Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di laurea in informatica

Pattern design per documenti strutturati: il problema della conversione da e per formati documentali tradizionali

Relatore:
Chiar.mo Prof.
Fabio Vitali

Correlatore:
Chiar.mo Prof.
Angelo Di Iorio

Presentata da:
Andrea Baroni

Sessione II
Anno accademico 2016/2017

Indice

Elenco delle figure

Elenco delle tabelle

Esempi

Capitolo 1

Introduzione

In questa dissertazione presento DOCX2ADF e ADF2DOCX, due software per la conversione di documenti. Il primo si occupa di trasformare documenti redatti attraverso Microsoft Word, dal formato Office Open XML al formato ADF, mentre il secondo converte documenti dal formato ADF al formato Office Open XML. Il formato ADF (ALSTOM Data Format), che spiego in dettaglio nel Capitolo 2, è un sottoinsieme di HTML 5 che impiega solamente 25 elementi; è stato ideato in quanto un formato basato su HTML permette di svolgere operazioni di manipolazione automatiche sui documenti, come per esempio analisi statistiche o estrazioni di particolari dati. ADF è basato su una particolare teoria, denominata teoria dei pattern per i documenti; l'idea è quella di applicare il concetto di pattern alla stesura di documenti. Come avrò modo di approfondire nel Capitolo 2, attraverso una ricerca è stato possibile individuare delle regolarità nei documenti tenendo in considerazione due importanti fattori:

- Cosa può contenere un elemento (*il content model*)
- Dove può essere posto un elemento (*il context*)

Questi due principi, *il content model* e *il context,* hanno portato alla definizione di otto pattern o modelli di struttura che si trovano nei testi.
Il motivo che ha portato allo sviluppo dei convertitori, è per far fronte alle esigenze dell'azienda ALSTOM che spiego nel seguito.
A seguito dell'adozione del formato ADF da parte di ALSTOM e dell'invenzione da parte del DASPLab di un editor in grado di redigere documenti direttamente nel formato ADF, è sorto il problema di convertire i documenti tecnici dell'azienda ALSTOM che attualmente sono tutti nel formato Office Open XML nel formato ADF. Il problema è stato affrontato implementando un convertitore da Office Open XML ad ADF che ho chiamato DOCX2ADF. Una direttiva interna di ALSTOM impone di avere tutti i documenti nel formato Office Open XML perciò è

stato deciso di aggiungere una “feature” all’editor , implementata attraverso il convertitore ADF2DOCX, in grado di salvare il documento ADF nel formato Office Open XML e di conseguenza visualizzarlo correttamente attraverso MS Word. Esamino il formato Office Open XML nel Capitolo 2 mentre nel Capitolo 3 discuto in mode generale dei convertitori DOCX2ADF e ADF2DOCX.
Per realizzare la conversione da ADF a Office Open XML e viceversa, è necessario studiare come i frammenti XML nel formato ADF, che descrivono, per esempio, paragrafi, tabelle e liste vengono resi nel formato Office Open XML; nel Capitolo 4 approfondisco la questione, esaminando le corrispondenze tra ADF e Office Open XML, mentre nel Capitolo 5 analizzo le limitazioni dei convertitori attraverso un elenco di elementi che non vengono gestiti.

Capitolo 2

Formati documentali e convertitori

Nel seguito descrivo le principali caratteristiche del formato Office Open XML o DOCX.
Spiego inoltre il particolare sottoinsieme di HTML 5 progettato per l'azienda ALSTOM denominato ADF e basato su RASH, su cui vado ad effettuare la conversione da documenti in formato Office Open XML.
Concludo il capitolo con i principali convertitori presenti sul mercato che effettuano la conversione da Office Open XML a HTML 5 e viceversa.

2.1 Il formato di MS Word Office Open XML

Office Open XML [1], chiamato anche OOXML, è il formato basato su XML, dei documenti, generati dai programmi, della suite Microsoft Office.
La specifica è stata sviluppata da Microsoft e adottata da ECMA International come ECMA-376 nel 2006. L'ultima versione è stata rilasciata nel dicembre 2016.
La specifica è stata anche adottata da ISO e IEC come ISO/IEC 29500.
È importante tenere presente che OOXML non è l'equivalente di Open Document Format (ODF). Open Document Format (ODF) e il formato dei documenti prodotti dalla suite Libre Office, è di fatto il principale concorrente del formato Office Open XML. Office Open XML ha sostituito il vecchio formato binario .doc che però continua ad essere supportato. Microsoft annunciò lo sviluppo del nuovo formato nel 2000 e nel 2005 rese noto che avrebbe partecipato al processo di standardizzazione attraverso Ecma International.

2.1.1 Specifiche incluse in ECMA-376

ECMA 376 include tre differenti specifiche, per i tre formati dei documenti prodotti da programmi per l'ufficio.

WordProcessingML, per i documenti prodotti da software di videoscrittura, SpreadsheetML per i fogli di calcolo e PresentazionML per le presentazioni. Inoltre include alcuni linguaggi di markup, il più importante dei quali, DrawingML, viene impiegato per la creazione di figure geometriche o l'inserimento di immagini, oltre che per il posizionamento di oggetti.
La specifica comprende lo schema XML. Ogni documento per essere considerato valido deve essere conforme allo schema e deve essere codificato in UTF-8 o UTF-16.

2.1.2 OOXML: un insieme di file compressi

La seconda parte di ECMA-376 è dedicata alla tecnologia Open Packaging Conventions (OPC). OPC si basa sul comune formato ZIP utilizzato per unire più file in un pacchetto comune. I file Office Open XML infatti non sono altro che un archivio compresso in formato ZIP che contiene vari file XML divisi in diverse parti che alla fine sono compressi in un singolo pacchetto. Il fatto di utilizzare più file XML consente di ridurre il rischio di corruzione di dati e permette un più veloce accesso ad essi.
I vari file XML, che compressi vanno a costituire un documento Office Open XML, possono contenere qualsiasi genere di dati, e per tenere traccia di questi dati senza basarsi sulla loro estensione si fa ricorso ad un file chiamato: `[Content_Types].xml`.
I riferimenti che i vari file possono avere, ad esempio verso immagini o siti web esterni piuttosto che verso parti interne dello stesso documento, sono specificati in un singolo file esterno, che sarà unico per ogni parte che compone il documento Office Open XML finale, in modo che la modifica di un riferimento porti solamente alla rettifica di un file.

2.1.3 Descrizione dei file XML che compongono un documento Office Open XML

Nel seguito descriverò i file XML che compongo un documento Office Open XML, soffermandomi su quelli effettivamente utilizzati dai programmi di conversione che esibisco in questa tesi.

Una lista dei file XML che compongono un esempio di un documento Office Open XML decompresso è la seguente:

1. /[Content_Types].xml
2. /_rels/.rels
3. /docProps/app.xml
4. /docProps/core.xml
5. /word/document.xml
6. /word/footer1.xml
7. /word/header1.xml
8. /word/_rels/document.xml.rels
9. /word/_rels/header1.xml.rels
10. /word/_rels/footer1.xml.rels
11. /word/endnotes.xml
12. /word/footnotes.xml
13. /word/fontTable.xml
14. /word/numbering.xml
15. /word/settings.xml
16. /word/styles.xml
17. /word/theme/theme1.xml
18. /word/media/image1.png

`[Content_Types].xml` è un file che obbligatoriamente deve esistere in ogni documento . Al suo interno si trova per ciascun file il suo tipo di dato, ad esempio per il file `/word/document.xml` si trova la stringa:

```
<Override PartName="/word/document.xml"
ContentType="application/vnd.openxmlformats-officedocument.word-
processingml.document.main+xml"/>.
```

Esempio 1: Content Type di document.xml

Inoltre vengono qui elencate anche tutte le possibili estensioni delle immagini presenti nella cartella media, es: `<Default Extension="png" ContentType="image/png"/>`. Se nella cartella media, che contiene tutte le possibili immagini del documento Office Open XML finale, è presente un'immagine con un'estensione che non è presente nel file `[Content_Types].xml`, il documento Office Open XML finale non risulterà valido e di conseguenza non potrà essere visualizzato nell'editor.
Il contenuto principale di ogni documento Office Open XML è contenuto dentro al file `document.xml`, il quale deve essere obbligatoriamente presente.

Dentro al file `document.xml` è presente la struttura logica del documento Office Open XML, che presenta in questo caso analogie con la struttura di un documento HTML, come si evince dalla Tabella 1.

Struttura principale di un documento Office Open XML	**Struttura principale documento HTML**

`<w:document>`
`  <w:body>`
`  <w:p/>`
`  </w:body>`
`</w:document>`
	`<html>`
	`  <head>`
	`  ...`
	`  </head>`
	`  <body>`
	`  ...`
	`  </body>`
	`</html>`

Tabella 1: Struttura documenti Office Open XML e documenti HTML

Dentro al file `document.xml` è possibile specificare elementi come paragrafi, tabelle, liste, riferimenti a parti interne o esterne del documento.
I paragrafi insieme alle tabelle sono gli elementi al più alto livello all'interno dell'elemento `<w:body>`. Per specificare un paragrafo si usa l'elemento `<w:p>`. Ogni paragrafo può avere delle proprietà, ad esempio può essere un titolo o avere una particolare posizione all'interno del documento; tutte queste caratteristiche sono specificate all'interno dell'elemento `<w:pPr>` figlio diretto dell'elemento `<w:p>`.
Ogni paragrafo che contiene testo, ha poi al suo interno un particolare elemento denominato `<w:r>` al cui interno verrà inserito del testo attraverso l'elemento `<w:t>`. L'elemento `<w:r>` è utilizzato, per definire una regione, all'interno di un paragrafo, che non definisce un blocco ma specifica alcune proprietà che saranno locali all'elemento. È utilizzato prevalentemente per attribuire delle proprietà non all'intero paragrafo ma solo ad alcune parole o gruppi di parole.

