math*** per contesti matematici.

2.2.2. Paragrafi

Sintatticamente, un paragrafo è rappresentato attraverso un elemento <w:p>. Le specifiche Office OpenXML **[MV07a]** permettono di impostare diverse proprietà a questo livello, riassunte nelle seguenti sottosezioni:

- ***Correzione automatica rientro destro griglia del documento***: questa proprietà specifica se il rientro destro di un paragrafo deve essere regolato automaticamente quando la griglia del documento è stata definita per la sezione del paragrafo.
- ***Correzione automatica della spaziatura testo latino e Oriente-Asia***: specifica se il carattere di inter- spazio deve essere regolato automaticamente tra le regioni del testo latino e le regioni di testo Est-Asiatico.
- ***Correzione Automatica della spaziatura testo est-asiatico e numerico***: specifica se tra i suoi caratteri di spazio deve essere regolato automaticamente la porzione di spazio compresa tra le regioni dei numeri e quelle di testo Est-Asiatico.
- ***Layout verso sinistro del paragrafo***: specifica che un paragrafo è presentato utilizzando la direzione da destra verso sinistra. E' importante per le lingue non occidentali.
- ***Ignora spaziatura sopra e sotto con stili "indent"***: specifica che l'inter-paragrafo spaziatura dovrebbe essere ignorato per i paragrafi contigui con lo stesso stile. Utile quando vengono visualizzate le liste.
- ***Punti di tabulazione personalizzati***: personalizza i punti di tabulazione di un paragrafo.
- ***Rientro paragrafo***: definisce le proprietà di rientro del paragrafo.
- ***Allineamento del paragrafo***.
- ***Allineamento verticale linea***: specifica l'allineamento verticale del testo nel paragrafo.

2.2.3. Runs

La corsa è sintatticamente rappresentata attraverso l'elemento <w:r>. A questo livello Office OpenXML consente di impostare le seguenti proprietà **[MV07a]**:

- ***Grassetto***: contrassegna una corsa in grassetto.
- ***Italico***: codifica una esecuzione come corsivo.
- ***Testo di confine***: specifica un bordo che delimiti la corsa.
- ***Caps***: forza l'applicazione per il rendering del contenuto eseguito in lettere maiuscole.
- ***Colori***: specifica il colore con cui dovrebbe essere riprodotta la corsa.
- ***Strike***: specifica che il contenuto di corsa deve essere visualizzato con una singola linea orizzontale.
- ***Effetto testo animato***: specifica effetti di animazione per il testo.
- ***Accento***: marca in rilievo un percorso della corsa.
- ***Larghezza corsa***: specifica una larghezza fissa di una corsa di contenuti.
- ***Evidenziazione***: evidenzia il contenuto della corsa con un colore a scelta.
- ***Lingua***: specifica la lingua con cui il contenuto della corsa deve essere interpretato.
- ***Controllo Ortografico***: disabilita il controllo ortografico per i livelli delle corse.
- ***Contorno***: codifica una corsa in modo che venga visualizzata come se avesse un contorno.
- ***Tipi di caratteri***: specifica il font da usare per visualizzare il contenuto della corsa.
- ***Spaziatura***: specifica la spaziatura tra i caratteri nel contenuto della corsa.
- ***Dimensione carattere***: specifica l'estensione del font per visualizzare un contenuto di esecuzione.
- ***Sottolineato***: specifica che il contenuto della corsa debba essere marcato.
- ***Testo nascosto***: è possibile specificare che un percorso di una corsa non venga visualizzato.

2.2.4. Tabelle

E' possibile specificare proprietà a livello di tabella, riga e cella **[MV07a]**.

A livello di tabella avremo:

- ***Tabella destra verso sinistra***: specifica che la visualizzazione delle celle che compongono la tabella avvenga da destra verso sinistra, invece che da sinistra verso destra (default).

- ***Allineamento della tabella***: viene utilizzato per determinare come la griglia dovrebbe essere posizionata rispetto ai margini della pagina.
- ***Rientro tabella***: specifica lo spazio tra il bordo della tabella e il margine della pagina.
- ***Sovrapposizione tabelle***: specifica che le altre tabelle possono sovrapporsi durante la visualizzazione di una di queste.
- ***Posizionamento***: specifica che la tabella non fa parte del flusso del testo e che deve essere posizionata relativamente alla pagina corrente.
- ***Larghezza tabella***: specifica la larghezza predefinita della griglia (si tratta di un parametro dell'algoritmo di layout indicato nelle proprietà di tabella inerenti).

A livello di riga troviamo:

- ***Riga tabella "not break" a piè di pagina***: indica che il contenuto della riga non può essere suddiviso su due pagine. Se necessario il contenuto della riga viene traslato all'inizio di una nuova pagina.
- ***Spaziatura tra celle della riga***: è possibile specificare lo spazio di default (lo spazio che intercorre tra le celle adiacenti ed i bordi della tabella) per le celle di una riga.
- ***Intestazioni***: imposta una riga come intestazione. In questo modo si ripete il suo contenuto su ogni pagina della tabella.
- ***Altezza***: è possibile indicare la profondità di una riga.

A livello di cella:

- ***Confine***: specifica i confini di una singola cella della tabella.
- ***"Cell Shading"***: specifica l'ombreggiatura di una singola cella.
- ***Inserimento Testo***: specifica la spaziatura tra caratteri e deve essere ridotto o aumentato in modo da adeguare la larghezza del testo corrente della cella.
- ***Margini cella***: specifica i margini di una cella della tabella.
- ***Larghezza cella***: indica la larghezza preferita di una cella della tabella.
- ***Orientamento testo***: specifica la direzione di un contenuto della cella: basso in alto, sinistra per destra, ruotato, da destra a sinistra.
- ***Allineamento verticale***: specifica l'allineamento verticale del contenuto di una cella.
- ***Giustificazione***: specifica la giustificazione del contenuto della cella: verticale, centrale, alto e basso.

A livello di formattazione condizionale, uno stile di tabella può definire alcune caratteristiche che possono essere applicate ad alcune regioni.

All'interno della tabella, è possibile indicare quali regioni devono acquisire queste proprietà.

Le regioni includono:

- Raggruppamenti righe n. dispari.
- Raggruppamenti colonne n. dispari.
- Raggruppamenti di righe numerate.
- Raggruppamenti di colonne numerate.
- Prima colonna.
- Prima riga.
- Ultima riga.
- Ultima colonna.
- Cella superiore desta.
- Cella superiore sinistra.
- Cella inferiore destra.
- Cella inferiore sinistra.
- Intera tabella.

2.2.5. Liste

Le liste non sono rappresentate attraverso elementi specifici all'interno del documento, ma vengono espressi attraverso paragrafi associati ad una numerazione.
Tale associazione indica che il punto è un elemento dell'elenco. Se due o più paragrafi sono associati alla medesima numerazione, allora è possibile affermare che essi appartengono alla stessa lista. Elenchi a più livelli sono ammessi **[MV07a]**. Il livello di un paragrafo all'interno di una numerazione viene specificato all'interno della proprietà inerenti. WordprocessingML accetta nove livelli di numerazione. Tali informazioni sono memorizzate all'interno di una singola parte del pacchetto, il quale fa riferimento al documento stesso. E' possibile specificare le proprietà di formattazione dei livelli di una singola numerazione. Ad esempio, se un comma dichiara di essere al livello k, allora è formattato utilizzando le proprietà di formattazione definite per il livello k.
Per ognuno dei nove livelli, è possibile specificare le seguenti proprietà:

- ***Giustificazione***: il simbolo dell'elemento viene giustificato comparendo di fronte al contenuto dell'elenco delle voci.
- ***Immagine***: è possibile associare un'immagine come simbolo di numerazione. L'immagine è definita utilizzando la sintassi VML.
- ***Valore di partenza***: il valore da utilizzare per il primo elemento di una lista di tale livello.

- ***Formato di numerazione***: specifica il formato numerico che deve essere utilizzato per tutte le numerazioni a quel livello. I possibili valori sono:

    - Cifre arabe Abjad, per l'arabo di numerazione.
    - Alfabeto arabo, per numerare i paragrafi con lettere dell'alfabeto Arabo.
    - Elenchi puntati, per indicare che i punti devono essere numerati con "proiettili".
    - Sistema di conteggio cinese, precisa che la numerazione deve utilizzare il sistema di conteggio cinese ascendete.
    - Numerazione decimale, precisa che la numerazione deve usare i numeri decimali.
    - Caratteri Latini, precisa che la numerazione deve utilizzare i caratteri latini.
    - Numerazione Romana, specifica che la numerazione deve utilizzare i numeri Romani.

- ***Spaziatura***: è possibile specificare il carattere di spazio da utilizzare tra il simbolo di numerazione e il contenuto del paragrafo.

2.2.6. Sezioni

Un documento word è costituito da una sequenza di sezioni, ciascuna contiene i contenuti a livello di blocco. Una sezione viene utilizzata per specificare alcune proprietà di formattazione applicabili a una regione di contenuto. Quest'ultimi in genere sono requisiti di formattazione (come anche alcune proprietà che specificano l'aspetto predefinito e il comportamento dei paragrafi della sezione).
Le sezioni non vengono memorizzate in modo nativo da WordProcessingML ma vengono registrate all'interno delle proprietà dell'ultimo paragrafo della sezione presa in considerazione **[MV07a]**.
Qui di seguito ne riportiamo le caratteristiche peculiari:

- ***Sezione layout verso sinistra***: indica che le proprietà a livello di sezione devono essere visualizzate da destra verso sinistra. Tale proprietà non influisce sul testo contenuto all'interno della sezione.
- ***Definizione di colonne***: specifica il numero di colonne con cui disporre le sezioni. Specifica inoltre la larghezza di ciascuna colonna e la spaziatura che vi intercorre.
- ***Griglia documento***: definisce la griglia da utilizzare per visualizzare con precisione i contenuti est-asiatici della sezione.

- ***End note Properties***: una collezione di proprietà di formattazione relative alle note di chiusura della sezione. Tra le proprietà incontriamo:
    - Formato numerazione endnote, specifica il formato da utilizzare per la nota finale di numerazione.
    - Valore iniziale, questa proprietà specifica il valore di partenza per le note di chiusura di numerazione.
    - Note di chiusura di posizione, dove le note di chiusura possono essere poste (sia alla fine del documento, sia alla fine della sezione).

- ***Piè di pagina di riferimento***: è possibile specificare quello che deve essere visualizzato come piè di pagina. Possiamo distinguere tra i piè di pagina per la prima pagina, piè di pagina per pagine dispari e per pagine pari. Non è definito all'interno del documento principale ma è un riferimento ad un risorsa esterna contenuta nel package a cui il documento stesso può accedere.
- ***Header di riferimento***: analogo al piè di pagina di riferimento.
- ***Accesso ai moduli dei campi***: specifica che il contenuto della sezione non è editabile, tranne per i campi di controllo.
- ***Numerazione righe***: definisce l'impostazione da utilizzare per il numero delle linee della sezione.
- ***Bordi pagina***: definisce i confini da visualizzare in ogni pagina della sezione.
- ***Impostazione numerazione pagina***: identifica le impostazioni da utilizzare per visualizzare i numeri di pagina.
- ***Dimensioni della pagina***: questa è una proprietà complessa che definisce, per ogni pagina:
    - Larghezza sezione.
    - Altezza sezione.
    - Orientamento (verticale o orizzontale) della sezione.

- ***Direzione testo***: specifica la direzione del flusso per il testo della sezione.

2.2.7. Media Objects

Tra i percorsi delle corse, è possibile indicare la presenza di un oggetto multimediale**[MV07a]**.
In particolare è possibile specificare la presenza degli oggetti seguenti:

- ***Drawing ML***: oggetto definito sia "inline" che "galleggiante". Usato per rappresentare le forme e altri oggetti grafici all'interno del documento.

- ***Video***: specifica un film, viene riprodotto su richiesta dell'utente. La presenza di video incorporati è definita attraverso oggetti VML.
- ***Embedded***: è un oggetto "inline" generico. Le sue proprietà di formattazione vengono definite utilizzando la sintassi VML.
- ***VML***: è un oggetto "inline" generico, è usato per descrivere formati per grafica vettoriale.

2.2.8. Campi e collegamenti ipertestuali

I campi sono utilizzati per rappresentare contenuti dinamici, sono semplici o complessi. Un campo semplice viene utilizzato quando il risultato (del campo) viene visualizzato dall'interpretazione con un insieme comune di proprietà di formattazione. In caso contrario, se una parte del risultato deve essere
formattato secondo alcune proprietà, e altre parti secondo altre proprietà, devono essere utilizzati i campi complessi. Un campo complesso è suddiviso in parti diverse e contigue, il suo inizio e fine sono contrassegnati da elementi speciali. I campi sono utilizzati per visualizzare informazioni dinamiche.
Un elenco parziale dei campi:

- Data e ora corrente.
- Dimensioni del file.
- Numero di caratteri.
- Commenti sul contenuto del documento.
- Creazione data.
- Parola chiave.
- Data ultima stampa.
- Numero di linee.
- Numero di pagine.
- Numero di parole.
- Titolo.
- Formule: definisce formule aritmetiche complesse possibilmente con riferimenti contenuti nelle celle di una tabella.
- Modulo campi:
    - Checkbox.
    - Elenco a discesa.
    - Riquadro.

- Bibliografia.
- Collegamento ipertestuale.

- Inclusione testo, un riferimento ad un documento esterno (non necessariamente ad un documento WordProcessingML).
- Riferimento di pagina: visualizza il numero della pagina che contiene il segnalibro di riferimento.

Un collegamento ipertestuale può anche essere definito usando un elemento specifico ovvero <w:Hyperlink>. WordProcessingML permette l'inserimento di script (macro) all'interno del documento **[MV07a]**. Si tratta di una definizione ottenuta mediante alcune proprietà del campo.

2.2.9. Intestazioni e Piè di pagina

Intestazioni e piè di pagina si riferiscono al testo, grafica o dati che possono essere visualizzati nella parte superiore o inferiore di ogni pagina di un documento WordProcessingML.
Come già discusso, ci sono tre diversi tipi di intestazioni e piè di pagina:

- Prima dell' Header e il footer della pagina.
- Numerazione dispari dell'header e il footer della pagina.
- Numerazione pari dell'header e il footer della pagina.

Ogni sezione si riferisce a diversi tipi di header e footer**[MV07a]**. Tuttavia, le intestazioni e piè di pagina non sono definite all'interno di una sezione. Essi sono definiti in alcune parti del pacchetto WordProcessingML a cui fa riferimento il documento principale. Più precisamente, il piè di pagina fa riferimento a "footer.docx, mentre l'header fa riferimento a "header.docx.

2.2.10. Indice

Il sommario di un documento può essere generato utilizzando i campi "TC" e "TOC". Il primo definisce il testo e il numero di pagina per una voce di sommario. Il secondo è utilizzato per costruire la tabella di contenuti e utilizza le informazioni specificate dai campi "TC" **[MV07a]**.

2.3. Struttura Package OpenDocument

Dopo aver spiegato in dettaglio il formato WordProcessingML, l'estensione "odt" rappresenta il documento di testo in Open Office. Andremo a paragonare gli elementi di composizione della struttura (dove possibile) con quelli dei documenti word sopra descritti.

L'OpenDocument package è un formato aperto basato su specifiche OASIS e sfrutta tecnologie quali ZIP e XML. Rappresenta i suoi formati, quali documenti di testo, fogli di calcolo, presentazioni e grafica, attraverso un pacchetto ZIP contenente un insieme di file. Quest'ultimo appare meno strutturato del package Office OpenXML.
I file che possiamo trovare all'interno del package sono:

- ***Content***: Contiene il documento principale.
- ***Meta***: Include le meta informazioni del documento.
- ***Settings***: Indica le proprietà di configurazione del documento.
- ***Styles***: Rappresenta le definizioni di stile definendo le proprietà di formattazione da applicare al documento.
- ***Manifest***: Definisce il mime-type per i file del package.

Come possiamo notare dalla lista dei file, il package ODT separa in maniera netta e completa gli elementi implementati dallo standard tra i quali: contenuti, proprietà di formattazione, parametri d'utente, immagini, macro e oggetti binari.

2.3.1. Struttura documento ODT

Un documento di testo "odt" è strutturato anch'esso ad albero, gli elementi che compongono tale struttura sono simili a quelli che caratterizzano un documento WordProcessingML.
Un tipico albero dovrebbe comprendere:

<***office:document-content***> è la radice del documento.

<***office:body***> definisce il corpo del documento e può contenere al suo interno uno o più elementi <office-text>.

<***office-text***> definisce una regione di testo e può contenere al suo interno uno o più <text:p>.

<***text:p***> dichiara l'elemento paragrafo e può contenere uno o più elementi strutturati quali <text:list> e/o <text:section>.

<***text:list***> definisce l'elemento lista; può contenere l'elemento <text:list-item> utilizzato per descrivere i valori che compongono la lista.

<***text:section***> dichiara una sezione nel documento e può contenere al suo interno uno o più elementi <text:p>.

<table:table> definisce una tabella, al suo intero possiamo trovare <table:table-row> che rappresenta la riga della tabella e <table:table-cell> che definisce una cella. All'interno di queste si possono trovare uno o più elementi <text:p>

Anche OpenDocument sfrutta standard per funzionalità non implementate nativamente.

Gli standard supportati sono **[BDEFMV05]**:
- ***Dublin Core***: utilizzato per la descrizione dei metadati (informazioni bibliografiche) all'interno del documento.
- ***MathML***: sfruttato per la definizione di formule matematiche.
- ***SVG***: per la grafica vettoriale.
- ***SMIL***: per contenuti multimediali.

2.3.2. Paragrafi, intestazioni e testo

I paragrafi e le intestazioni in OpenDocument sono rappresentati da elementi diversi:

- <text:p> descritto precedentemente.
- <text:h> ha un attributo "text:level" il cui valore interno descrive il livello di una sola linea.

Entrambi in WordProcessingML vengono descritti da un elemento unico. Le specifiche OpenDocument consentono di attribuire un set di caratteristiche ai paragrafi; i possibili valori sono:

- ***Line-height***: specifica una linea di altezza fissa, questa proprietà sovrascrive il normale calcolo dell'altezza della linea.
- ***Line-height-at-least***: è una lunghezza che specifica l'altezza minima della linea.
- ***Line-spacing***: specifica una distanza fissa tra le righe di un paragrafo.
- ***Text-align & text-align-last***: specifica l'allineamento del testo.
- ***Margin-left & margin-right & text-indent***: indica il rientro da destra, da sinistra e l'indentazione del testo.

- ***Margin-top & margin bottom***: descrivono il controllo della spaziatura prima e dopo un paragrafo. Il valore di questi attributi è una lunghezza o una percentuale relativa allo stile padre.
- ***Break-before & break-after***: indica se mettere una colonna o un interruzione di pagina primo o dopo il paragrafo. Il valore predefinito è "auto", consente all'applicazione di prendere la decisione se inserire una pausa prima del testo.
- ***Orphans & windows***: fornisce il numero minimo di righe di un paragrafo prima e dopo una interruzione di pagina.
- ***Border-right & border-bottom***: definiscono i bordi del paragrafo per ognuno dei quali è possibile definire:

    - Larghezza: specifica la dimensione delle parole che possono essere spesse o sottili.
    - Stile: i confini possono essere trasparenti, solidi o doppi.
    - Colore: è una cifra esadecimale che rappresenta il valore del colore.

- ***Border-line-width & border-width-side***: indica la spaziatura delle linee specificando la larghezza della linea interna, lo spazio tra le righe e la larghezza della linea esterna.

Alcune delle proprietà che possono essere assegnate ad un paragrafo WordprocessingML e che non possono essere assegnate in OpenDocument sono:

- Regolazione automatica della spaziatura del testo latino e Est-Asia.
- Correzione automatica rientro destro durante l'utilizzo di griglia documento.
- Ignora spaziatura sopra e sotto quando si utilizzano gli "indent".
- Mirroring Rientri.
- Ombreggiatura paragrafo.

Le proprietà che possono essere assegnate in ODT ma non in WordProcessingML sono:

- Grandezza relativa carattere.
- Larghezza "under-line".
- Sottolineatura modalità parola.
- Angolo di rotazione del testo.
- Scala di rotazione testo.

Il set di caratteri in OpenDocument prevede la famiglia di sillabe e ideogrammi in CJK (cinese-giapponese-coreano).

Le lingue che non hanno la normale interpretazione semplice da sinistra verso destra, devono essere definite nella classe "ComplexTextLayout" tali lingue sono:

- Arabo.
- Ebraico.
- Hindi.
- Thai.

OpenDocument, come di seguito rappresentato, permette di attribuire caratteristiche indipendenti alle lingue presenti nel documento **[O'Re05]**. In questo esempio il testo Asiatico verrà interpretato normalmente, mentre il testo definito come Layout complesso, verrà interpretato in grassetto.

```
<style:style style:name="T5" style:family="text">
    <style:text-properties fo:font-weight="bold"
      style:font-weight-asian="normal"
      style:font-weight-complex="bold"
    </style:style>
```

2.3.3. Sezioni

Le sezioni sono rappresentate sintatticamente dall'elemento <text:section>. Questo elemento per definizione richiede l'attributo "name" e il valore server per identificare la sezione. L'elemento prevede anche un attributo facoltativo ovvero "style-name": questo valore identifica la classe di stili associata, la quale contiene un riferimento alla sezione attraverso <style:section-properties>.
OpenDocument prevede un'insieme comune di proprietà associate alle sezioni. Tali proprietà riguardano principalmente l'organizzazione del testo in colonne e, anche se in WordProcessingML le sezioni vengono utilizzate per definire le proprietà della pagina, non sarebbe sbagliato paragonare tali elementi. Infatti, in WordProcessingML, le sezioni rappresentano un'insieme di proprietà per l'organizzazione dei paragrafi in colonne.
Un esempio che non ha una contro parte è che in ODT le sezioni possono essere nidificate, mentre in WordProcessingML no. L'insieme degli attributi in ODT vengono definiti dall'elemento <style:section-properties> che prevede <style:columns>, il quale indica la suddivisione della pagina in

colonne, per ogni colonna abbiamo <style:column> il quale prevede **[O'Re05]**:

- ***Column-count***: indica il numero di colonne per la suddivisione della pagina.
- ***Column-gap***: indica la spaziatura fra le colonne.
- ***Rel-width***: definisce la larghezza della colonna.
- ***Start-indent & end-indent***: definiscono la spazio fra le colonne con valori assoluti.

  Per separare le colonne esiste un elemento apposito: <style:column-sep>. Quest'ultimo possiede il seguente insieme di attributi :

  - ***Width***: definisce la larghezza della linea di separazione.
  - ***Color***: espresso in esadecimale rappresenta il colore.
  - ***Height***: indica l'altezza della linea di separazione espressa in percentuale.
  - ***Vertical-align:*** rappresenta l'allineamento superiore, centrale e inferiore delle colonne.

  Le proprietà che possono essere definite in WordProcessingML (e non hanno un corrispondete in Odt per le sezioni) sono:

  - Section type.
  - Right to left section layout.
  - Document Grind.
  - Line numbering settings.
  - Page borders, margin.
  - Page numbering setting.

Al contrario, in ODT possono essere definiti i seguenti attributi (che non hanno una contro parte in WordProcessingML):

- ***Line style, width, height and color***: precedentemente descritti.
- ***Conditional-formatting***: è possibile nascondere una sezione nel documento attraverso una valutazione condizionale.

2.3.4. Tabelle

L'elemento <table:table>, che definisce la tabella, presenta una sintassi simile a quella del linguaggio HTML. Infatti, è possibile definire una riga attraverso <table:table-row>, una colonna attraverso <table:table-column>,

oppure una cella attraverso <table:table-cell>. Anche in OpenDocument è possibile attribuire proprietà a livello di tabella, colonna,riga o cella.

A livello di colonna è possibile specificare:

- ***Column-width***: il valore della larghezza delle colonne.
- ***Rel-column-width***: la larghezza del colonne con valori relativi.

A livello di riga è possibile definire:

- ***Row-height***: il valore dell'altezza di riga.
- ***Min-row-height***: il minimo valore con il quale devono essere formattare le righe.

  A livello di cella, attraverso <style:table-cell-properties>, è possibile impostare:

- ***Vertical-align***: l'allineamento superiore, centrale e inferiore delle cella della tabella.
- ***Background-color***: il colore della cella.

A livello di tabella troviamo:

- ***may-break-between-rows***: impone di non dividere la tabella in pagine e colonne.
- ***Keep-with-next***: indica la conservazione del paragrafo successivo.

Sempre a livello di tabella, ma attraverso l'elemento <style:table-properties>, è possibile specificare:

- ***Width***: la larghezza della tabella.
- ***Rel-width***: un larghezza espressa con un valore relativo.
- ***Margin-left***: il rientro da sinitra.
- ***Margin-right***: il rientro da destra.
- ***Margin-top***: il margine superiore.
- ***Align***: l'allineamento superiore, centrale o inferiore della tabella.

Alcune proprietà definite in WordProcessingML (che non hanno una contro parte in OpenDocument) sono:

- Default cell Magin.
- Table indentation.

- Table layout.
- Table look.
- Table overlap.
- Positioning.

Altre proprietà definite in OpenDocument non supportate da WordProcessingML sono:

- ***Page Number***: indica il numero di pagine da utilizzare per rappresentare la tabella.
- ***Table background & background image***: è possibile specificare un colore di sfondo o un immagine.
- ***Break before & after***: inserisce una pagina e/o una pausa prima o dopo la tabella.

2.3.5. Liste

Le liste in OpenDocument sono rappresentate dall'elemento <text:list>. Le specifiche Office OpenXML non prevedono questo elemento in quanto le rappresentano attraverso la nidificazione a livelli dei paragrafi.
ODT, in funzione del tipo di lista che si vuole rappresentare, mette a disposizione diverse implementazioni tra le quali troviamo:

- <text:list-level-style-bullet>: indica una lista generica non numerica.
- <text:list-level-style-number>: definisce una lista ordinata.

Per ogni valore della lista viene definito un elemento <text:list-item> che può avere un paragrafo che racchiude il contenuto testuale. Si possono definire proprietà di formattazione a livello di lista attraverso l'elemento <text:list-style> **[O'Re05]** mentre l'elemento lista sarà riferito attraverso l'attributo "style:name".
Di seguito riportiamo le proprietà di formattazione:

- ***text:level***: indica un numero intero nell'intervallo [1,10] .
- ***text:style***: il valore ha un riferimento al file "style.xml" contenuto nel package del documento.

Per gli elenchi puntati lo standard definisce molti simboli qui di seguito illustrati:

- ***Bullet-char***: un singolo carattere Unicode utilizzato come "pallottola" (ammesso solo per i livelli puntati).
- ***Num-format***: indica il formato del numero.
- ***Num-suffisso***: rappresenta il carattere e/o caratteri da inserire dopo il numero. E' un attributo facoltativo per gli elenchi puntati.
- ***Num-prefix***: indica i/il caratteri/e da inserire prima del numero.