Un esempio di paragrafo con testo al centro è il seguente:

```
<w:p>
  <w:pPr>
    <w:jc w:val="center"/>
  </w:pPr>
  <w:r>
    <w:t>Semplice paragrafo con testo al centro</w:t>
  </w:r>
</w:p>
```

Esempio 2: Esempio di un paragrafo nel formato Office Open XML

Per definire proprietà diverse all'interno dello stesso paragrafo si usa l'elemento `<w:rPr>` come figlio diretto dell'elemento `<w:r>`, un confronto con HTML è possibile vederlo nella Tabella 2.

OOXML	**HTML**

`<w:p>`
`  <w:r>`
`  <w:rPr>`
`  <w:b/>`
`  </w:rPr>`
`  <w:t xml:space="preserve">Questo testo è in grassetto </w:t>`
`  </w:r>`
`  <w:r>`
`  <w:t xml:space="preserve">, questo non ha alcun stile,</w:t>`
`  </w:r>`
`  <w:r>`
`  <w:rPr>`
`  <w:sz w:val="28"/>`
`  <w:szCs w:val="28"/>`
`  <w:u w:val="single"/>`
`  </w:rPr>`
`  <w:t xml:space="preserve">`
`  mentre questo è sottolineato ed ha dimensione maggiore`
`  </w:t>`
`  </w:r>`
`</w:p>`
	`<p>`
	`  <strong>`
	`  Questo testo è in grassetto`
	`  </strong>`
	`  questo non ha alcun stile`
	`  <u style="font-size:200%">`
	`  mentre questo è sottolineato ed ha dimensione maggiore`
	`  </u>`
	`</p>`

Tabella 2: Confronto paragrafi OOXML e paragrafi HTML.

Come è possibile osservare nella Tabella 2, per specificare la dimensione dei caratteri in un file Office Open XML viene usato l'attributo `w:val` all'interno dell'elemento `w:sz`; w:val=1 equivale a mezzo punto tipografico, che approssimativamente equivale a 0.018 cm.
Le tabelle in Office Open XML presentano forti analogie con il modo in cui sono espresse attraverso HTML.
In particolare come elemento radice viene utilizzato `<w:tbl>`, che è l'unico elemento che può stare al livello di `<w:p>`. Per le righe e le celle vengono usati rispettivamente gli elementi `<w:tr>` e `<w:tc>`. Un esempio è possibile vederlo nella Tabella 3 in cui vengono omessi per semplicità alcuni elementi. In particolare `<w:tblPr>` è impiegato per specificare le proprietà globali della tabella, come l'uso di un particolare stile, proprietà che possono essere sovrascritte dalle proprietà delle righe o delle celle tramite l'uso degli elementi `<w:trPr>` e `<w:tcPr>` rispettivamente. L'elemento `<w:tblGrid>` contiene l'elemento `<w:gridCol>` che insieme all'attributo `<w:w>` serve a specificare la larghezza di ogni colonna.

OOXML	HTML
<w:tbl> <w:tblPr> ... </w:tblPr> <w:tblGrid> ... </w:tblGrid> <w:tr> <w:trPr> <w:tblHeader/> </w:trPr> <w:tc> <w:p> <w:r> <w:t>TITOLO1</w:t> </w:r> </w:p> </w:tc> <w:tc> <w:p> <w:r> <w:t>TITOLO2</w:t> </w:r> </w:p> </w:tc> </w:tr> <w:tr> <w:tc> <w:p> <w:r> <w:t>Riga1 cella 1</w:t> </w:r> </w:p> </w:tc> <w:tc> <w:p> <w:r> <w:t>Riga1 cella 2</w:t> </w:r> </w:p> </w:tc> </w:tr> </w:tbl>	<table> <tr> <th>TITOLO1</th> <th>TITOLO2</th> </tr> <tr> <td>Riga1 cella 1</td> <td>Riga1 cella 2</td> </tr> </table>

Tabella 3: Confronto tabelle OOXML e tabelle HTML

È interessante notare come in un file OOXML vengano gestiti i riferimenti. Per ciascuna cartella presente all'interno di un file OOXML infatti, è presente una cartella denominata `_rels` al cui interno si trova un file XML che termina con `.rels`, il nome di questo file è quello in cui sono presenti i riferimenti. Per esempio nella cartella word è presente il file `document.xml`, i riferimenti presenti in esso saranno collocati nel file `word/_rels/document.xml.rels`, all'interno di questo file sono presenti riferimenti a parti esterne del documento come al foglio di stile `/word/styles.xml` ma anche a immagini o siti web.
I riferimenti possono essere di due tipi: espliciti o impliciti. Per i riferimenti espliciti la risorsa in questione viene specificata attraverso l'uso di un Id, attributo di un elemento di nome `<Relationship>` il quale conterrà la locazione della risorsa. Per esempio in `document.xml` potremmo avere un' immagine che viene specificata nel seguente modo: `<a:blip r:embed="rId35">` , nel file `document.xml.rels` si avrà:

```
<Relationship Id="rId35"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image25.png"/>
```

Esempio 3: Locazione dell'immagine con id : rId35 specificata dall'attributo Target

che riporta all'immagine: image25.png.
Per i riferimenti impliciti invece non viene fatto uso di un elemento `<Relationship>` ma si suppone che se è presente il riferimento allora esisterà la risorsa. Per esempio se è presente un riferimento a una nota a piè pagina del tipo:

```
<w:footnoteReference r:id="2">
```

Esempio 4: Riferimento a una nota a piè di pagina

allora sicuramente esisterà un file `footnotes.xml` che al suo interno avrà un elemento `<w:footnote w:id=”2”>` che conterrà il testo della nota a piè pagina.
OOXML non ha il concetto di pagina, però alcune informazioni che riguardano la paginazione, sono memorizzate in una particolare sezione alla fine del file `document.xml`, dentro all'elemento `<w:sectPr>` che contiene tali informazioni come i riferimenti ai file che definiscono le intestazioni e i piè di pagina, come si può vedere nel seguente esempio:

```
<w:sectPr>
  <w:headerReference w:type="default" r:id="rId9"/>
  <w:footerReference w:type="default" r:id="rId10"/>
</w:sectPr>
```

Esempio 5: Informazioni riguardo la paginazione

L'attributo `<w:type>` permette di specificare intestazioni e piè di pagina differenti per la prima pagina, pagine pari o dispari.
Le intestazioni sono specificate nel file `/word/header1.xml` mentre i riferimenti presenti in esso nel file: `/word/_rels/header1.xml.rels`, i piè di pagina invece nel file `/word/foooter1.xml` mentre i riferimenti nel file `/word/_rels/footer1.xml.rels`.
Un' importante differenza che sussiste con il formato HTML riguarda il modo in cui sono definite le liste. In HTML la lista è trattata come un elemento gerarchico, ovvero si definisce un elemento radice come `<ol>` per le liste ordinate, e in seguito si vanno a definire i singoli elementi che compongono la lista tramite i tag `<li>` un esempio è visibile nella Tabella 4 .
Le liste in OOXML Tabella 4 non hanno una struttura gerarchica, gli elementi che compongono la lista sono tutti allo stesso livello, specificati per mezzo dell'elemento: `<w:p>`. Ciò che distingue una lista da un normale paragrafo è specificato nel campo dedicato alle proprietà `<w:pPr>`, attraverso l'elemento figlio `<w:numPr>` il quale contiene due elementi `<w:ilvl>` e `<w:numId>`. `<w:ilvl>` è impiegato per specificare il livello di un elemento in una particolare lista attraverso l'attributo `<w:val>` , partendo da `"w:val=0"` per indicare il primo livello e procedendo sequenzialmente. `<w:numId>` attraverso l'attributo `<w:val>` specifica un Id univoco per ogni lista, che è anche memorizzato nel file `word/numbering.xml`, il quale contiene le informazioni specifiche di ogni lista, come, per esempio, il fatto che una lista sia ordinata o meno.

OOXML	HTML
<w:p> <w:pPr> <w:pStyle w:val="Elenco"/> <w:numPr> <w:ilvl w:val="0"/> <w:numId w:val="2"/> </w:numPr> </w:pPr> <w:r> <w:t>Primo elemento</w:t> </w:r> </w:p> <w:p> <w:pPr> <w:pStyle w:val="Elenco"/> <w:numPr> <w:ilvl w:val="0"/> <w:numId w:val="2"/> </w:numPr> </w:pPr> <w:r> <w:t>Secondo elemento</w:t> </w:r> </w:p>	 - Primo elemento - Secondo elemento

Tabella 4: Liste in OOXML e in HTML

2.2 Razionalizzazioni di HTML 5

In questo paragrafo descrivo il particolare sottoinsieme di HTML 5 progettato per l'azienda ALSTOM che nel seguito per semplicità chiamerò ADF (ALSTOM DATA FORMAT). Dal momento che ADF è basato su un particolare dialetto di HTML 5 denominato RASH, prima di mostrare le nuove caratteristiche di ADF discuterò di RASH.