Ad esempio, una lista (a), (b) utilizzerebbe i seguenti attributi:

style:num-prefix ="(" ,
style:number-format="a",
style:num-suffix=")".

2.3.6. Media Objects

Open Office utilizza SVG per la grafica vettoriale e SMIL per i contenuti multimediali. Questi linguaggi di markup estendono le potenzialità grafiche. Per i fotogrammi OpenDocument Office definisce un elemento <draw:frame>**[O'Re05]** non solo ma può contenere uno o più elementi <draw:text-box>.

Di seguito riportiamo le proprietà che si possono attribuire per la formattazione:

- ***Svg:width***: indica la dimensione del frame.
- ***(Svg:x, svg:y)***: indicano le coordinate di rappresentazione (nel caso non siano già state impostate dall'attributo "align").
- ***Svg:z-index***: è utilizzato per l'overlapping di elementi.
- ***Anchor-type***: definisce un'ancora con il quale riferire l'oggetto.
- ***Draw:color-mode***: indica da modalità colore da utilizzare per rappresentare il disegno.
- ***Draw:transparency***: rappresenta la percentuale di trasparenza del disegno.
- ***Mirror***: indica la posizione.
- ***Fo:clip***: indica i valori con i quali effettuare il ritaglio dell' immagine.
- ***Draw:luminance***: indica la percentuale di luminosità.
- ***Draw:gamma***: indica i livelli dei toni della luminosità

2.3.7. Campi e Collegamenti ipertestuali

I collegamenti ipertestuali vengono definiti come <text:a>, mentre OpenDocument permette l'inserimento di valori dinamici all'interno dei campi.

I campi dinamici sono:

- <text:date> per la Data.
- <text:time> per l'ora.
- <text:page-number> specifica il numero di pagine, utilizza dei valori per la pagina corrente, precedente o successiva.

2.3.8. Intestazioni e Piè di pagina

In ODT il contenuto e l'aspetto di intestazioni e piè di pagina sono definiti attraverso le pagine master e la pagina layout.
Le proprietà di formattazione di intestazioni e piè di pagina sono definite nella pagina di layout, mentre il loro contenuto è definito con le pagine master. I layout di pagina e le pagine master vengono memorizzati all'interno del file che definisce gli stili del documento. Per i documenti di testo le intestazioni e i piè di pagina contengono testo normale definito da paragrafi, tabelle, liste, sezioni ecc..
ODT permette di attribuire diverse intestazioni e piè di pagina per pagine di destra e sinistra. Tuttavia, non permette di definire intestazioni e piè di pagina diversi per la prima pagina, come WordprocessingML, al contrario, fa.

2.3.9. Indice

In ODT, per generare una tabella di contenuti comunemente definita come indice è possibile marcare una regione di testo. Siccome per tale scopo queste strutture si sovrappongono, è possibile specificare un profilo di livello della tabella (ovvero le voci dell'indice). Le voci della tabella di contenuti sono costituite da regioni di testo contrassegnate esplicitamente e da intestazioni. E' possibile indicare come generare la tabella e il suo layout. ODT permette di assegnare le informazioni richieste tramite la sintassi XML (cosa che si verifica anche in WordProcessinML, con la differenza che la sintassi risulta essere più complessa).

CAPITOLO 3

STRUMENTI di CONVERSIONE

3.1. OpenXMLODFConverter

Come ci saremo già accorti, i formati documentali appena descritti sono molto differenti. Pur essendo standard, i modelli di documento che utilizzano OpenOffice e Office non sono compatibili – infatti, editare un documento word con Open Office (o viceversa), non ci garantisce la corretta formattazione del documento. Entrambe le tecnologie, forniscono delle soluzioni affinché l'utente non sia limitato ad utilizzare il documento creato con l'applicativo che l'ha generato. Questo avviene attraverso funzionalità che permettono l'utente di salvare il documento digitale in un altro formato o esportarlo per un diverso lettore digitale.
Quando l'applicativo non lo permette possiamo verificare se il software che stiamo utilizzando supporta un componente aggiuntivo che estende le funzionalità di base, oppure, possiamo ricorrere a supporti indipendenti che effettuano queste traduzioni. Adesso introdurremo il "OpenXMLODFConverter", oggetto dell'ambiente di testing, per il quale abbiamo realizzato alcuni test di verifica che descrivono la gestione degli elementi dei formati, marcando i successi e gli insuccessi del processo di conversione.
"OpenXMLODFConverter" è un progetto interno della facoltà di Bologna, lo stato attuale prevede la conversione dal formato "docx" di Office al formato "odt" di Open Office. Lo scopo di questa tesi è stato quello di apportate test sistematici al traduttore, cercando di integrare diverse tipologie di verifica. Le tecniche e gli strumenti utilizzati nella verifica saranno spiegati nel capitolo successivo. Introduciamo la struttura logica del traduttore.

"OpenXMLODFConverter" è una piattaforma basata su due livelli di formalizzazione **[M07]** :

- SOM (Sintattic Object Model) .
- COM (Concettual Object Model).

SOM è il modello di basso livello molto vicino alla sintassi del formato del documento, mentre COM è un modello di alto livello che si basa sulle specifiche del Pentaformato (strutture, metadati e presentazioni).

Al livello COM abbiamo la possibilità di lavorare con concetti di alto livello come le sezioni i capitoli. E' basato sul livello SOM sottostante che a sua volta dipende dal formato del documento di partenza.
Dati due documenti distinti, le loro proprietà SOM saranno probabilmente molto diverse; al contrario le loro organizzazioni concettuali saranno simili. Di conseguenza, l'approccio di conversione risulterà più facile a livello COM in quanto potrebbero condividere dei concetti comuni (cosa che non accade a livello semantico**[M07])**.
Il processo di conversione si basa su cinque passi fondamentali. Al fine di convertire un documento D in un documento D', di formato diverso, la procedura di traduzione effettua i seguenti passi:

- Serializza il documento D in un formato L creando il corrispettivo SOM.
- Crea il corrispondente COM.
- Converte l'elemento COM appena creato nell'elemento COM per il formato L'.
- Ne crea un corrispettivo SOM.
- Serializza il SOM ottenuto nel formato di destinazione L'.

3.1.2. Approcci di conversione OpenXMLODFConverter

La struttura del traduttore, come descritto precedentemente, si presta ad essere suddivisa in un livello concettuale ed uno sintattico. L'approccio concettuale promuove un modello di documento astratto, sul quale si possono mappare entrambi i formati da convertire. Il modello sintattico, al contrario, analizza le caratteristiche del documento per capirne il modello di formato. L'approccio utilizzato dal convertitore è basato sulla segmentazione del documento in cinque dimensioni**[VM09]**. Il "pentaformato" è un pattern sviluppato come progetto di tesi interno alla facoltà di Bologna. Il modello a documento segmentato identifica i componenti più rilevanti del documento, al fine di ricombinare e riutilizzare le parti per diversi domini e applicazioni**[DiI07]**.

Il modello è strutturato cosi come segue:

- ***Contenuto***: sono le informazioni piatte come testo e immagini.
- ***Struttura***: indica il ruolo degli elementi testuali con le loro relazioni, al fine di rendere un testo interpretabile e lavorabile.
- ***Presentazione***: l'insieme delle caratteristiche visive e tipografiche aggiunte per massimizzare l'impatto del documento sui lettori umani. Questo livello cerca di rafforzare quello che è di per sé espresso dai contenuti strutturali.

- ***Comportamento***: le azioni dinamiche degli eventi su un documento, necessario per modellare l'interattività dei contenuti dinamici.
- ***Metadati***: l'insieme delle informazioni bibliografiche del documento.

Una volta che l'elaborato è rappresentato in termini di "pentaformato" possiamo facilmente**[VM09]**:

- Generare lo stesso documento con un formato diverso (conversione di formato semplice).
- Generare un documento con lo stesso contenuto e una presentazione diversa (contenuto reflow).
- Generare una rappresentazione astratta della presentazione indipendente dal contenuto (template).
- Forzare un insieme predefinito e fisso di presentazione, comportamenti o caratteristiche dei metadati (omogeneizzazione).
- Convertire metadati per uno schema predefinito (classificazione).

Il modello "pentaformato" ha bisogno di lavorare ad un livello concettuale. La conversione concettuale si presenta più elastica, in quanto si basa su un modello astratto dove risulta più facile effettuare la traduzione. La conversione sintattica è invece meno flessibile, perché genera un mapping uno a uno**[VM09]** e, nel caso non siano supportate eventuali caratteristica, si deve ricorrere all'ausilio di convenzioni.

3.1.3. Struttura documento OpenXMLODFConverter

Il modello intermedio di "OpenXMLODFConverter", non implementa tutte le categorie di elementi previsti dai formati (il progetto infatti è ancora in via di sviluppo). Il gruppo di ricerca è riuscito a descrivere la struttura degli elementi principali per la conversione da Open XML a OpenDocument.
La struttura logica della TestSuite, argomento del prossimo capitolo, si costruisce proprio a partire da questi elementi. Il convertitore implementa nel suo formato intermedio (COM) due classi di oggetti, la prima per il documento di input da tradurre e la seconda per il documento di output risultante. Le caratteristiche implementate per i formati sono **[VM07b]** :

- Paragrafi.
- Intestazioni.
- Collegamenti ipertestuali espliciti (non vengono definiti campi dinamici).
- Liste.
- Tabelle.

Essendo il progetto ancora in via di sviluppo, non otteniamo sempre il risultato atteso, ragion per cui abbiamo bisogno di apportare dei test per verificare le funzionalità che dovrebbero garantire i requisiti. Attraverso questa tesi, abbiamo cercato di realizzare un strumento di verifica, che organizzasse logicamente i file e che automatizzasse il processo di testing mettendo in risalto statistiche e valutazioni prodotti dal processo di traduzione.

3.2. Open XML / ODF translator add-ins for Office

E' un applicazione sviluppata in collaborazione con:

- Dialogika (Sviluppo e istituzioni europee scenari di test - Germania).
- Sonata-Software (sviluppo e End to End prove di funzionalità - Bangalore, India).
- Microsoft (finanziamenti, Architettura e tecniche di orientamento e di progetto di co-coordinamento).
- Novell (Linux porting di OpenOffice.org e di integrazione).
- Clever Age (Precedentemente Development & Project Management - Francia e Polonia).
- Aztecsoft (Precedentemente End to End prove di funzionalità - India).

L'"add-ins" implementa un traduttore per i formati OpenXML e OpenDocument. Essendo ancora in via di sviluppo non è stata inclusa in modalità nativa tra le funzionalità di Office **[CDS09]**.

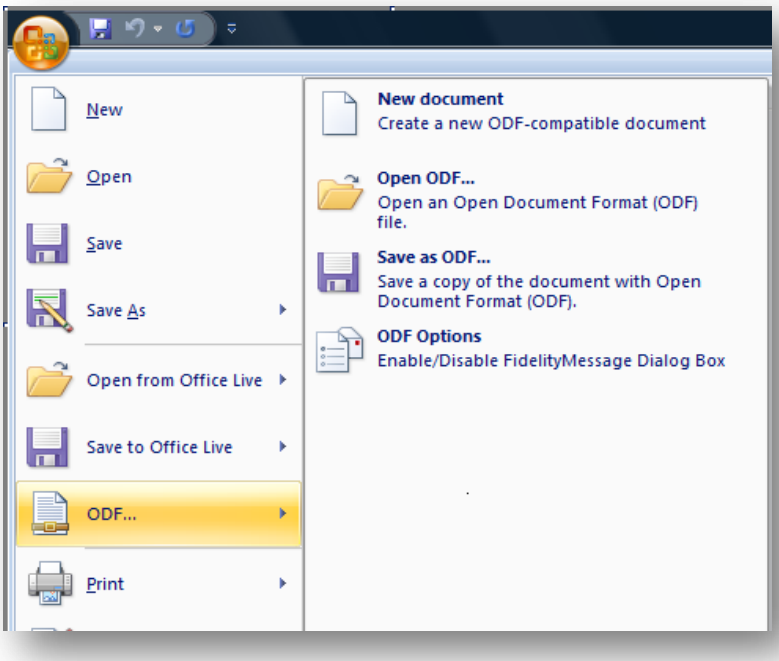


FIGURA 1: add-ins Office

Installato il software come componente aggiuntivo, l'applicazione offre una modalità grafica dell'add-ins come mostrato in figura 1, oppure in modalità batch per eseguire l'applicazione da terminale senza dover utilizzare Office. La piattaforma è basata su trasformazioni XSL tra i due formati XML, supportando i formati word, excel e power point **[CDS09]**.
Il plugin è stato utilizzato dall'ambiente di testing per avere un sopporto nella valutazione dei risultati ottenuti attraverso "OpenXMLODFConverter". L'add-ins offre una soluzione più stabile in quanto è possibile osservare che l'approccio utilizzato produce una perdita d'informazione (effetti, contenuti, oggetti) inferiore rispetto a quella prodotta da "OpenXMLODFConverter". Il motivo è riconducibile alla differenza di tempo che intercorre tra la nascita degli applicativi. "L'add-ins" di Office è un progetto sostenuto già da diversi anni, quindi ha attraversato diverse revisioni e versioni (attraverso fasi di testing) che hanno contribuito al miglioramento del software nel tempo. Gli scenari proposti dal team di sviluppo del plugin hanno attuato principi descritti dall'ingegneria del software per la fase di test. Vengono proposti test funzionali, performance test, test di regressione e bug detection test. Da questi modelli abbiamo realizzato la TestSuite con lo scopo di supportare "OpenXMLODFConverter" nella fase di testing, cercando di aiutare logicamente la supervisione dei risultati e la valutazione di questi. Il capitolo successivo è dedicato a questi modelli.

CAPITOLO 4

PROGETTAZIONE di TEST SUITE

In questo capitolo descriveremo degli scenari di testing, ponendo l'attenzione sulle metodologie proposte dall'ingegneria del software per la fase di verifica, in riferimento ai metodi utilizzati dalla test suite realizzata per "OpenXMLODFConverter".

4.1. Descrizione testing

Spesso la produzione di un software è caratterizzata da un "ciclo di vita". Se il programma prevede una metodologia di sviluppo, allora sarà caratterizzata da diverse attività che garantiranno la correttezza e l'efficienza del prodotto finito. Le attività che descrivono il "ciclo di vita" di un software possono variare in funzione dell'ambito in cui vengono adottate. Infatti molte volte, l'analisi preliminare sul contesto in cui il software dovrà essere inserito costituisce la prima attività che da svolgere. Questa prevede la descrizione dell'analisi di fattibilità, l'analisi e la modellazione del dominio applicativo e l'analisi dei requisiti. L'analisi preliminare e' una fase importante in cui si decidono gli argomenti ritenuti primari. La seconda attività che caratterizza il ciclo di vita è la fase di progetto. E' il momento in cui si definiscono le linee essenziali della struttura da realizzare in funzione dei requisiti determinati nell'analisi. Successivamente si attraversa la fase di implementazione, in cui si realizza concretamente il software: è il momento in cui il team di sviluppo concretizza i meccanismi e realizza algoritmi (descritti con linguaggio previsto per l'implementazione). Successivamente, si attuano le prove di verifica per la validità del prodotto **[IS08]**. L'ingegneria del software definisce questo stadio come "fase di verifica e convalida". Infatti, le attività di verifica e convalida si svolgono dopo ogni fase di sviluppo del software. Queste attività, insieme a quella di debugging[1], sono intercalate, ovvero, quando si trovano errori durante il testing si applicano modifiche di correzione.
Il test (verifica e convalida) e il debugging hanno però obiettivi diversi **[IS07]**:

[1] *Il debuging è un'attività che consiste nella individuazione della porzione di software affetta da errore (bug) rilevati.*

- I processi di verifica e convalida servono per stabilire l'esistenza di difetti in un sistema software;
- Il processo di debugging localizza e corregge questi difetti.

In fine, la fase di manutenzione comprende tutte le attività di modifica successive al suo rilascio. Tale fase viene impiegata per correggere errori non previsti, per applicare dei miglioramenti al prodotto oppure possono servire per adattare il software a nuovi ambienti operativi **[IS08]**. Porremo la nostra attenzione sui processi di verifica e convalida.

4.2. Modelli di testing e tester

Come descritto precedentemente, la fase di testing consiste in una serie di operazioni effettuate durante e dopo lo sviluppo di un software, con lo scopo di esonerare l'utente finale da errori o da malfunzionamenti. L'ingegneria del software descrive due tipi di modelli principali di test generici: test di unità e test di integrazione. I test di unità o dei componenti analizzano i difetti testando i singoli componenti del programma, che possono essere: funzioni, oggetti o componenti riutilizzabili. Mentre i test di integrazione o di sistema analizzano l'integrazione dei sottosistemi composti dai singoli moduli. L'obiettivo del test dei componenti è confermare che il sistema soddisfi i suoi requisiti funzionali e non funzionali, e che non abbia comportamenti imprevisti **[IS07]**. I difetti ignorati in questa fase emergono nei test di sistema. Il processo di testing ha due obiettivi distinti:

- ***Dimostrare allo sviluppatore e al cliente che il software soddisfa i suoi requisiti:*** per il software personalizzato questo significa che dovrebbe esserci almeno un test per ogni requisito utente e di sistema; per i prodotti software generici implica invece che dovrebbero venir effettuati test per tutte le funzionalità del sistema che saranno incorporate nella release **[IS07]**.

- ***Scoprire gli errori o i difetti nel software, dove il comportamento del software è errato, indesiderato o non si conforma alle sue specifiche:*** il test dei difetti ricerca tutti i tipi di comportamento indesiderati del sistema, come crash, interazioni non volute con altri sistemi, calcoli errati e corruzioni dati **[IS07]**.

Il primo obiettivo porta ai test di convalida: controlla che il sistema funzioni correttamente usando un dato insieme di "test case" che riflettono l'uso previsto. Il secondo obiettivo porta ai test dei difetti: i test case sono progettati per rivelare le carenze, possono essere deliberatamente non visibili all'utilizzatore e non hanno bisogno di riflettere il normale modo d'uso del

sistema. Per il test di convalida un test completo con successo esprime che il sistema si sta comportando correttamente, invece per il test dei difetti evidenzia un difetto che porta il sistema a comportarsi in modo errato **[IS07].** I test di verifica, sono scelti attraverso diversi criteri; un mezzo per verificare la qualità dei test può essere rappresentato attraverso un insieme opportuno di casi di verifica. Analizzando più in dettaglio questi casi notiamo:

- *Test Funzionali*: verificano che le specifiche funzionali siano state implementate correttamente, garantiscono la correttezza delle funzionalità **[AM04]**.
- *Test non Funzionali*: si suddividono ulteriormente in:

    - ***Test Prestazionali***: verificano che il sistema soddisfi i requisiti prestazionali richiesti.
    - ***Test di Usabilità***: verificano i requisiti di usabilità, come la facilità di comprensione dell'interfaccia o l'adeguatezza dei contenuti.
    - ***Test di Stress***: ne verificano le performance e i limiti, sottoponendo il programma a diverse prove "estreme" **[AM04]**.

- *Test di non Regressione*: servono a verificare la correttezza delle funzionalità, dopo modifiche effettuate al codice di base.
- *Test di Sistema-Integrazione*: test applicato al sistema finito, utilizzato per convalidare i requisiti funzionali.

Una volta organizzato l'insieme dei test, l'ingegneria del software prevede una fase di automazione dei test stessi. Un workbench di test del software è un'insieme integrato di strumenti per supportare il processo di test. Tra questi troviamo:

- ***Oracolo****:* genera le previsioni dei risultati attesi; gli oracoli possono essere versioni precedenti del programma o sistemi prototipo. Il test back-to-back[2] richiede l'esecuzione parallela dell'oracolo e del programma da testare per evidenziare le differenze nei loro output.

- ***Comparatore di file****:* confronta i risultati dei test del programma con i risultati di test precedenti e ne evidenzia le differenze; i comparatori sono utilizzati nel test di regressione dove si confrontano i risultati dell'esecuzione di diverse versioni. Nell'eventualità in cui sono utilizzati test automatici, quest'ultimo può essere invocato all'interno dei test stessi.

[2] *processo di ottimizzazione di una strategia di negoziazione utilizzando i dati storici per osservare se ha una validità predittiva su dati attuali.*

- ***Generatore di report:*** fornisce le funzioni di definizione e generazione dei report per i risultati dei test **[IS07]**.

Gli strumenti devono essere configurati e adattati al sistema specifico. Possono esserne aggiunti di nuovi per testare specifiche caratteristiche delle applicazioni, oppure è possibile prepararli manualmente attraverso la creazione di un insieme di risultati attesi dai test se non sono disponibili versioni precedenti del programma da utilizzare come oracolo **[IS07].**

CAPITOLO 5

DESCRIZIONE TEST SUITE OpenXML - ODF

In capito descriveremo la struttura interna della test suite distinguendone l'organizzazione logica da quella sintattica. Partendo dalla descrizione di quest'ultime, arriveremo ad analizzare la composizione dei test di verifica attuati dalla TestSuite.

5.1. Progettazione test suite e automazione dei test

Dopo aver introdotto i modelli proposti dall'ingegneria del software sugli ambienti di testing, si passerà alla descrizione della fase di progettazione della test suite, seguita da quella dell'automazione del processo di testing. La test suite, ha lo scopo di evidenziare le caratteristiche implementate in "OpenXMLODFConverter". Attraverso le metodologie descritte precedentemente, vengono riprodotti degli scenari di testing tra i quali: test funzionali, test di performance e test di errore (attraverso scenari proposti per soluzioni simili presenti sulla rete). Quando si testano traduttori di formato i test funzionali possono esser ricondotti alla verifica della conservazione delle proprietà del documento di partenza, ovvero quando traduciamo un formato X in X' ci aspettiamo che i due documenti siano molto simili. I casi in cui dovremmo notare delle anomalie sono quando traduciamo un elemento del documento di partenza che non è supportato dal formato di destinazione. A questo punto, il traduttore può decidere di non considerare l'elemento omettendolo nella traduzione, oppure, può attuare una strategia affinché riesca a trovare una corrispondenza anche se astratta tra gli elementi previsti dai formati. Come descritto nel primo capitolo, i formati documentali sui quali si basa "OpenXMLODFConverter" sono OpenXML e OpenDocument. Dai requisiti del progetto già sappiamo che l'applicativo non gestisce tutte le funzionalità previste dai formati (infatti lo scopo della test suite e quello di proporre una soluzione per facilitare la verifica delle funzionalità). Esse sono intese in termini di risultati visivi, in quanto se traduciamo un documento word che ha un titolo in grassetto, ci aspettiamo che anche nel writer di OpenOffice sia rappresentato come tale.
La test suite non garantisce una soluzione per tutte le funzionalità, ma verifica un insieme di classi divise in relazioni descritte dagli elementi da testare. Infatti, le sue sotto directory sono organizzate logicamente in

contesti, ognuno dei quali offre un set di file di verifica per gli elementi previsti in quel contesto (paragrafi, tabelle, immagini ecc.). Un altro vantaggio che introduce la test suite è il suo riutilizzo; infatti, una volta che viene modificato il sorgente del progetto, si dovranno riapplicare le verifiche (test di regressione). La test suite, grazie a l'aggiunta di report, conserva la descrizione del file precedentemente convertito. Una volta apportate le modifiche alla struttura del convertitore, la test suite può esser riutilizzata con lo scopo di offrire al programmatore uno strumento per confrontare i due risultati (il comparatore di file svolge questo ruolo a livello software), apportando così le proprie valutazioni. Oltre ai test funzionali, la test suite propone dei test di performance, ovvero file word composti da un'insieme significativo di pagine per verificare i tempi impiegati, la gestione degli spazi in memoria e i consumi delle risorse. I test di errore sono le verifiche degli insuccessi del convertitore, quando l'esecuzione fallisce viene aggiornato il report con la notifica dell'insuccesso.

Infine, abbiamo realizzato un tester che automatizzasse la fase di testing. L'applicazione esegue l'esecuzione del convertitore su tutti i file che compongono la suite aggiornando e/o creando il report per documentare l'esito. Il tester promuove delle soluzioni che hanno preso ispirazione dai modelli proposti per l'automazione di testing dell'ingegneria del software. La gestione dei report da parte del tester è una di queste: offre uno strumento per descrivere le caratteristiche che vogliamo testare nella conversione riferendosi al file che attuerà la verifica. Il report se non è previsto nella directory di test, viene generato in seguito alla conversione di un file per descrivere i risultati ottenuti dopo la traduzione. Il tester prevede anche una documentazione globale attraverso una pagina HTML nel quale vengono riportati tutti gli esiti dei file. Tutte le note trascritte nei report vengono annotate nella pagina della documentazione – così lo sviluppatore può avere sempre a vista lo stato complessivo dei test. Il tester non effettua verifiche alle singole unità in quanto non conosce il codice sorgente del convertitore bensì utilizza una politica "Black Box", ovvero considera il software (che si sta testando) come una scatola nera, valutando i comportamenti in base ai parametri d'ingresso e quelli in uscita. Infine, non avendo precedenti test con i quali apportare paragoni, abbiamo utilizzato nel tester l'add-in di office. L'applicazione svolge il compito di oracolo, in quanto il tester dopo l'esecuzione del convertitore (OpenXMLODFConverter), esegue la traduzione sullo stesso documento con il plugin di Office, con lo scopo di avere un supporto di confronto nel valutazione dei risultati. Attraverso i risultati del tester, lo sviluppatore quindi potrà visionare i comportamenti ottenuti per gli stessi test nelle diverse versioni della release.

5.1.2. Organizzazione logica della Test Suite

Dopo un attento processo di analisi sulla struttura del convertire, ci siamo preoccupati di realizzare una suite di test in grado di poter verificare le funzionalità implementate. La Suite dei file di testing è stata scelta ed organizzata in funzione degli elementi descritti precedentemente nel convertitore OpenXMLODFConverter.