2.2.1 Pattern per la creazione di documenti

I formati RASH e ADF sono stati progettati seguendo una particolare teoria denominata teoria dei pattern per i documenti [2] [3].
Seguendo la teoria dei pattern gli elementi di HTML 5 sono raggruppati in quattro categorie visibili nella Tabella 5, in accordo a due principi: elementi che possono/non possono contenere testo e elementi che possono/non possono contenere altri elementi:

	Elementi che non possono contenere testo	**Elementi che possono contenere testo**
Elementi che non possono contenere altri elementi	Vuoti *Marker*	Solo testo *Flat*
Elementi che possono contenere altri elementi	Contenitori di elementi ma non di testo *Bucket*	Contenitori di elementi e testo *Mixed*

Tabella 5 Principi su cui si basa la teoria dei pattern per i documenti

Dal momento che solo due categorie possono contenere altri elementi il modello descritto nella Tabella 5 genera esattamente otto pattern come si può vedere nella tabella seguente:

	Marker	**Flat**	**Bucket**	**Mixed**
Marker	-	-	-	-
Flat	-	-	-	-
Bucket	Meta	Field	Container	Block
Mixed	Milestone	Atom	Popup	Inline

Tabella 6 L'insieme base dei pattern

Nella seguente tabella fornisco una breve descrizione degli otto modelli di struttura che si trovano nei testi, i quali ci permettono di identificare e rappresentare tutti gli elementi che vengono impiegati nella strutturazione dei testi.

Nome Pattern	**Spiegazione**
Meta	È un elemento vuoto *(Marker)* inserito all'interno di un contenitore *(Bucket)* . La sua posizione non è rilevante e solo la sua esistenza ha importanza. È il perfetto candidato per rappresentare le strutture dei metadati, da qui il suo nome.
Milestone	È un elemento vuoto *(Marker)* all'interno di un contenitore di elementi e testo *(Mixed).* La sua posizione all'interno del documento è il suo più importante contributo.
Field	È un elemento che contiene solo testo, inserito all'interno di un *Bucket* (contenitore di elementi ma non di testo), è usato come contenitore di dati, la sua posizione non è rilevante.
Atom	È un elemento che contiene solo testo all'interno di un *Mixed* (contenitore di testo e elementi). È un elemento molto raro sostituito il più delle volte da elementi *Inline.*
Container	È un elemento che può contenere altri elementi, ma non testo, inserito all'interno di un *Bucket* (contenitore di elementi ma non di testo). Questo pattern è "l'impalcatura" principale dei documenti Rash e ADF, è il più importante degli otto presentati in questa tabella.
Popup	È un elemento che può contenere altri elementi, ma non testo, inserito all'interno di un *Mixed* (contenitore di elementi e testo).
Block	È un elemento che può contenere elementi e testo all'interno di un *Bucket*(contenitore di elementi ma non di testo). Un esempio di questo elemento è l'elemento `<p>` di HTML.
Inline	È un elemento che può contenere elementi e testo all'interno di un *Mixed* (contenitore di elementi e testo). Gli elementi *Inline* non sono strutturali ma caratterizzano frammenti di testo aggiungendo per esempio effetti tipografici o particolari significati semantici.

Tabella 7: Elenco di pattern per la creazione di documenti

Nella prossima tabella inserisco gli elementi di RASH e ADF dividendoli nei vari pattern.

Pattern	**Elementi Rah**	**Elementi ADF**
Inline	`a, code, em, math, q, span, strong, sub, sup, svg`	`a,i,span,b,sub,sup,u`
Block	`figcaption, h1, p, pre, th`	`figcaption,h1,p,th`
Popup	nessuno	nessuno
Container	`blockquote, body, figure, head, html, li, ol, section, table, td, tr, ul`	`body,figure,head,html ,li,ol,ul,tr,td,table`
Atom	nessuno	nessuno
Field	`script, title`	`script`
Milestone	`img`	`img`
Meta	`link, meta`	`link`

Tabella 8: Elementi RASH e ADF divisi per pattern

2.2.2 RASH

Research Articles in Simplified HTML (RASH) [3] è un sottoinsieme di HTML 5 che ne limita l'uso a soli 31 elementi. Permette l'inclusione di statement RDF [4] in diversi formati come JSON-LD [5], RDF/XML [6] , Turtle [7], RDFa [8]. Per migliorare l'accessibilità dei documenti permette l'uso di Digital Publishing WAI-ARIA Module 1.0 [9] e Accessible Rich Internet Applications 1.1 [10].
Una lista degli elementi di HTML5 permessa è la seguente:`<a>`, `<blockquote>`, `<body>`, `<code>`, `<em>`, `<figcaption`, `<figure>`, `<h1>`, `<head>`, `<html>`, `<img>`, `<li>`, `<link>`, `<math>`, `<meta>`, `<ol>`, `<p>`, `<pre>`, `<q>`, `<script>`, `<section>`, `<span>`, `<strong>`, `<sub>`, `<sup>`, `<svg>`, `<table>`, `<td>`, `<th>`, `<title>`, `<tr>`, `<ul>`.
Tutti gli elementi precedentemente elencati hanno una forte semantica strutturale definita esplicitamente nella specifica di HTML5 [11]. Vengono usati dei ruoli specifici attraverso l'uso dell'attributo `<role>` per marcare ulteriormente il ruolo di alcuni elementi.
Per esempio `<section>` indica una sezione ma `<section role=”doc-abstract”>` indica che in realtà quella sezione è il sommario dell'articolo; `<li>` indica un elemento di una lista ma `<li role=”doc-biblioentry”>` viene indicato per specificare un riferimento bibliografico a un certo articolo specificato nel testo. Un esempio di alto livello di un documento RASH è il seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html
 xmlns="http://www.w3.org/1999/xhtml"
 prefix="
 schema: http://schema.org/
 prism: http://prismstandard.org/namespaces/basic/2.0/">
 <head>
  <-- Metadati del documento (autori, affiliazioni, ecc.) -->
 </head>
 <body>
  <-- Contenuto del documento -->
 </body>
</html>
```

Esempio 6: Struttura ad alto livello di un documento in formato RASH

I metadati hanno la seguente forma:

```
<head>
...
<title>RASH: Research Articles in Simplified HTML -- Documenta-
tion - Version 0.5, February
15, 2016</title>

<!-- Dati dell'autore -->
<meta about="#sp" name="dc.creator" content="Silvio Peroni"/>
<meta about="#sp" property="schema:email" content="silvio.pero-
ni@unibo.it"/>
<link about="#sp" property="schema:affiliation" href="#cs-
unibo"/>

<!-- Affiliazioni -->
<meta about="#cs-unibo" property="schema:name" content="Depart-
ment of Computer Science and
Engineering, University of Bologna, Bologna, Italy"/>

<!-- Parole chiave -->
<meta property="prism:keyword"
<meta property="prism:keyword"
<meta property="prism:keyword"
<meta property="prism:keyword"
content="format for scholarly papers"/>
content="HTML"/>
content="markup language"/>
content="Semantic Publishing"/>

<!-- Categorie di classificazione →
<meta name="dcterms.subject" content="Applied computing,Document
management and text
processing,Document preparation,Markup languages"/>
<meta name="dcterms.subject" content="Applied computing,Document
management and text
processing,Document preparation,Annotation"/>
...
</head>
```

Esempio 7: Metadati inseriti in un documento RASH

È interessante notare come avviene la numerazione delle sezioni e delle sottosezioni. RASH non permette di usare gli elementi `<h2>` `<h3>` … `<h6>` per definire i ti-

toli delle sottosezioni ma è lecito usare solo l'elemento `<h1>` come primo elemento di una sezione per definirne il titolo.
Per definire una sottosezione è necessario inserire un elemento `<section>` annidato e RASH si occuperà di numerare correttamente tutte le sezioni, come specificato nel seguente esempio:

```
<body>
 <!-- Sommario -->
 <section role="doc-abstract">
  <h1>Sommario</h1>
  <p>Testo del sommario.</p>
 </section>
 <!-- Sezioni normali -->
 <section id="prima_sezione">
  <h1>Titolo di primo livello</h1>
  <p>Un paragrafo</p>
  <p>Un altro paragrafo</p>
   <section id="seconda_sezione">
    <h1>Titolo di secondo livello</h1>
    <pre><code>Blocco con codice</code></pre>
    <ul>
     <li><p>Elemento di una lista non ordinata</p></li>
     <li><p>Un altro elemento di una lista non ordinata</p></li>
    </ul>
   </section>
 </section>
</body>
```

Esempio 8: Esempio di definizione di sezioni e sottosezioni in un documento RASH

2.2.3 ADF

ADF [2] è un formato che si basa su RASH e discosta da esso solo per alcuni particolari.
Una prima differenza riguarda la lista degli elementi ammessi che viene ristretta a: `<section id="..."><h1></h1>...</section>` per le sezioni, `<h1>`, `<span>`, `<p>`, `<figure>`, `<table>`, `<tr>`, `<th>`, `<td>`, `<figcaption>`, `<img>`, `<ul>`, `<ol>`, `<li>`, `<a href="">`, `<b>`, `<i>`, `<u>`, `<sup>`, `<sub>`,`<body>`,`<script>`,`<link>`,`<head>`,`<html>`.
In particolare, come accade con RASH, l'elemento `<h1>` è ammesso solo come primo elemento di `<section>`, `<img>` compreso dentro `<p>` e gli elementi che com-

pongono una lista non possono contenere testo libero ma devono sempre avere al loro interno l'elemento `<p>`.
Dal momento che ADF è stato progettato per documenti aziendali la struttura generale del documento è diversa. In particolare viene aggiunta una nuova sezione all'interno dell'elemento `<body>` chiamata "front matter" o pre-testo.
La scelta di introdurre una nuova sezione è dettata dal fatto che i metadati che possono essere specificati in un documento RASH non sono sufficienti a poter esprimere tutte le caratteristiche che può possedere un documento industriale.
Nel "front matter" avviene la visualizzazione dei metadati che sono definiti nell'elemento `<head>` all' interno dell'elemento `<script type="application/ld+json" id="doc-metadata">` tramite il formato JSON-LD, ciò permette di non essere vincolati ad un limitato numero di metadati.

La struttura generale, alla luce delle nuove considerazioni è la seguente:

```
<?xml     version="1.0"  encoding="UTF-8"?>
<!DOCTYPE html>
<html     xmlns="http://www.w3.org/1999/xhtml"
     vocab="http://www.alstom.com/sse/"
prefix="doc:http://www.alstom.com/sse/
     tpl:http://www.alstom.com/sse/templates/">
 <head>
  <link   ...  />
  <script ...> </script>
  <script type="application/ld+json"    id="doc-metadata">
   <--    Metadati del documento   -->
  </script>
 </head>
 <body>
  <section id="frontmatter" role="doc-frontmatter" class="doc-ro">
   <!--   Contenuto del front matter    -->
  </section>
  <section id="content"  role="doc-content">
   <--    Contenuto del documento  -->
  </section>
  <section id="endnotes" role="doc-endnotes">
   <!--   Footnotes -->
  </section>
 </body>
</html>
```

Esempio 9: Struttura ad alto livello di un documento in formato ADF

La visualizzazione grafica dei metadati avverrà per mezzo di una tabella.
Oltre alla visualizzazione dei metadati nella sezione dedicata al front-matter vengono specificate alcune informazioni che riguardano il documento come l'indice delle sezioni, figure e tabelle.
Tutti gli indici nel documento ADF vengono generati dinamicamente tramite l'uso di una particolare libreria.
Una caratteristica introdotta in ADF è la possibilità di specificare intestazioni e piè di pagina che non è possibile specificare in RASH.
Per specificare questi nuovi elementi è necessario introdurre una nuova sezione dopo il "front matter" con la seguente sintassi:

```
<section id = " layout-areas " >
 <section id=" header ">

  <!-- Contenuto intestazione -->

 </section>
 <section id=" footer ">

  <!-- Contenuto piè di pagina -->

 </section>
</section>
```

Esempio 10: Modalità per inserire intestazioni e piè di pagina in un documento in formato ADF

2.2.3.1 Sezioni e sottosezioni nel formato ADF

Le sezioni e le sottosezioni nel formato ADF, sono specificate allo stesso modo di RASH. Nella Figura 1 è visibile un esempio di un documento in formato ADF, visualizzato mediante un browser, in cui vengono visualizzate sezioni e sottosezioni.

1. INTRODUZIONE

1.1. Scopo del documento

Scopo del presente documento è descrivere gli interventi da eseguire all'apparato ACEI di Carimate ed agli impianti di alimentazione IS, in relazione alle modifiche di PRG oggetto previste in appalto.

1.2. Campo di applicazione

Il presente documento è applicabile all'appalto per il "Potenziamento tecnologico della tratta Monza – Chiasso mediante la realizzazione di un ACCM e delle opere complementari e accessorie nonché realizzazione dell'ERTMS livello 2 sulla linea Milano Centrale/Milano Smistamento – Monza – Chiasso".

1.3. Descrizione del documento

Il documento contiene una descrizione degli interventi da eseguire in cabina e agli enti di piazzale ed agli impianti di alimentazione IS propedeutici e conseguenti alle modifiche di PRG previste in appalto.

Figura 1: Documento ADF visualizzato mediante un browser

Il codice HTML inserito all'interno della sezione deputata al contenuto[1] che descrive la Figura 1 è il seguente:

```
<section>
   <h1><span   role="pdx-hcounter"></span>INTRODUZIONE</h1>
   <section>
      <h1><span    role="pdx-hcounter"></span>Scopo del
documento</h1>
      <p> Scopo del presente documento è [...]</p>
   </section>
   <section>
      <h1><span    role="pdx-hcounter"></span>Campo di
applicazione</h1>
      <p>Il presente documento è [...]</p>
   </section>
   <section>
      <h1><span    role="pdx-hcounter"></span>Descrizione del
documento</h1>
      <p>Il documento contiene una [...]</p>
   </section>
</section>
```

Esempio 11 Sezioni e sottosezioni nel formato ADF

2.3 Convertitori da Office Open XML a HTML5

Estrarre informazioni da un file Office Open XML è molto oneroso e complicato, in quanto non è possibile inserire metadati che descrivano il significato delle parole che compongono un documento nel formato Office Open XML. Ad esempio non è possibile indicare che una particolare stringa all'interno di un documento identifica l'autore di un libro o articolo scientifico.
Questa limitazione impedisce a un'applicazione esterna di estrarre informazioni in modo automatico.
Al contrario HTML permette di inserire all'interno dei propri documenti attributi che attribuiscono un significato preciso agli elementi. Il processo di attribuzione di una semantica agli elementi di un documento HTML è reso possibile attraverso l'uso di un modello per la rappresentazione dei dati chiamato RDF [4].
RDF è basato su triple *soggetto-predicato-oggetto* dette statement.

1 La struttura che descrive le varie sezioni all'interno di un documento ADF è visibile nell'Esempio 9 a pagina 20

Un possibile uso di RDF nel linguaggio HTML è realizzabile attraverso l'uso di RDFa [8](Resource Description Framework in attributes).La versatilità di HTML ha però un limite, infatti richiede un certo lasso di tempo e una discreta pratica per poter essere padroneggiato adeguatamente.
Per i motivi sopra elencati si sono sviluppati software che effettuano la conversione da documenti scritti con un editor testuale come Microsoft Word in documenti HTML; nel seguito ne elenco alcuni analizzandone le limitazioni, che hanno portato allo sviluppo dei programmi che sono oggetto di questa dissertazione.

2.3.1 Mammoth e PyDocX

Mammoth[2] è un software open source, che offre la possibilità di convertire documenti Office Open XML in documenti HTML con alcune limitazioni.
I riferimenti a parti interne del documento Office Open XML come sezioni, figure e parti del testo non vengono gestiti, inoltre la dimensione originaria delle immagini non viene mantenuta.
Non viene applicato nessun stile al documento.
PyDocX[3] offre tutte le funzionalità di Mammoth, in più applica uno stile di default al documento HTML di destinazione e mantiene le dimensioni originali delle immagini presenti nel file Office Open XML.
Sia Mammoth che PyDocX non eseguono correttamente la conversione verso il formato ADF, in quanto il documento Office Open XML visibile nella Figura 8 a pagina 34 è tradotto in modo erroneo, nel modo seguente:

```
<h1>INTRODUZIONE</h1>
   <h2>Scopo del documento</h2>
   <p> Scopo del presente documento è [...]</p>
   <h2>Campo di applicazione</h2>
   <p>Il presente documento è [...]</p>
   <h2>Descrizione del documento</h2>
   <p>Il documento contiene una [...]</p>
```

Esempio 12 Frammento HTML non conforme al formato ADF

La traduzione corretta effettuata dal convertitore DOCXADF è disponibile nell'Esempio 11 a pagina 22.

2 https://github.com/mwilliamson/python-mammoth
3 https://github.com/CenterForOpenScience/pydocx

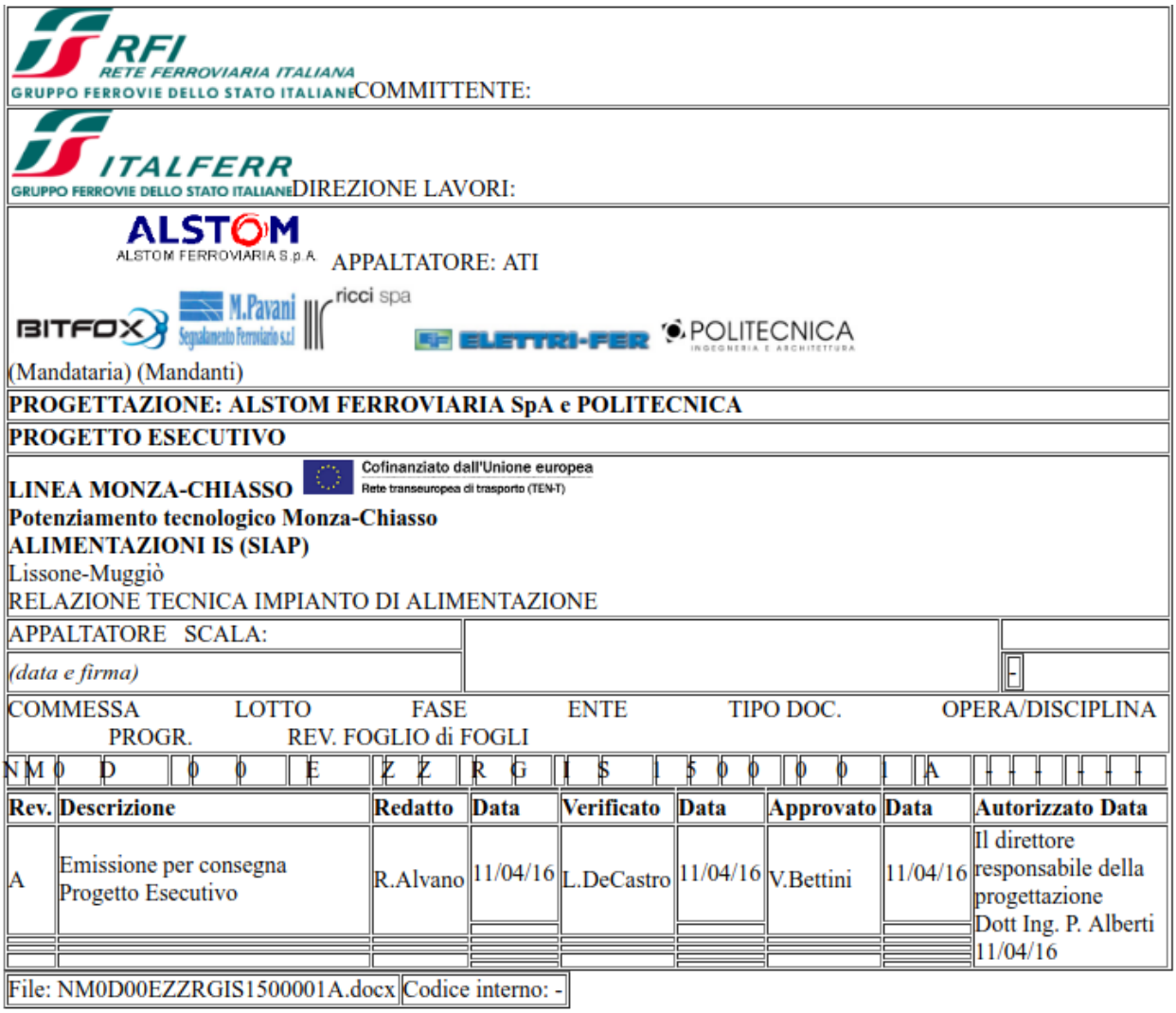
Nelle due figure che seguono (Figura 2 e Figura 3) , mostro come una tipica copertina dei documenti dell'azienda ALSTOM nel formato Office Open XML (visibile nella Figura 4 a pagina 32), viene tradotta da Mammoth e PyDocX rispettivamente, nel formato HTML. Si può notare come Mammoth non mantenga la dimensione originale delle immagini e non applichi bordi alle tabelle. PyDocX al contrario preserva la dimensione delle immagini e applica bordi alle tabelle. Entrambi non gestiscono il posizionamento delle immagini.


RFI
RETE FERROVIARIA ITALIANA
GRUPPO FERROVIE DELLO STATO ITALIANE COMMITTENTE:
ITALFERR
GRUPPO FERROVIE DELLO STATO ITALIANE DIREZIONE LAVORI:
APPALTATORE: ATI
ALSTOM
ALSTOM FERROVIARIA S.p.A.
BITFOX
M.Pavani
Segnalamento Ferroviario s.r.l
ricci spa
ELETTRI-FER
POLITECNICA
INGEGNERIA E ARCHITETTURA
(Mandataria) (Mandanti)
PROGETTAZIONE: ALSTOM FERROVIARIA SpA e POLITECNICA
PROGETTO ESECUTIVO
Cofinanziato dall'Unione europea
Rete transeuropea di trasporto (TEN-T)
LINEA MONZA-CHIASSO
Potenziamento tecnologico Monza-Chiasso
ALIMENTAZIONI IS (SIAP)
Lissone-MuggiÃ²
RELAZIONE TECNICA IMPIANTO DI ALIMENTAZIONE
APPALTATORE
(data e firma)
SCALA:
-
COMMESSA LOTTO FASE ENTE TIPO DOC. OPERA/DISCIPLINA PROGR. REV. FOGLIO di FOGLI
N M 0 D 0 0 E Z Z R G I S 1 5 0 0 0 0 1 A - - - - - -

Rev.	Descrizione	Redatto	Data	Verificato	Data	Approvato	Data	Autorizzato Data
								Il direttore responsabile della progettazione
A	Emissione per consegna Progetto Esecutivo	R.Alvano	11/04/16	L.DeCastro	11/04/16	V.Bettini	11/04/16	Dott Ing. P. Alberti 11/04/16

File: NM0D00EZZRGIS1500001A.docx Codice interno: -

Figura 2: Copertina dei documenti dell'azienda ALSTOM nel formato HTML, tradotta da MAMMOTH



RFI RETE FERROVIARIA ITALIANA
GRUPPO FERROVIE DELLO STATO ITALIANE COMMITTENTE:

ITALFERR
GRUPPO FERROVIE DELLO STATO ITALIANE DIREZIONE LAVORI:

ALSTOM
ALSTOM FERROVIARIA S.p.A. APPALTATORE: ATI

BITFOX M.Pavani Segnalamento Ferroviario s.r.l ricci spa ELETTRI-FER POLITECNICA
INGEGNERIA E ARCHITETTURA
(Mandataria) (Mandanti)

PROGETTAZIONE: ALSTOM FERROVIARIA SpA e POLITECNICA

PROGETTO ESECUTIVO

LINEA MONZA-CHIASSO Cofinanziato dall'Unione europea
Rete transeuropea di trasporto (TEN-T)
Potenziamento tecnologico Monza-Chiasso
ALIMENTAZIONI IS (SIAP)
Lissone-Muggiò
RELAZIONE TECNICA IMPIANTO DI ALIMENTAZIONE

APPALTATORE SCALA:		
(data e firma)		

COMMESSA	LOTTO	FASE	ENTE	TIPO DOC.	OPERA/DISCIPLINA	PROGR.	REV.	FOGLIO	di	FOGLI
NM0	D	0 0	E	Z Z	R G	I S	1 5 0 0	0 0 1	A	

Rev.	Descrizione	Redatto	Data	Verificato	Data	Approvato	Data	Autorizzato Data
A	Emissione per consegna Progetto Esecutivo	R.Alvano	11/04/16	L.DeCastro	11/04/16	V.Bettini	11/04/16	Il direttore responsabile della progettazione Dott Ing. P. Alberti 11/04/16

File: NM0D00EZZRGIS1500001A.docx Codice interno: -

Figura 3 Copertina dei documenti dell'azienda ALSTOM, nel formato HTML tradotta da PyDocX

2.4 Convertitori da HTML5 a Office Open XML

Molte aziende richiedono che i propri documenti siano scritti con editor testuali ad esempio Microsoft Word. Modificare un file HTML infatti non è un' operazione semplice ed intuitiva.

L' Azienda per cui si è deciso di sviluppare i convertitori che presento in questa dissertazione ha una propria direttiva interna, la quale richiede di avere i propri documenti in formato Office Open XML.

Sul mercato sono presenti numerosi software che effettuano la conversione da HTML5 a Office Open XML, però anche in questo caso data la grossa discrepan-

za che sussiste tra i due formati è molto difficile trovare un software che sia adatto allo scopo.

2.4.1 PHPDOCX e XmlDocx

PhpDocx[4] è un software proprietario e a pagamento che permette di creare documenti Office Open XML richiamando funzioni da una libreria o effettuando la conversione da un documento HTML preesistente.
È fornito, come si evince dal nome, attraverso una libreria PHP.
Permette di creare quasi tutti gli elementi presenti in un file Office Open XML, tranne ovviamente quelli che non hanno una naturale corrispondenza. Le intestazioni e i piè di pagina che possono essere presenti in un documento Word, a meno di usare convenzioni , non possono essere creati.
Il sottoinsieme di HTML5 progettato per l'azienda ALSTOM (spiegato in dettaglio nel paragrafo 2.2) offre una modalità per specificare le intestazioni e i piè di pagina, inoltre dato il particolare modo in cui viene effettuata la numerazione delle sezioni, questo software e tutti quelli presenti sul mercato non sarebbero in grado di effettuare la conversione in modo corretto.
XmlDocx[5] è un software proprietario e a pagamento, che offre le stesse funzionalità di PhpDocx. È implementato in quasi tutti i linguaggi: Java, Python, Node.js, Ruby, C, C++, C#.

2.4.2 PHPWord

PHPWord[6] è un software open source che permette di creare dinamicamente documenti Office Open XML.
Fa parte di un più ampio gruppo di librerie denominato PhpOffice, impiegato per la creazione e la modifica di documenti realizzabili attraverso software compresi nella raccolta Office, di cui fa parte Microsoft Word. È una libreria scritta interamente in PHP. Per poter essere efficacemente usata necessita della conoscenza del formato Office Open Xml. PHPWord non riesce a convertire correttamente la struttura delle sezioni, di un documento nel formato ADF. Nell'Esempio 13 viene proposto un programma che attraverso l'uso della libreria PHPWord crea un do-

4 https://www.phpdocx.com/
5 http://www.xmldocx.com/
6 https://github.com/PHPOffice/PHPWord

cumento nel formato Office Open XML. IL risultato di tale conversione è erroneo in quanto gli elementi `<h1>` vengono tradotti tutti con titoli di livello 1 invece che usare un livello inferiore a seconda del livello di profondità della sezione in cui il titolo è inserito.