Com'è possibile notare in figura 2, la test suite è composta principalmente da sei diversi contesti. La struttura di quest'ultimi è basata sulle funzionalità previste per i documenti word offerte da Office. La prima categoria di elementi che incontriamo è relativa ai caratteri. In questa directory vengono testate le lingue di supporto di word nonché i set di stili per caratteri – per entrambe dei quali vengono analizzati diversi valori ammissibili. Tali valori sono identificati dai file contenuti a quel livello della test suite. Inoltre, sono stati testati proprietà di formattazione relative ai caratteri stessi. Segue la directory relativa a intestazioni e piè di pagina. Anche in questo contesto i file di test hanno la prerogativa di analizzare la struttura di tali elementi al fine di verificare le caratteristiche da questi introdotte (come, ad esempio, collegamenti ipertestuali, formattazioni di intestazioni, griglie di collegamenti ma non solo). Proseguendo nell'analisi logica dei contesti incontriamo la sezione relativa alle illustrazioni. Questa presenta una struttura basata sulle tipologie di figure illustrative offerte da Office Word. Tali categorie posso essere suddivise in forme, linee, frecce, diagrammi e immagini. In ogni sotto-sezione sono stati proposti file contenenti attributi relativi ai suddetti campi. Non solo ma per le immagini sono state definite proprietà relative alla grandezza a al formato.
Successivamente troviamo la sezione dedicata ai paragrafi. Come prima, anche in questa sotto-directory troviamo file relativi alle impostazioni di lista, di rientro e di colore del testo. A loro volta, quest'ultimi includono le caratteristiche del campo preso in esame (ad esempio, la sotto-directory Liste distingue due classi di elenchi a seconda che quest'ultimi siano puntati o numerati). Va notato che questa sotto-directoy conserva i test relativi alle liste – non a caso è stata inclusa all'interno della directory Paragrafo per rispecchiare lo standard del documento di input che appunto prevede la rappresentazione delle liste attraverso le proprietà di paragrafo.
Seguono le sezioni relative alle tabelle. Come sopra, anche qui si riscontra una suddivisione degli attributi in base al font, alle colonne, alle righe e alle celle. Per ognuno di questi è possibile definire la larghezza, l'altezza ma anche il numero.

In ultimo troviamo la sotto-directory relativa al layout di pagina. Quest'ultima individua le caratteristiche inerenti alle interruzioni e alle dimensioni. Mentre le interruzioni si riferiscono alle colonne e alle pagine, le dimensioni fanno riferimento al formato del documento.
L'obiettivo della test suite è stato quello di creare uno scenario di testing organizzato logicamente in funzione della categoria di oggetti da testare in OpenXMLODFConverter. La sua struttura ha lo scopo di semplificare la ricerca logica dei file di testing per il programmatore che vorrà utilizzarla. La struttura stessa verrà riportata in seguito all'esecuzione del tester nella documentazione finale – in modo da rendere logica l'organizzazione della tabella in cui verranno inseriti gli esiti delle prove.

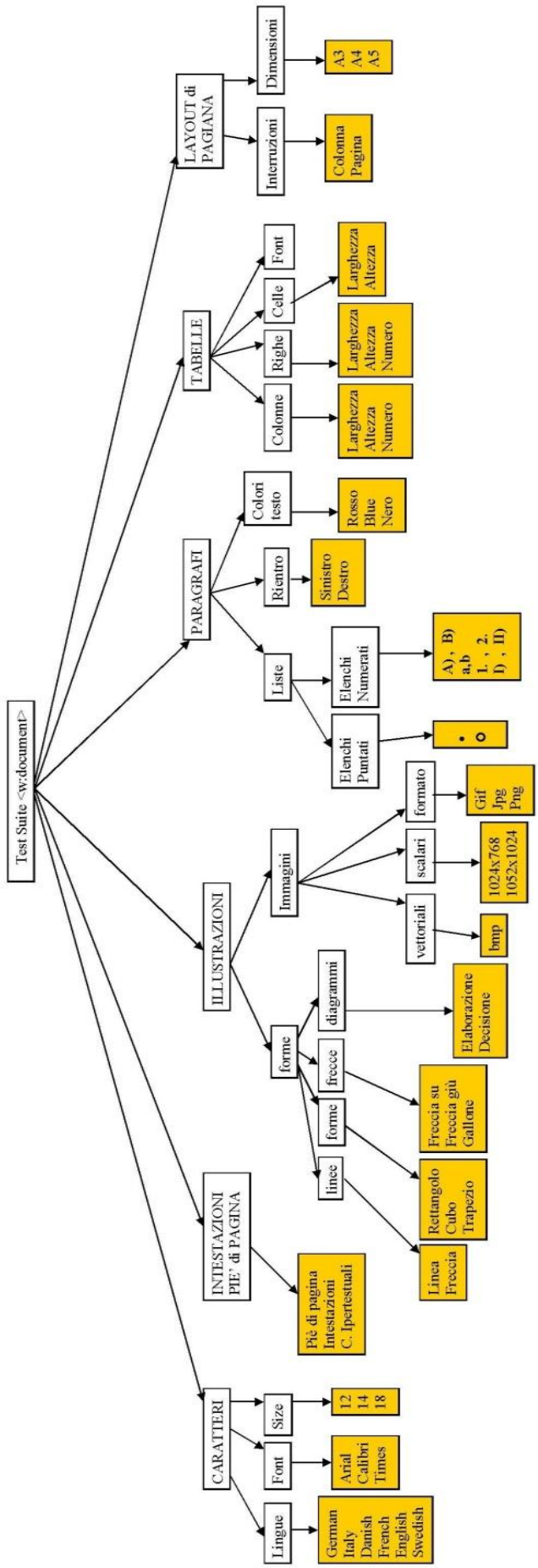


Figura 2: ***Descrizione logica TestSuite***

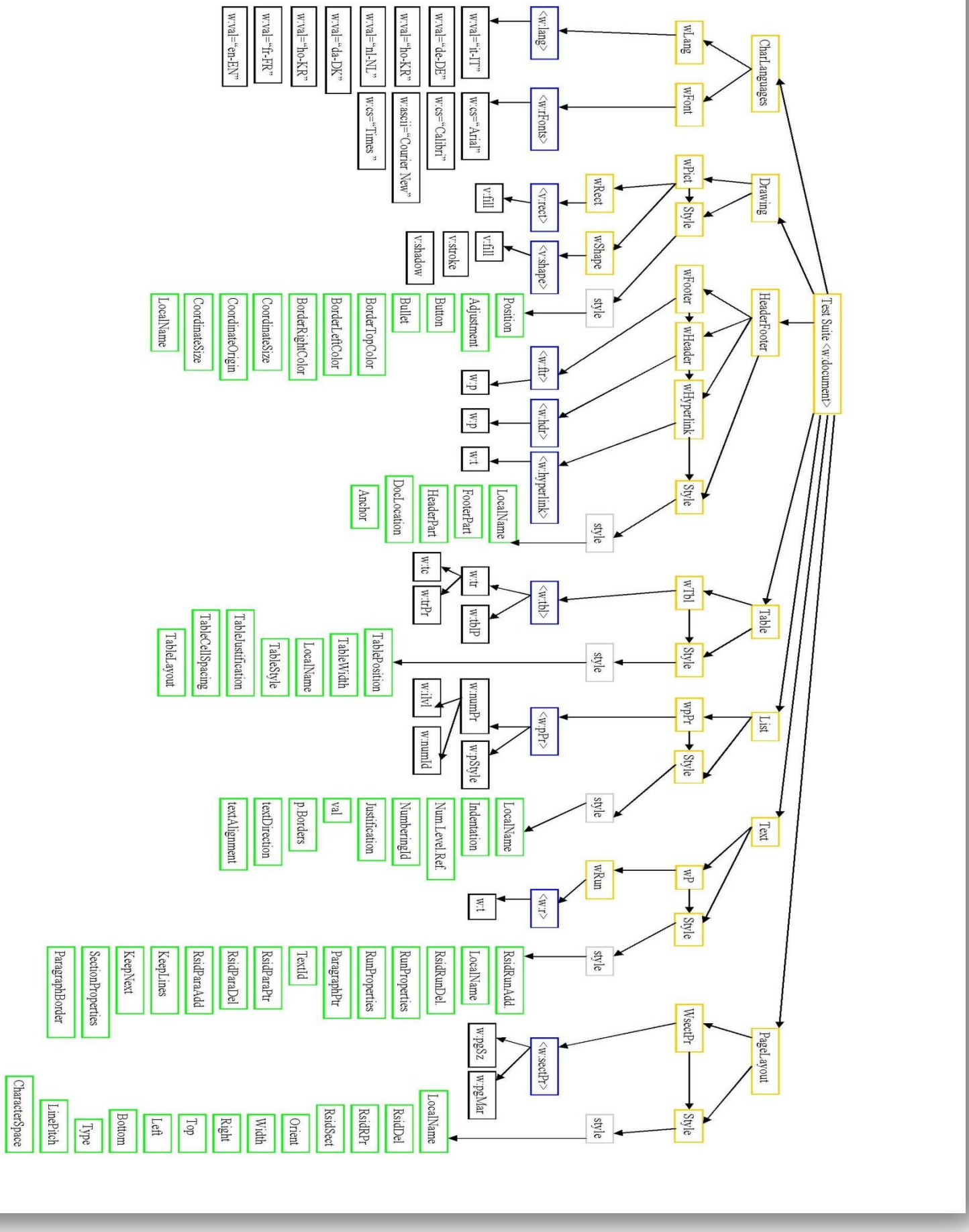


Figura 3: ***Descrizione Sintattica TestSuite***

5.1.3. Organizzazione sintattica della Test Suite

Dopo aver descritto la struttura logica degli elementi testati, la figura 3 ne mostra la configurazione sintattica. I nomi che incontriamo all'interno dei nodi dell'albero sono mappati sulla sintassi di descrizione prevista dallo standard, in riferimento al documento di input WordProcessingML.
Quindi, percorrendo l'albero dalla radice, gli elementi colorati in arancione rappresentano le directory che compongono la Suite mentre gli elementi colorati in blu costituiscono gli oggetti testati all'interno dei file di testing. Ogni figlio del nodo radice può racchiudere al suo interno una sotto-directory Style contenente file che testano gli attributi delle sotto-directory associate.

Il primo livello dell'albero rappresenta la gerarchia delle classi di elementi da testare:

- ***Char*Language:** Set di caratteri e lingue.
- ***Drawing*:** Elementi grafici come linee,curve,forme, immagini.
- ***HeaderFooter*:** Intestazioni e piè di pagina.
- ***Table*:** Tabelle.
- ***List*:** Liste.
- ***Text*:** Paragrafi.
- ***PageLayout*:** Layout e proprietà di formattazione pagina documento.

Va notato che le due organizzazioni differiscono in termini di categorie. Mentre la rappresentazione logica vuole esser d'ausilio per la comprensione della classificazione degli elementi in base alla loro funzionalità prevista da Office, quella sintattica si pone l'obiettivo di dividere i contesti in funzione delle categorie di oggetti definite dallo standard OpenXML. Per ogni nodo di questo livello, il nome attribuito alle directory rappresenta la sintassi del'elemento che vogliamo testare. Nella fattispecie troviamo:

- **CharLanguage**: è lo scenario dei test, effettuati sui caratteri e lingue. Infatti i nodi che comprende al suo interno sono:

    - **wLang**: I file all'interno, testano l'elemento **<w:lang>**.
    - **wFont**: troviamo i test per set di caratteri, l'elemento testato è **<w:rFonts>**.

- **Drawing**: percorrendo questo nodo troviamo **wPict** e Style, questo scenario testa gli elementi disegno, con alcune proprietà. Al suo interno troviamo:

    - **wPict**:
      wRect: testato l'elemento **<w:rect>**
    - **wShape**: testato l'elemento **<w:shape>**
    - **Style**: Proprietà di stile degli oggetti forma

- **HeaderFooter**: Testate proprietà e stili per gli elementi che definiscono le intestazioni e piè di pagina. I nodi al suo interno sono

    - **wHeader**: testato l'elemento **<w:hdr>**
    - **wFooter**: testato l'elemento **<w:ftr>**
    - **wHyperlink**: testato l'elemento **<w:hyperlink>**
    - **Style**: Stili di formattazione per intestazioni e note a piè di pagina.

- **Table:** è il contesto in cui sono stati testati gli elementi sintattici della tabella con alcune delle relative proprietà. Al suo interno troviamo:

    - **wTbl**: testato l'elemento **<w:tbl>**, con i relativi nodi figli **<w:tr>** e **<w:tc>**.
    - **Style**: Testati stili di formattazione per tabelle.

- **List**: in questo nodo vengono testati gli elementi che esprimo le liste. Come descritto precedentemente le liste vengono definite attraverso le proprietà di paragafo. Al suo interno troviamo:

    - **wpPr**: testate proprietà di livello per i paragrafi, attraverso l'elemento **<w:pPr>**, con i relativi nodi figli **<w:numPr>**, **<w:ilvl>** e **<w:numId>**.
    - **Style**: Testati gli stili dei paragrafi espressi da **<w:pStyle>**.

- **Text:** In questo contesto sono stati testati gli elementi paragrafi, corse e le proprietà di tali, logicamente correlati con gli elementi testuali. Al suo interno troviamo:

    - **wP**: Testati proprietà paragrafi per l'elemento **<w:p>** percorrendo il nodo:
    - **wRun**: testato l'elemento **<w:r>**
    - **Style**: Testati gli stili di formattazione per **<w:p>** e **<w:r>**

- **PageLayout**: Testate proprietà e stili di layout di una o più pagine del documento. Sono state testate features e stili per il layout del documento. In questo contesto sono stati testati documenti formati da molte pagine per testare la gestione delle risorse e dello spazio in memoria. Al suo interno troviamo:

    - **WsectPr**: testate proprietà di layout, l'elemento è **<w:sectPr>**.
    - **Style**: testati stili di sezione e layout.

L'insieme dei file analizzati fin ora compongono l'insieme delle verifiche apportate per valutare i requisiti di "OpenXMLODFConverter". I risultati ottenuti saranno discussi successivamente nella sezione del tester.

5.2. Performance Test

Le performance del programma sono state testate attraverso i test di stress. Attraverso questo tipo di verifica, abbiamo analizzato i tempi di traduzione e la gestione delle risorse utilizzate in termini di memoria. L'ambiente hardware e software su cui sono stati effettuati i test comprende:

- ***Processore***: Celeron M 1,40GHz.
- ***Ram***: 1,24 GB.
- ***Sistema Operativo***: Windows XP Home Edition.

La sezione PageLayout (sopra descritta), oltre a i test di verifica per i layout pagine, include diversi file composti da un'insieme significativo di pagine, rappresentano i file che descrivono i performance test.
I file sono:

- test_1496pages.docx
- test_1994pages.docx
- test_14000pages.docx
- test_20000pages.docx

I valori risultanti dalla esecuzione del convertitore "OpenXMLODFConverter" sono stati confrontati con i tempi impiegati dall'add-ins di Office al fine di ottenere un valido metro di giudizio.
La media dei risultati ottenuti dalle verifiche sono stati:

Convertitore OpenXMLODFConverter:

- Cpu utilizzatata 80%
- Memoria utilizzata 108.000KB (omettendo proprietà di stile, verificabile dall TestSuite risultante dal tester).
- Media dei tempi: 4 minuti.

Add-ins Office:

- Cpu utilizzata 97%
- Memoria utilizzata 166.300KB (gestite tutte le proprietà di stile).
- Media dei tempi: 4 minuti.

La gestione delle risorse del convertitore non discosta molto da quella dell'add-ins di Office. E' possibile verificare però che nei file risultanti dal processo di conversione vi siano molte differenze, attribuibili alla gestione della funzionalità delle applicazioni.

5.3. Bug Detection Test

Questo tipo di verifica verte a cercare le situazioni in cui il convertitore termina senza produrre il risultato. Infatti, la documentazione generata dal tester, evidenzia in rosso, nello schema finale dei risultati, tutte le esecuzioni che non sono andate a buon fine, nel ultimo capitolo saranno descritte più dettagliatamente.

CAPITOLO 6

DESCRIZIONE SISTEMA OoTESTER

In questo capitolo parlerò delle funzionalità del sistema, la configurazione e l'esecuzione del tester descrivendo le politiche utilizzate e le prestazione attese ed ottenute.

6.1. Descrizioni generali su OoTester

I tester di verifica forniscono un valido strumento di prova per il programmatore. Le politiche che adottano i tester possono essere Strutturali o Comportamentali. Le politiche Strutturali vengono adottati dai tester in grado di verificare le singole unità della struttura. Nei contesti di testing, sono definite attraverso il nome di "White Box Testing". Il tester che adotta questa politica generalmente effettua delle verifiche all'interno dei singoli moduli o alle singole unità non solo ma controlla la logica della struttura interna e del del codice. Il tester che utilizza questa strategia deve avere conoscenze dettagliate sul codice e può sfruttare altri tester per esaminare la struttura interna al fine di scoprire quali unità, istruzioni o blocchi non funzionano correttamente **[R07a]**. Le politiche comportamentali invece vengono adottate nelle verifiche funzionali. Questi vengono definiti "Black Box Testing", ovvero, si verifica il programma considerandolo come una scatola nera, il tester non conosce la struttura interna, ma analizza i risultati in funzione dei dati di input applicati. OoTester adotta questa strategia, ovvero esegue "OpenXMLODFConverter"(senza conoscere il codice), con lo scopo di apportare valutazioni sui risultati ottenuti segnalando nella documentazione i casi di failure[3]. OoTester è il nome apportato al tester. L'acronimo "Oo" indica i formati che interagiscono con le applicazioni interne, ovvero OpenXML e OpenDocument. Volendo descrivere le funzionalità del tester, OoTester prevede come passaggio di parametri un documento Office word oppure una directory di documenti word. Nel primo caso convertirà il documento con il traduttore OpenXMLODFConverter e con l'add-in di Office generando una directory all'interno del percorso corrente comprendente i risultati delle traduzioni, il documento di input e un report XML, documentando il successo o l'insuccesso della conversione. Nel secondo caso, il tester prevede due input: la directory dei Test e quella di destinazione. L'applicazione traduce tutti i nodi della directory riproponendo

[3] *Il termine failure si riferisce allo stato o condizione di insuccesso.*

lo stesso file system nel percorso scelto dall'utente. Questa directory deve essere la TestSuite descritta precedentemente. Ogni sotto directory al suo interno prevede un documento "docx" che rappresenta il file di testing. Può prevedere anche un report XML, che descrive le caratteristiche del file. Se questo report esiste verrà riportato nella directory risultante e sarà modificato in funzione dei risultati. Altrimenti, verrà generato dal tester, documentando l'esito della prova. Infine, OoTester offre la possibilità di generare la documentazione HTML, dove vengono riportati tutti gli esiti dei file testati, con le relative statistiche.

6.2. Installazione dell'ambiente

Il processo di installazione del tester è composto da due fasi:

- **Copia**: prevede la copia della directory del tester in una posizione scelta dall'utente.
- **Configurazione**: consiste nel settaggio del file di configurazione denominato "pathConf.xml". Viene configurato il percorso dell'interprete java con il quale sarà eseguito il convertitore unibo. Di seguito riportiamo la struttura del file "pathConf.xml".

```
<PATH>
   <elem name="java"src="C:/Programmi/Java/jdk1.6.0_18/bin/java.exe" />
</PATH>
```

Il file viene utilizzato anche temporaneamente durante l'esecuzione del tester per salvare i percorsi di input e di output scelti dall'utente.

6.2.1. Esecuzione del tester

Una volta eseguito il tester, possiamo scegliere la modalità grafica o quella batch[4]. L'interfaccia grafica semplifica solo il passaggio di parametri di input e la creazione della documentazione.

[4] In informatica, il termine batch viene utilizzato per riferirsi a un insieme di comandi o programmi.

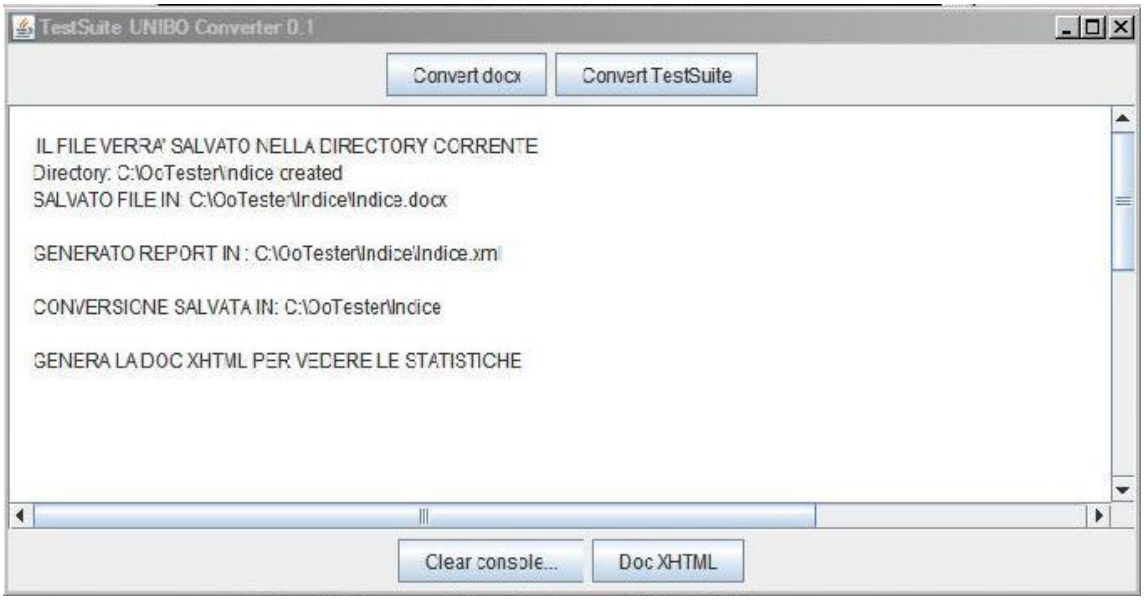


FIGURA 3: modalità grafica

Come possiamo notare in figura 3, attraverso l'interfaccia, è possibile caricare un documento singolo con il metodo "Convert docx", oppure caricare la test suite di file attraverso "Convert TestSuite". Nel primo caso, verrà convertito il documento prima utilizzando "OpenXMLODFConverter" e successivamente attraverso la modalità batch dell'Add-in di Office, generando così una directory all'interno del percorso corrente. La directory risultante comprenderà:

- Il documento di input "docx".
- Il documento risultante dalla conversione "OpenXMLODFConverter", denominato "resUnibo.odt".
- Il documento risultante dalla conversione dell'add-in di Office, denominato "resPlugOff.odt".
- Il Report XML.

Analizziamo ora la seconda modalità di passaggio dei parametri – attraverso la modalità batch.

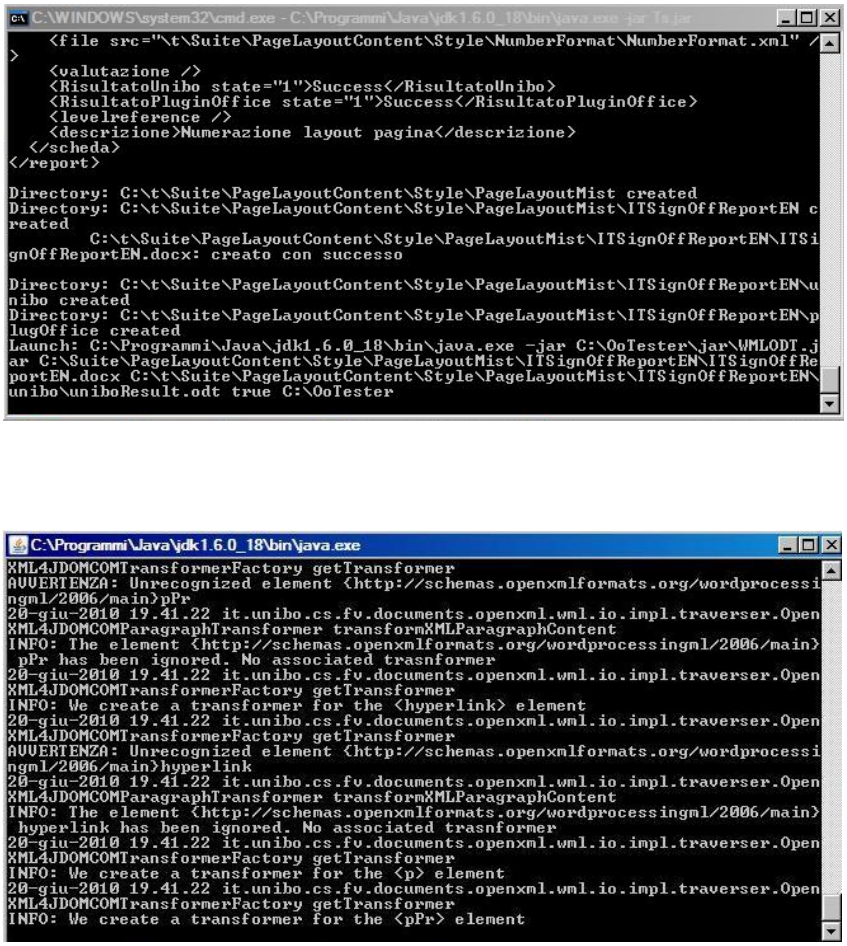


FIGURA 4: modalità batch

Quando si esegue il tester per convertire la Test Suite, l'applicazione ha bisogno del percorso della directory dei test e la destinazione del risultato. In entrambe le modalità, il tester offre le medesime funzionalità. Infatti, come è possibile notare in figura 4, una volta che l'utente sceglie i percorsi, il tester effettua le seguenti operazioni:

- Recupera ricorsivamente le directory presenti all'interno.
- Per ogni directory, crea nel percorso scelto dall'utente la medesima directory.
- Salva il documento word e il report XML contenuti all'interno.
- Se il report non esiste, lo genera.
- Aggiorna l'indice dei file (" index.xml " contiene i link dei file testati).
- Esegue "OpenXMLODFConverter" sul documento word.
- Se il convertitore produce il risultato viene salvato nella directory scelta.
- Esegue l'add-in di Office sul documento word.
- Se l'add-in produce il risultato, questo viene salvato nella directory scelta.

- Verifica se i traduttori hanno prodotto il risultato, sia nel caso positivo che in quello negativo si aggiorna il report con le valutazioni della conversione.

L'output del tester conterrà:

- Documento word
- Risultato "OpenXMLODFConverter"
- Risultato Add-in office
- Repor XML

CAPITOLO 7

DOCUMENTAZIONE E VALUTAZIONE DEI RISULTATI

Oggetto di questo capitolo è l'analisi critica della documentazione generata da OoTeser in seguito ai test effettuati sulla test suite presa in considerazione descrivendo le valutazioni ottenute dai risultati.

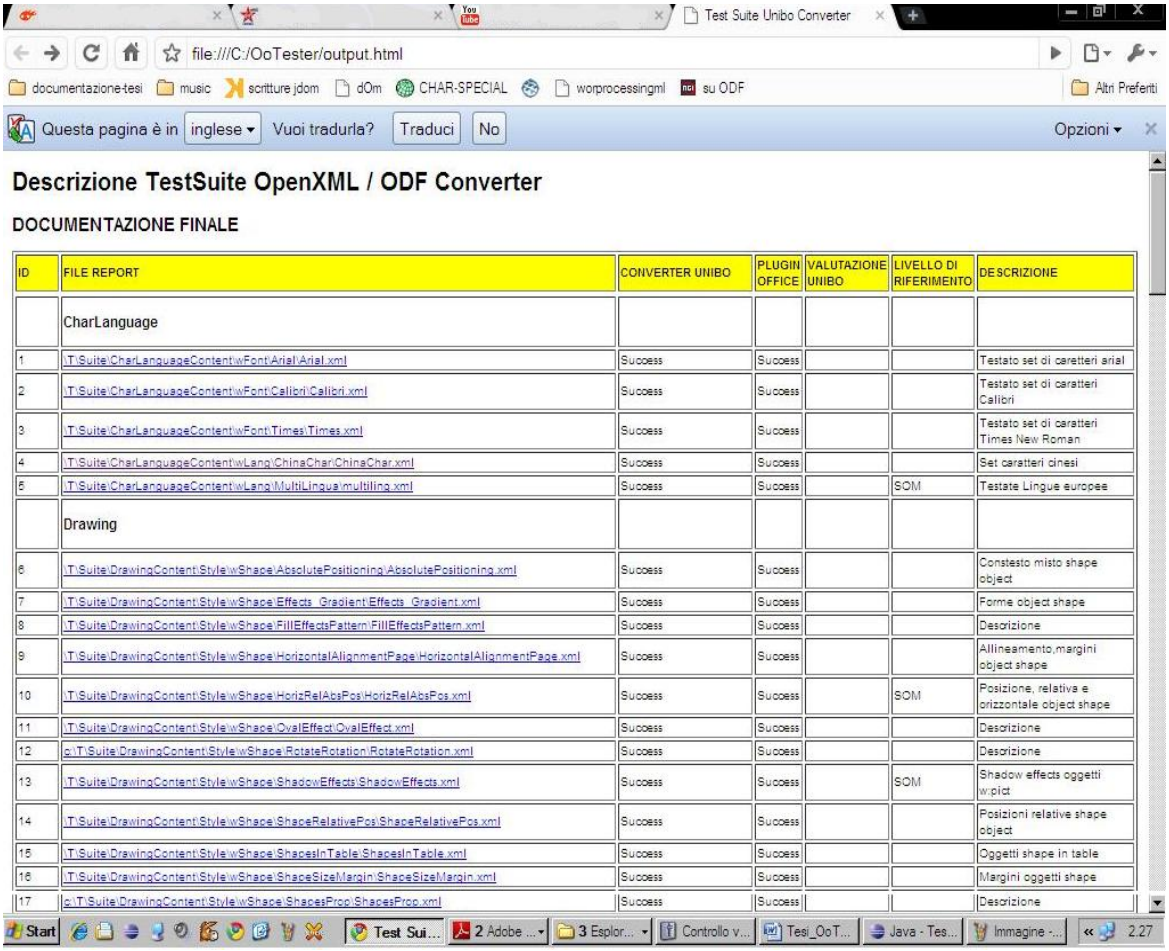
7.1. Documentazione

Il tester prevede anche una funzionalità per generare la documentazione. Quest'ultima viene creata in funzione dei report previsti dalla TestSuite. I report sono stati pensati per descrivere il ruolo del documento word, situato ad un determinato livello della directory. Il programmatore potrebbe volere annotare dei riscontri visivi; cosi facendo in un secondo momento potrà ritrovarli nella documentazione finale.
Il report è strutturato nel seguente modo:

```
<report>
  <scheda id="1">
    <file src="\CharLanguage\wFont\Arial\Arial.xml" />
    <valutazione />
    <RisultatoUnibo state="1">Success</RisultatoUnibo>
    <RisultatoPluginOffice state="1">Success</RisultatoPluginOffice>
    <levelreference />
    <descrizione>Testato set di caretteri arial</descrizione>
  </scheda>
</report>
```

Il primo elemento descrive l'identificativo del report, il campo viene aggiornato dal tester quando si verifica il documento a cui fa riferimento il report. Continuando troviamo il percorso del report, il campo valutazione è stato previsto per note di valutazione che il programmatore potrebbe imprimere attraverso riscontri visivi. I campi Risultato vengono aggiornati se il report esiste nella directory che si sta testando; se invece non era presente, allora vengono generati nel momento in cui si crea il report, verificando lo stato del risultato della conversione. "Level reference" è un campo che indica

il livello di riferimento dell'oggetto testato nel documento, in relazione al codice del convertitore previsto per dettagli tecnici. Infine il campo descrizione include una breve esposizione delle caratteristiche del file. Questo campo è utilizzato dalla documentazione per capire se il report è stato inserito dall'utente o se è stato generato dal tester. Il tester genera il campo con una stringa di caratteri di default. Se non è stato modificato, nella documentazione vengono ricordati il numero di report da aggiornare.



Test Suite Unibo Converter

file:///C:/OoTester/output.html

documentazione-tesi | music | scritture jdom | dOm | CHAR-SPECIAL | worprocessingml | su ODF | Altri Preferiti

Questa pagina è in inglese | Vuoi tradurla? | Traduci | No | Opzioni

Descrizione TestSuite OpenXML / ODF Converter

DOCUMENTAZIONE FINALE

ID	FILE REPORT	CONVERTER UNIBO	PLUGIN OFFICE	VALUTAZIONE UNIBO	LIVELLO DI RIFERIMENTO	DESCRIZIONE
	CharLanguage					
1	\T\Suite\CharLanguageContent\wFont\Arial\Arial.xml	Success	Success			Testato set di caretteri arial
2	\T\Suite\CharLanguageContent\wFont\Calibri\Calibri.xml	Success	Success			Testato set di caratteri Calibri
3	\T\Suite\CharLanguageContent\wFont\Times\Times.xml	Success	Success			Testato set di caratteri Times New Roman
4	\T\Suite\CharLanguageContent\wLang\ChinaChar\ChinaChar.xml	Success	Success			Set caratteri cinesi
5	\T\Suite\CharLanguageContent\wLang\MultiLingua\multiling.xml	Success	Success		SOM	Testate Lingue europee
	Drawing					
6	\T\Suite\DrawingContent\Style\wShape\AbsolutePositioning\AbsolutePositioning.xml	Success	Success			Contesto misto shape object
7	\T\Suite\DrawingContent\Style\wShape\Effects_Gradient\Effects_Gradient.xml	Success	Success			Forme object shape
8	\T\Suite\DrawingContent\Style\wShape\FillEffectsPattern\FillEffectsPattern.xml	Success	Success			Descrizione
9	\T\Suite\DrawingContent\Style\wShape\HorizontalAlignmentPage\HorizontalAlignmentPage.xml	Success	Success			Allineamento,margini object shape
10	\T\Suite\DrawingContent\Style\wShape\HorizRelAbsPos\HorizRelAbsPos.xml	Success	Success		SOM	Posizione, relativa e orizzontale object shape
11	\T\Suite\DrawingContent\Style\wShape\OvalEffect\OvalEffect.xml	Success	Success			Descrizione
12	c:\T\Suite\DrawingContent\Style\wShape\RotateRotation\RotateRotation.xml	Success	Success			Descrizione
13	\T\Suite\DrawingContent\Style\wShape\ShadowEffects\ShadowEffects.xml	Success	Success		SOM	Shadow effects oggetti w:pict
14	\T\Suite\DrawingContent\Style\wShape\ShapeRelativePos\ShapeRelativePos.xml	Success	Success			Posizioni relative shape object
15	\T\Suite\DrawingContent\Style\wShape\ShapesInTable\ShapesInTable.xml	Success	Success			Oggetti shape in table
16	\T\Suite\DrawingContent\Style\wShape\ShapeSizeMargin\ShapeSizeMargin.xml	Success	Success			Margini oggetti shape
17	c:\T\Suite\DrawingContent\Style\wShape\ShapesProp\ShapesProp.xml	Success	Success			Descrizione

Start | Test Sui... | 2 Adobe ... | 3 Esplor... | Controllo v... | Tesi_OoT... | Java - Tes... | Immagine -... | 2.27

Figura 5.1: ***documentazione tester***

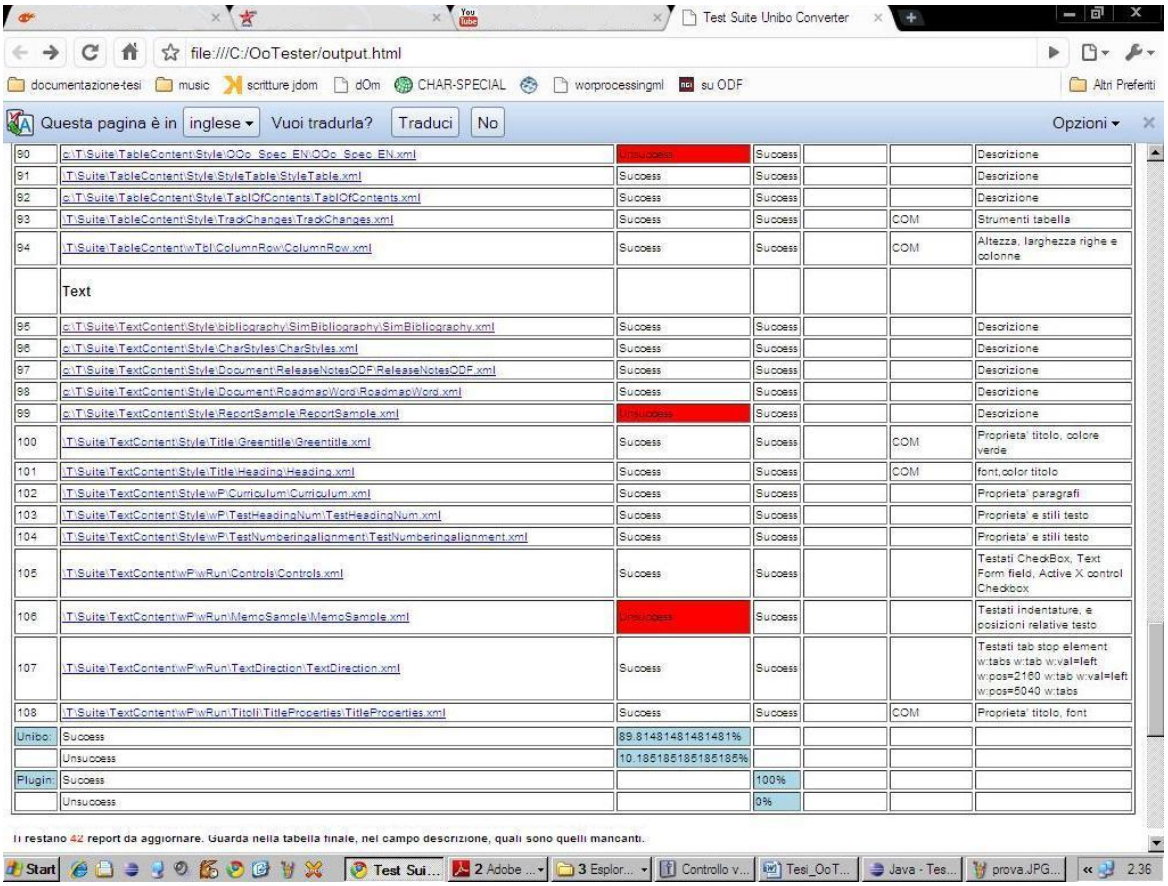


Figura 5.2: ***documentazione tester***

Una volta conclusa la conversione della suite, il programmatore attraverso il metodo "DocHTML" può generare la documentazione HTML sui test effettuati. La documentazione viene generata dinamicamente attraverso l'insieme dei report salvati temporaneamente in "finalItem.xml" a seguiot del quale viene effettuata una trasformazione XSLT (che produce la pagina HTML). Come è possibile vedere in figura 5, la tabella dei risultati è composta dall'identificativo del documento, il percorso del report, la valutazione del successo o insuccesso, valutazioni tecniche e descrizioni. In fine sono riportate le percentuali sull'esito delle traduzioni.

7.2. Valutazione dei Risultati

Il tester ha eseguito le traduzioni per la test Suite composta da 107 file. Le valutazioni ottenute attraverso le traduzioni effettuate con il convertitore unibo hanno visto l'89% percento di successi è l'11% di insuccessi. Ovvero risulta che l'11% di traduzioni non ha generato risultato in quanto in presenza di alcuni elementi previsti dallo standard di input, non essendo

gestiti nel convertitore, arrestano la normale esecuzione di elaborazione. Per il restante 89%, il convertitore ha generato il risultato, terminando normalmente l'esecuzione . I risultati ottenuti però, non si riferiscono al modo con il quale sono stati convertiti. Non di meno, visionando la directory risultante, si nota che il 60% della struttura XML del documento principale di partenza viene tradotta in quella del modello di output, ovvero vengono tradotti gli elementi principali, quali paragrafi, tabelle, liste con alcune proprietà base relative allo stile. Mentre, il 40% della struttura non gestita può essere identificata con un 20% per le caratteristiche di stile omesse nel processo di traduzione e un 20% attribuibile agli elementi del package di OpenXML non gestiti nel convertitore – quali elementi illustrativi, Header e Footer e immagini.

I test di performance di OpenXMLODFConverter sono stati confrontati con quelli del plug-in di Office. Nonostante i due risultati non differiscano sensibilmente l'uno dall'altro, è bene notare che i riscontri visivi del plug-in proprietario si avvicinano con maggiore fedeltà all'originale. Nonostante i tempi di conversione del "convertitore unibo" risultino di alcuni ordini inferiori, è importante notare che una tale velocità è certamente dovuta a un minor carico di elaborazione. Anche nella gestione delle risorse si verifica una tale situazione. Le valutazioni così effettuate vogliono quindi solo essere una base di partenza per verifiche future. La comparazione dettagliata delle applicazione sarà possibile solo quando la release OpenXMLODFConverter offrirà una consistenza migliore nell' implementazione delle sue funzionalità.

CAPITOLO 8

CONCLUSIONI

I formati documentali rappresentano un punto di forza per la creazione e lo scambio di informazioni. Abbiamo visto come scelte diverse di rappresentazione compromettano la normale interpretazione per gli applicativi che la effettuano.

L'esigenze degli utenti spingono sempre più alla realizzazione di nuove tecnologie per sostenere lo scambio e la conservazione dei documenti digitali.

Lo studio di ricerca ha analizzato applicativi per traduzioni di formato, come OpenXMLODFConverter e l'add-ins di Office. I confronti nascono dall'esigenza di avere termini di paragone nella valutazione dei test, per il miglioramento dei risultati. Analizzando gli strumenti e le tecniche su cui si basano gli ambienti di testing, siamo riusciti a realizzare un supporto per OpenXMLODFConverter. Gli ambienti di testing dividono l'insieme delle verifiche da relizzare per il testing, dall'automazione che li effettua. Entrambi si basano su principi di progettazione descritti dalla disciplina dell'ingegneria del software. Questi principi sono stati utilizzati per la creazione dei test di verifica e del tester che li utilizza.

Il supporto dell'ambiente di testing avvantaggia la valutazione anche nel tempo, infatti, abbiamo visto come una semplice documentazione possa descrive i risultati ottenuti, apportando paragoni in funzione delle differenti versioni della release.

In qualunque ambiente di sviluppo software, la necessità di garantire la qualità delle funzionalità comporta una accurata fase di verifica; gli strumenti di testing supportano la valutazione dei requisiti, semplificando il giudizio dei risultati e il miglioramento delle qualità funzionali.

In conclusione, gli strumenti realizzati per il progetto di tesi (tester, insieme di test e documentazione) non hanno l'intenzione di proporre nuove metodologie per la creazione di ambienti di testing. Al contrario, vuole farsi

carico di supportare il processo di verifica di OpenXMLODFConverter durante la fase di sviluppo. Solo riproponendo test sistematici in seno allo sviluppo del progetto – e grazie al supporto di un'applicazione che semplifica la fase di verifica automatizzando l'esecuzione dei test – saremo in grado di verificare l'evoluzione di ogni singola funzionalità constatando come nel tempo questi ottimizzano la comprensione dei risultati e semplificano il lavoro di testing per il programmatore.

BIBLIOGRAFIA

[AM04] A. Abran, J.W.Moore. *Guide to Software Engineering Body of Knowledge. The Institute of Electical and Electronics Engineers, IEEE Computer Society Order Number C2330, ISBN 0-7695-2330-7, Library of Congress Number 20005921729.* Editors, P.Bourque, R.Dupuis, Swebok, aprile, 2004.

[BDEFMV05] M. Brauer , P. Durusau, G. Edwars, D. Faure, T. Magliery, D.Vogelheim. OASIS Standard (1 may 2005). *Open Document Format for Office Applications (OpenDocument) V1.0*. *http://books.evc-cit.info/odbook/* , 2005.

[CDS09] Clever Age, DIaLOGIKa, Sonata Software Ltd. Open XML / ODF Translator Add-ins for Office. *http://odf-converter.sourceforge.net/*, 2009.

[Dil07] A. Di Iorio. *A document segmentation model: Pentaformat. In Pattern-based Segmentation of Digital Documents: Model and Implementation, Tecnical Report. Bologna*, UBLCS, 2007.

[IS06] ISO/IEC 26300. *Information technology - Open Document Format for Office Applications (OpenDocument) v1.0.* http://www.iso.org/iso/catalogue_detail.htm?csnumber=43485, 2006. Ultima visita: maggio, 2010.

[IS07] I. Sommerville. *SOFTWARE ENGINEERING 08 Edition by arrangement with Pearson Education Limited, Unit Kingdom*. Ian Sommerville. Pearson Paravia Bruno Mondadori 978-88-7192-354-3, 2007.

[IS08] ISO/IEC. Lifecycle Software.
Software Engineering Process Technology
http://www.12207.com/12207-news.htm
ISO/IEC 12207, 2008.

[IS10] ISO. ISO/IEC DIS 29500 receives necessary votes for approval as an International Standard. *Office Open XML file formats.*
http://www.iso.org/iso/pressrelease.htm?refid=Ref1123, 2010. Ultima visita: maggio, 2010.

[M07] P. Marinelli. Office Open XML Open Document Coverter.
http://vitali.web.cs.unibo.it/Progetti/OpenXMLODFConverter , 2007.

[MR03] Microsoft-Redmond, november, 2003. *Overview ofWordprocessingML*, Microsoft Corporation, - One Microsoft Way -Redmond, WA 98052-6399, USA.
http://rep.oio.dk/Microsoft.com/officeschemas/wordprocessingml_article.htm , 2003.

[MV07a] P. Marinelli and F. Vitali. *WordprocessingML to Open Document Conversion.* Bologna, 2007.

[MV07b] P. Marinelli and F. Vitali. *Design Issues on Text DocumentsConversion.* Bologna, 2007.

[N06] T. Ngo, Editor, Ecma TC45. *Office Open XML Overview* Ecma TC45, Tom Ngo, Editor, 2006. *http://www.ecmainternational.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf* , 2006.

[O'Re05] O'Reilly & Associates. *OASIS OpenDocument Essentials.* *http://books.evc-cit.info/index.html*, 2005.

[R06] F. Rice, Microsoft Corporation. *Introducing the Office (2007) Open XML File Format*. *http://msdn.microsoft.com/enus/library/aa338205(office.12).aspx* , msdn, may, 2006.

[R07a] B. REX, Wiley&Sons Ltd.. Software Testing WhiteBoxTestingStrategy. *PRAGMATIC SOFTWARE TESTING, BECOMING AN EFFECTIVE AND EFFICIENT TEST PROFESSIONAL*. 2007.

[VM09] F. Vitali and P. Marinelli. *Interoperability across different ISO/IEC file formats: the pentaformat approach*. ISO meeting, Parigi, Dec, 2009.

[We08] Weir Rob. *Only a little bit more than you need to know in: The World of ODF*. IBM, Tshwane, June, 2008.

APPENDICE A

TEST SUITE

CHAR LANGUAGE

\wFont\Arial\Arial.docx

\wFont\Calibri\Calibri.docx

\wFont\Times\Times.docx

\wLang\ChinaChar\ChinaChar.docx

\wLang\MultiLingua\multiling.docx

DRAWING

\Style\wShape\AbsolutePositioning\AbsolutePositioning.docx

\Style\wShape\Effects_Gradient\Effects_Gradient.docx

\Style\wShape\FillEffectsPattern\FillEffectsPattern.docx

\Style\wShape\HorizontalAlignmentPage\HorizontalAlignmentPage.docx

\Style\wShape\HorizRelAbsPos\HorizRelAbsPos.docx

\Style\wShape\OvalEffect\OvalEffect.docx

\Style\wShape\RotateRotation\RotateRotation.docx

\Style\wShape\ShadowEffects\ShadowEffects.docx

\Style\wShape\ShapeRelativePos\ShapeRelativePos.docx

\Style\wShape\ShapesInTable\ShapesInTable.docx

\Style\wShape\ShapeSizeMargin\ShapeSizeMargin.docx

\Style\wShape\ShapesProp\ShapesProp.docx

\Style\wShape\Triangle\Triangle.docx

\Style\wShape\VMLAbsolutePos\VMLAbsolutePos.doc

\Style\wShape\vmlRelativePos\vmlRelativePos.docx

\Style\wShape\VML_shapes\VML_shapes.docx

\Style\wShape\WrapBehindText\WrapBehindText.docx

\Style\wShape\Wrapinfrontoftext\Wrapinfrontoftext.docx

\Style\wShape\WrapInfront_text\WrapInfront_text.docx

\Style\wShape\WrapInLine\WrapInLine.docx

\wPict\wRect\FillEffectsVarients\FillEffectsVarients.docx

\wPict\wShape\Arrowwidth\Arrowwidth.docx

\wPict\wShape\Effects_Arrows\Effects_Arrows.docx

\wPict\wShape\JCMissionStatement\JCMissionStatement.docx

\wPict\wShape\TestfeaturesAll\TestfeaturesAll.docx

\wPict\wShape\Trasparenza\Transerancy.docx

\wPict\wShape\VMLWrapSquare\VMLWrapSquare.docx

HEADER FOOTER

\Style\NumberFormat\NumberFormat1\NumberFormat1.docx

\Style\NumberFormat\NumberFormatA\NumberFormat_A.docx

\Style\NumberFormat\NumberFormat_1\NumberFormat_1.docx

\Style\NumberFormat\NumberFormat_a\NumberFormatA.docx

\Style\TestTOC2007\TestTOC2007.docx

\wFooter\Foot\Foot.docx

\wFooter\FramesFooter\FramesFooter.docx

\wHeader\FramesHeader\FramesHeader.docx

\wHyperlink\HeadFoot_Chapter\HeadFoot_Chapter.docx

\wHyperlink\TOCTableEntry\TOCTableEntry.docx

LIST

\Style\Type\Liste123\Liste123.docx

\Style\Type\ListeABC\ListeABC.docx

\Style\Type\Liste_abc\Liste_abc.docx

\Style\Type\Liste_i_ii\Liste_i_ii.docx

\wpPr\ListToList\ListToList.docx

\wpPr\plusList\plusList.docx

\wpPr\TestListIndent\TestListIndent.docx

PAGE LAYOUT

\Style\FaxSample\FaxSample.docx

\Style\LayoutMisto\LayoutMisto.docx

\Style\LetterSample\LetterSample.docx

\Style\NumberFormat\NumberFormat.docx

\Style\PageLayoutMist\ITSignOffReportEN\ITSignOffReportEN.docx

\Style\PageLayoutMist\LayRichKidd\LayRichKidd.docx

\Style\PageLayoutMist\Orientation\Orientation.docx

\Style\PageSetup_Column\PageSetup_Column.docx

\Style\TrustRelease\TrustRelease.docx

\WsectPr\Margin\Custom_Margin\Custom_Margin.docx

\WsectPr\Margin\HorizontalAlignColumn\HorizontalAlignColumn.docx

\WsectPr\Margin\HorizontalAlignRelative\HorizontalAlignRelative.docx

\WsectPr\Margin\HorizontalVerticalRelative\HorizontalVerticalRelative.docx

\WsectPr\Margin\HorzontalVerticalMargin\HorzontalVerticalMargin.docx

\WsectPr\PageNumbering\PageNumbering.docx

\WsectPr\PageNumberOffset\PageNumberOffset.docx

\WsectPr\Performance571\performance571.docx

\WsectPr\Sezioni\Break\Break.docx

\WsectPr\Sezioni\Break_ContinuousPage\Break_ContinuousPage.docx

\WsectPr\Sezioni\Break_EvenPage\Break_EvenPage.docx

\WsectPr\Sezioni\Break_NextPage\Break_NextPage.docx

\WsectPr\Sezioni\Break_OddPage\Break_OddPage.docx

\WsectPr\test1496pages\test1496pages.docx

\WsectPr\test_14000_1745pages\test_14000_1745pages.docx

\WsectPr\test_1994pages\test_1994pages.docx

\WsectPr\test_20000_2492pages\test_20000_2492pages.docx

\WsectPr\test_2243pages\test_2243pages.docx

PICTURES

\Style\CodeSigning\CodeSigning.docx

\Style\FillEffectsPicture\FillEffectsPicture.docx

\Style\Textboxes\Textboxes.docx

\wDrawing\EmbeddedOLEObjects\EmbeddedOLEObjects.docx

\wDrawing\FrameColumn\FrameColumn.docx

\wDrawing\HandboekOOo2Writer\HandboekOOo2Writer.docx

\wDrawing\Illustrations_Picture\Illustrations_Picture.docx

\wDrawing\imagelost2\imagelost2.docx

\wDrawing\imagelost3\imagelost3.docx

\wDrawing\imagelost4\imagelost4.docx

\wDrawing\OOo2aken-BUG\OOo2aken_ET.docx

\wDrawing\WordArt\WArt\WArt.docx

\wDrawing\WordArt\WordArt2003\WordArt2003.docx

TABLE

\Style\OOo_Spec_EN\OOo_Spec_EN.docx

\Style\StyleTable\StyleTable.docx

\Style\TablOfContents\TablOfContents.docx

\Style\TrackChanges\TrackChanges.docx

\wTbl\ColumnRow\ColumnRow.docx

TEXT

\Style\bibliography\SimBibliography\SimBibliography.docx

\Style\CharStyles\CharStyles.docx

\Style\Document\ReleaseNotesODF\ReleaseNotesODF.docx

\Style\Document\RoadmapWord\RoadmapWord.docx

\Style\ReportSample\ReportSample.docx

\Style\Title\Greentitle\Greentitle.docx

\Style\Title\Heading\Heading.docx

\Style\wP\Curriculum\Curriculum.docx

\Style\wP\TestHeadingNum\TestHeadingNum.docx

\Style\wP\TestNumberingalignment\TestNumberingalignment.docx

\wP\wRun\Controls\Controls.docx

\wP\wRun\MemoSample\MemoSample.docx

\wP\wRun\TextDirection\TextDirection.docx

\wP\wRun\Titoli\TitleProperties\TitleProperties.docx

I file che compongo la suite sono stati reperiti attraverso:

- http://odf-converter.sourceforge.net/documentation.html
  nella sezione ODF Add-in for word in TestSuite file for docx.

- http://sourceforge.net/projects/odf-converter/files/
  nella sezione Documentation Word Translator.

- Parte prodotti da me e parte attraverso ricerche generiche dalla rete su documenti di lavoro word.