```
<?php
include_once 'Sample_Header.php';


echo date('H:i:s') , ' Create new PhpWord object' , EOL;
$phpWord = new \PhpOffice\PhpWord\PhpWord();

$section = $phpWord->addSection();
$html = '<section   >
   <h1>INTRODUZIONE</h1>
   <section>
      <h1>Scopo del documento</h1>
      <p> Scopo del presente documento è [..]</p>
   </section>
   <section>
      <h1>Campo di applicazione</h1>
      <p>Il presente documento è applicabile [..] </p>
   </section>
   <section>
      <h1>Descrizione del documento</h1>
      <p>Il documento contiene una descrizione [..].</p>
   </section>
</section>';


\PhpOffice\PhpWord\Shared\Html::addHtml($section, $html);

// Save file
echo write($phpWord, basename(__FILE__, '.php'), $writers);
if (!CLI) {
    include_once 'Sample_Footer.php';
}
```

Esempio 13: Programma scritto mediante PHP, che fa uso della libreria PHPWORD

Capitolo 3

DOCX2ADF e ADF2DOCX

In questo capitolo mostro DOCX2ADF e ADF2DOCX, i due convertitori che ho realizzato nel mio lavoro di tesi, per la conversione di documenti dal formato Office Open XML ad ADF e da ADF a Office Open XML rispettivamente.
A seguito dello sviluppo da parte del DASPLab di un' applicazione web in grado di produrre documenti ADF, è sorta la necessità di implementare un convertitore per eseguire la conversione dei documenti dell'azienda ALSTOM, che attualmente sono nel formato Office Open XML in documenti ADF. D'altro canto l'azienda ha una propria direttiva interna che impone di avere tutti i documenti nel formato Office Open XML perciò si è reso necessario lo sviluppo di un ulteriore convertitore per la conversione di documenti dal formato ADF a Office Open XML. Tutti e due i convertitori che presento sono utilizzabili a linea di comando ma sono anche stati integrati nell'editor online.

3.1 DOCX2ADF

DOCX2ADF è composto da due parti: una componente Java e un foglio si stile XSLT 2.0.

3.1.1 Caratteristiche generali di DOCX2ADF

Il programma Java si occupa di applicare il foglio di stile XSLT 2.0 al documento Office Open XML decompresso per produrre il documento HTML finale. Per raggiungere tale scopo compie i seguenti passi:

1. Prendere in input il file Office Open XML da convertire.
2. Prendere in input la cartella dove inserire il documento finale.
3. Estrarre i file XML dal documento Office Open XML, insieme alle immagini presenti in esso e inserire il tutto in una cartella temporanea.

4. Applicare il foglio di stile ai file XML e produrre il documento HTML finale.
5. Eliminazione della cartella di lavoro.

Il software Java viene fornito come un JAR (Java Archive) eseguibile a linea di comando, la sintassi per eseguirlo è la seguente:

```
java -jar docx2adf.jar -i <nome file.docx> -o <cartella di destinazione se non c'è viene creata> [ -content ] [ -topages ] [ -h ]

Comandi opzionali:

-content:    Specificando questa opzione verrà generato un documento in formato ADF che conterrà solo la parte dentro all'elemento <section id = "content> senza generare il frontmatter e gli indici delle sezioni, figure e tabelle.

-topages:    Attraverso questa opzione si avranno tutti i comportamenti di -content in più per ogni immagine in un formato non visualizzabile attraverso un browser verrà inserito un'elemento <span> vuoto.

-h:          Mostra un messaggio in cui viene spiegato come usare il programma di conversione a linea di comando e infine termina l'esecuzione.
```

Esempio 14: Sintassi per eseguire il software DOCX2ADF

La conversione è realizzata attraverso un unico foglio di stile, per realizzarlo mi sono basato su un precedente lavoro di tesi che si occupava di effettuare la conversione da documenti in formato Office Open XML a documenti RASH [12].
A causa della discrepanza che sussiste tra i formati Office Open XML e HTML non è possibile realizzare una conversione perfetta tra i due formati, perciò si devono adottare compromessi quando, nel processo di conversione, si incontrano elementi che non hanno un naturale corrispettivo nel formato in cui si va ad eseguire la conversione.

Per poter essere realizzata la conversione, i documenti in Word devono fare uso degli stili predefiniti, per esempio se si vuole che un elemento sia considerato come un titolo `<h1>` in HTML, in Word deve essere marcato attraverso lo stile titolo1.

3.1.2 Peculiarità dei documenti redatti dall'azienda ALSTOM nel formato Office Open XML

Prima che venisse sviluppato un editor in grado di redigere documenti direttamente nel formato ADF, i documenti dell'azienda ALSTOM erano creati direttamente con MS Word. Tutti i documenti redatti con MS Word per ALSTOM hanno una particolare struttura che ricalca un modello. Il modello che seguono può essere sintetizzato nel seguente modo:

1. all'inizio è presente una copertina in cui sono presenti i metadati, che descrivono e contestualizzano il documento, nella maggior parte dei casi occupa una o due pagine.
   Un esempio di come è graficamente organizzata è possibile vederlo nella Figura 4, al suo interno sono specificate le informazioni del progetto che il documento descrive.
   Alcune informazioni sono:
   - Nome del progetto (esempio Milano – Monza – Chiasso);
   - Committente;
   - Direzione lavori;
   - Appaltatore.

Figura 4: Tipico esempio di copertina dei documenti ALSTOM

2. a seguire troviamo gli indici delle sezioni, tabelle e figure, un esempio è possibile vederlo nelle figure: 5, 6, 7 rispettivamente.

SOMMARIO

Figura 5: Indice Sezioni

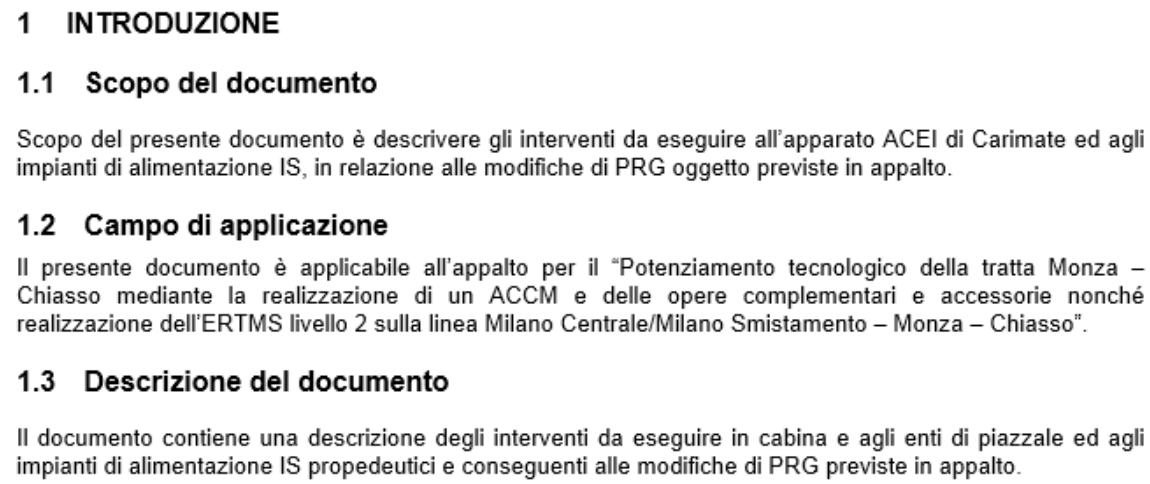
INDICE DELLE TABELLE

Figura 6: Indice Tabelle

INDICE DELLE FIGURE

Figura 7: Indice figure

3. infine il contenuto del documento, con le sezioni e le sottosezioni numerate. Ogni sezione contiene esattamente un titolo, una sottosezione viene specificata attraverso un titolo di livello inferiore rispetto alla sezione padre. Un esempio è disponibile nella: Figura 8 .



1 INTRODUZIONE

1.1 Scopo del documento

Scopo del presente documento è descrivere gli interventi da eseguire all'apparato ACEI di Carimate ed agli impianti di alimentazione IS, in relazione alle modifiche di PRG oggetto previste in appalto.

1.2 Campo di applicazione

Il presente documento è applicabile all'appalto per il "Potenziamento tecnologico della tratta Monza – Chiasso mediante la realizzazione di un ACCM e delle opere complementari e accessorie nonché realizzazione dell'ERTMS livello 2 sulla linea Milano Centrale/Milano Smistamento – Monza – Chiasso".

1.3 Descrizione del documento

Il documento contiene una descrizione degli interventi da eseguire in cabina e agli enti di piazzale ed agli impianti di alimentazione IS propedeutici e conseguenti alle modifiche di PRG previste in appalto.

Figura 8: Esempio di sezioni e sottosezioni in un documento dell'azienda ALSTOM nel formato Office Open XML

Un vincolo che è stato imposto nel processo di conversione verso il formato ADF è stato quello di rispettare la dimensione delle immagini. Nel momento in cui viene effettuata la conversione verso ADF la copertina è inserita all'interno di un particolare elemento di ADF denominato `<section id="frontmatter">`. Le pagine che immediatamente seguono la copertina, sono occupate dagli indici delle sezioni, tabelle, figure. Gli indici in MS Word sono trattati come dei campi, ovvero degli oggetti che sono creati automaticamente sulla base dei titoli o delle didascalie, presenti nel documento. Dal momento che ADF utilizza una particolare libreria scritta in JavaScript in grado di generare automaticamente gli indici delle sezioni, tabelle, figure non è stato necessario convertire questa parte.
Infine dopo gli indici è presente il contenuto del documento che formalmente inizia con il primo titolo di livello uno. Tutto ciò che è contenuto tra il primo titolo di livello 1 e la fine del documento, ad eccezione delle note a piè di pagina, nel processo di conversione verso ADF, sarà inserito nell'elemento di ADF: `<section id="content">.` Anche se non sono state trovate note a piè di pagina nei documenti forniti da ALSTOM, si è ritenuto di fornire comunque supporto a

questi elementi i quali in una futura versione del convertitore DOCX2ADF saranno inseriti all'interno dell'elemento ADF `<section id=”endnotes” />`.

3.1.3 Elementi gestiti da DOCX2ADF

Nella Tabella 9 elenco gli elementi di Office Open XML che hanno una naturale corrispondenza nel linguaggio HTML 5, ovvero quelli che vengono correttamente convertiti in elementi HTML.

Elementi gestiti da DOCX2ADF	**Note**
Titoli	È necessario utilizzare gli stili predefiniti di Word a seconda del livello del titolo es: Titolo1, Titolo2, ...
Paragrafi	I paragrafi vengono correttamente convertiti.
Tabelle	All'interno delle celle delle tabelle è possibile inserire qualsiasi oggetto come figure, liste o altre tabelle.
Liste	Dal momento che HTML permette solo due tipi di liste è possibile utilizzare solo liste ordinate o non ordinate. Vengono gestite anche le liste con più livelli di indentazione.
Immagini	Sono permesse solo le immagini in un formato visualizzabile anche in HTML, es: png, jpeg. La dimensione delle immagini viene preservata.
Riferimenti	È possibile inserire riferimenti a sezioni, immagini, tabelle e a qualsiasi altro elemento oltre che a siti web esterni.
Testo in grassetto, corsivo, sottolineato, con apici e con pedici	Il testo in **grassetto**, *corsivo*, <u>sottolineato</u>, con $^{\text{apici}}$ e con $_{\text{pedici}}$ viene correttamente convertito in HTML. Sono le

Elementi gestiti da DOCX2ADF	Note
	uniche caratterizzazioni tipografiche che è possibile specificare nel documento Office Open XML da convertire; tutte le altre vengono ignorate.
Didascalie delle tabelle e delle figure	Si deve usare lo stile didascalia, disponibile come stile predefinito in MS Word.
Campi	I campi sono una particolare funzionalità che MS Word mette a disposizione per generare automaticamente alcune informazioni come il numero di pagine o la pagina corrente. È essenzialmente un modo per inserire delle variabili all'interno di un documento. Dal momento che in ADF non esiste tale funzionalità i campi verranno tradotti come normale testo.
Caselle di testo	Le caselle di testo sono particolari sezioni all'interno del documento Office Open XML che non seguono il flusso di testo, ma possono essere posizionate in qualsiasi posto all'interno del documento. Possono contenere qualsiasi genere di elementi. In ADF questo elemento è convertito come una normale sezione di testo.

Tabella 9: Elementi gestiti correttamente dal convertitore DOCX2ADF

3.2 ADF2DOCX

Il secondo convertitore che ho realizzato, ADF2DOCX partendo da un file nel formato ADF realizza un documento nel formato Office Open XML visualizzabile correttamente attraverso MS Word.

La Figura 9 sintetizza l'operato di DOCX2ADF e ADF2DOCX.

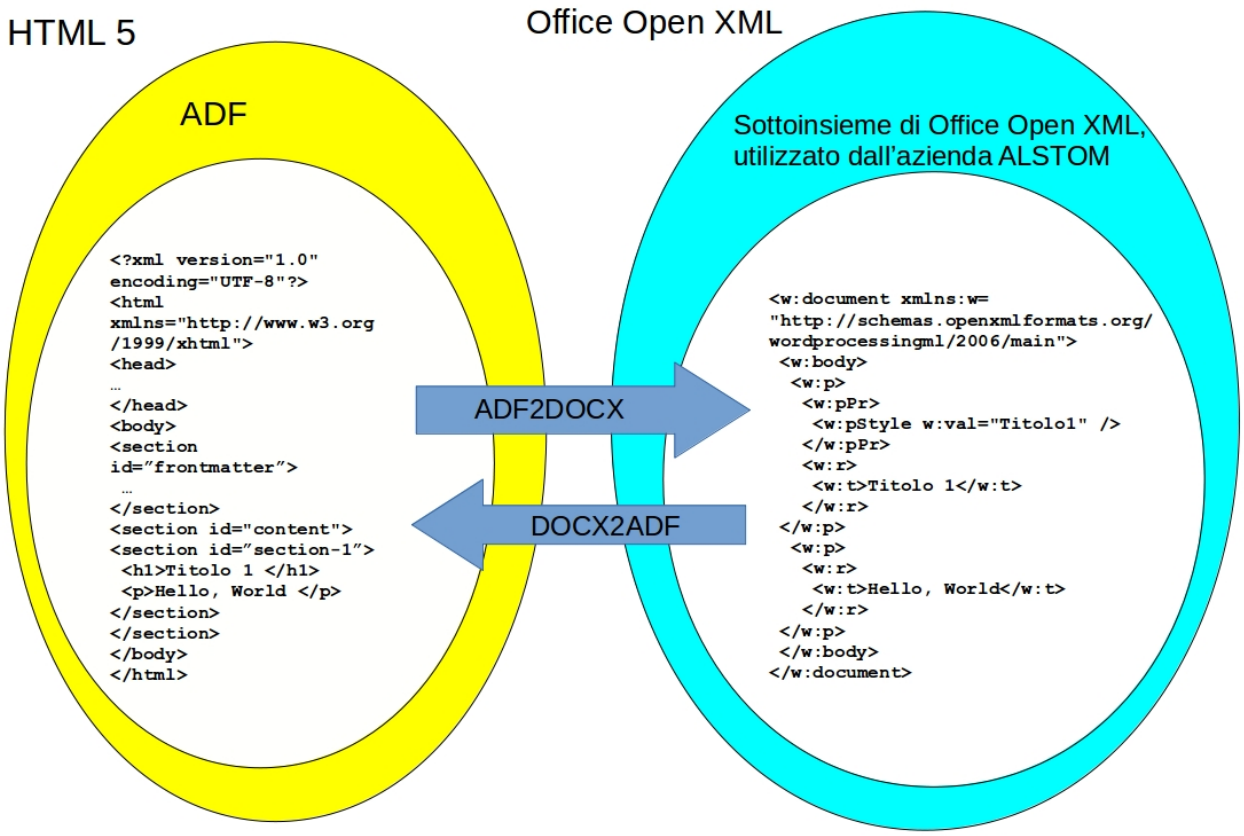


Figura 9: DOCX2ADF e ADF2DOCX a confronto

3.2.1 Caratteristiche generali

ADF2DOCX è composto da due parti principali:

- Una componente Java che si occupa di preparare l'infrastruttura generale,
- Una parte XSLT la quale realizza la conversione, attraverso vari fogli di stile XSLT 2.0. Ho deciso di realizzare più fogli di stile al posto di uno, come avrò modo di spiegare nel prossimo capitolo, in quanto essendo il documento Office Open XML composto da più file XML, un unico foglio di stile sarebbe risultato troppo complesso da gestire.

La componente Java si occupa di:

1. Prendere in input il file HTML da convertire.
2. Prendere in input il nome che sarà attribuito al file di output, l'estensione deve essere .docx.
3. Applicare i fogli di stile XSLT al documento HTML preso in input.
4. Aggiungere i file XML, prodotti dall'applicazione dei fogli di stile XSLT al documento HTML, ad una cartella temporanea.
5. Aggiungere alla cartella temporanea un foglio di stile di default denominato `styles.xml`, precedentemente concordato con l'azienda ALSTOM, per la corretta visualizzazione del documento.
6. Comprimere la cartella temporanea creando in questo modo il file Office Open XML visualizzabile attraverso MS Word.

Una caratteristica interessante è la possibilità di specificare un foglio di stile al documento Office Open XML finale prodotto dalla conversione in alternativa a quello di default in modo da specificare alcune proprietà del documento, come il tipo e la dimensione dei font nonché di imporre una numerazione delle sezioni al documento. Per preparare un foglio di stile è necessario scrivere un documento Office Open XML con le caratteristiche desiderate ed in seguito estrarre il file `styles.xml` all'interno della cartella `/word` e fornirlo in input al programma come specificato nell'Esempio 15.

La sintassi per eseguire ADF2DOCX è la seguente:

```
java -jar adf2docx.jar -i <file.html da convertire> -o <nome
file.docx che sarà prodotto in output> [ -style styles.xml ]
[ -content ] [ -h ]

Comandi opzionali:

-style styles.xml:     Se viene specificato questo comando, che
                       richiede obbligatoriamente un file in input
                       di nome styles.xml, al documento Office Open
                       XML finale viene fornito un particolare
                       foglio di stile in sostituzione a quello di
                       default.

-content:              Specificando questa opzione, nel documento
                       Office Open XML finale non vengono inclusi
                       i metadati e gli indici delle sezioni,
                       tabelle e figure, ovvero non viene
                       effettuata la conversione di tutto il
                       contenuto incluso nell'elemento <section
                       id="frontmatter> all'interno del documento
                       HTML.

-h:                    Viene mostrato un messaggio in cui viene
                       mostrato come utilizzare ADF2DOCX.
```

Esempio 15: Sintassi per eseguire ADF2DOCXF

3.2.2 Peculiarità di ADF2DOCX

Una caratteristica innovativa di ADF2DOCX rispetto ai software presentati nel paragrafo 2.4 è la possibilità di inserire al interno del documento Office Open XML intestazioni e piè di pagina recuperandoli da una particolare sezione all'interno del documento HTML che si intende convertire.
Oltre alle intestazioni e piè di pagina che un utente può specificare o meno, ogni pagina prodotta da ADF2DOCX, tranne la prima, in quanto essa è dedicata alla rappresentazione dei metadati e funge da copertina, avrà indicato nell'intestazione il numero di pagina corrente, seguito dal numero di pagine totali.

L'inserimento dei numeri di pagina non viene recuperato dal documento HTML ma viene calcolato da MS Word all'apertura del documento, in quanto viene inserito come uno speciale campo.
Quest'ultimo approccio è utilizzato anche per l'inserimento degli indici di sezioni, tabelle e figure.
La scelta di utilizzare la modalità dei campi ovvero di particolari oggetti che vengono automaticamente aggiornati, all'apertura del documento attraverso MS Word, per la produzione dei numeri di pagina e degli indici è dettata dal fatto che l'aggiunta di un nuovo elemento provocherebbe l'aggiornamento automatico di queste sezioni senza la necessità di un aggiornamento manuale.

3.2.3 Elementi ADF correttamente convertiti nel formato Office Open XML

Nel seguito discuterò degli elementi ADF che vengono correttamente gestiti nel processo di conversione da ADF a Office Open XML.
Nella fase di conversione non vengono inclusi gli elementi che vengono automaticamente generati nel file di destinazione Office Open XML.
Gli elementi che vengono effettivamente convertiti sono i seguenti:

- **Titoli**: Nel formato ADF per specificare i titoli delle sezioni e sottosezioni sono permessi solo elementi `<h1>`, a seconda del livello di indentazione in cui si trovano verranno resi correttamente nel formato Office Open Xml. Ad esempio, nel seguente frammento ADF:

```
<section>
  <h1> Titolo di livello uno </h1>
  <p> Testo nella prima sezione </p>
  <section>
    <h1> Titolo di livello due </h1>
    <p> Testo nella sottosezione </p>
  </section>
</section>
```

Esempio 16: Sezione e sottosezione in ADF

  il primo elemento `<h1>` è reso come un titolo di livello uno mentre il secondo come un titolo di livello due attraverso gli stili predefiniti che mette a disposizione MS Word;

La traduzione dell'Esempio 16 operata da ADF2DOCX è visibile nella seguente figura:

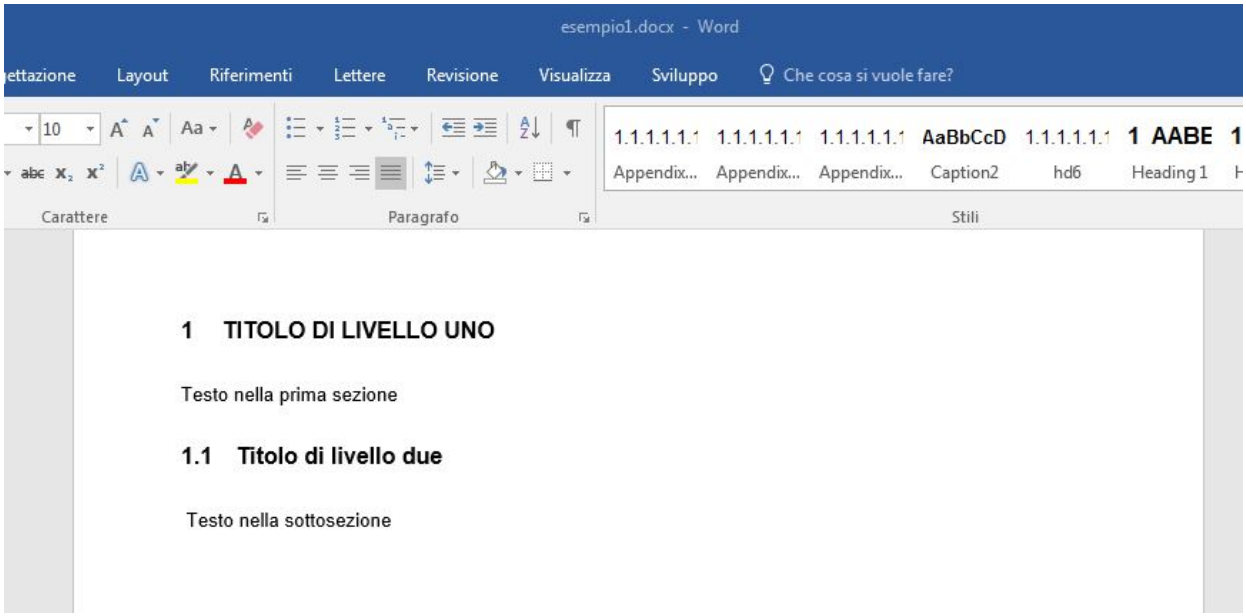


Figura 10 File Office Open XML prodotto da ADF2DOCX

È possibile notare come in Figura 10 vengano automaticamente numerate le sezioni e le sottosezioni.

- **Paragrafi**;
- **Testo in grassetto, corsivo, con apici e pedici;**
- **Riferimenti:** I riferimenti possono essere a parti interne del documento come a sezioni, immagini tabelle o a parti esterne del documento come siti web. Inoltre qualsiasi paragrafo può essere referenziato.
  Un requisito richiesto è che per essere correttamente gestiti i riferimenti a immagini o tabelle è che esse debbano avere le didascalie;
- **Tabelle:** Quando una tabella ricopre più di una pagina le righe di intestazione vengono ripetute;
- **Didascalie**.
- **Intestazioni e Piè di pagina**
  Le intestazioni e i piè di pagina sono parti fissi all'interno di un documento Office Open XML posizionate rispettivamente nella parte alta e bassa di una singola pagina. I documenti Office Open XML prodotti da

ADF2DOCX hanno un'intestazione fissa che indica il numero di pagina corrente e quello totale. Nella prima pagina che è dedicata alla copertina non sono inserite intestazioni o piè di pagina. Dal momento che in ADF ed in HTML non esiste tale concetto, si è dovuto usare una convenzione. Nel seguente esempio mostro come sia possibile specificare un intestazione con testo e due immagine in un documento ADF:

```
<section id="layout-areas" style="display:none;">
   <section id="header">
      <!-- Contenuto Header -->
      <p>Un'immagine di Alstom qui: <img src="image3.png" /> e
      un'altra qua: <img src="image4.png" /> </p>
   </section>
   <section id="footer">
      <!-- Contenuto FOOTER -->
   </section>
</section>
```

Esempio 17: Esempio di intestazione in un documento nel formato ADF

In un documento ADF come si può notare dall'attributo `style="display:none;"`, le intestazioni e i piè di pagina non saranno visibili.

La traduzione dell'Esempio 17 nel formato Office Open XML operata da ADF2DOCX è visibile nella seguente figura:

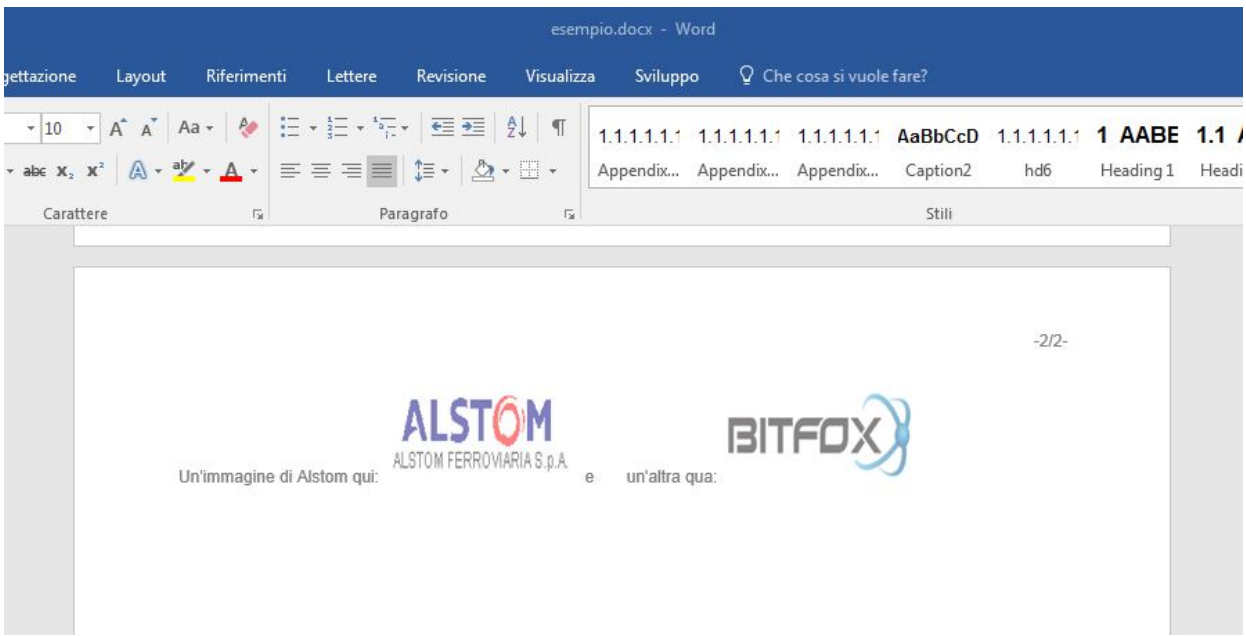


Figura 11: Intestazione in un documento Office Open XML prodotto da ADF2DOCX

È possibile notare come sia inserito di default il numero di pagina corrente e quello che rappresenta il numero di pagine totali.

- **Liste**: Le liste vengono correttamente gestite, possono essere di due tipi: liste ordinate o non ordinate. Una caratteristica interessante è la possibilità di specificare anche liste annidate. Nei prossimi due esempi mostro come una lista annidata viene scritta nel formato ADF e successivamente convertita da ADF2DOCX nel formato Office Open XML.

```
<h1> <span role="pdx-hcounter"> </span>Liste:</h1>
<p>Loscopo di ALSTOM include:</p>
<ul>
   <li>
      <p>IXL: funzionalità ACCM,</p>
   </li>
   <li>
      <p>ETCS: funzionalità ETCS L2 BL3 (Terra e Bordo),   </p>
   </li>
   <li>
      <p>ATP Nazionale: funzionalità SCMT, </p>
   </li>
   <li>
      <p>funzionalità ausiliarie</p>
      <ul>
         <li>
            <p>POWER SUPPLY: Alimentazioni Impianti di
Segnalamento (ALIM IS),</p>
         </li>
         <li>
            <p>Diagnostica e Manutenzione di tutte le nuove
Alimentazioni (D-M),</p>
         </li>
         <li>
            <p>N-T:RETI impianti ALSTOM e Telefonia di Linea
(STSI),</p>
         </li>
         <li>
            <p>Security degli Shelter: Anti-Intrusione/Controllo
Accessi (AI-CA), Rilevamento Incendi (RI) e Telesorveglianza
(TVCC).</p>
         </li>
      </ul>
   </li>
</ul>
```

Esempio 18: Lista annidata nel formato ADF

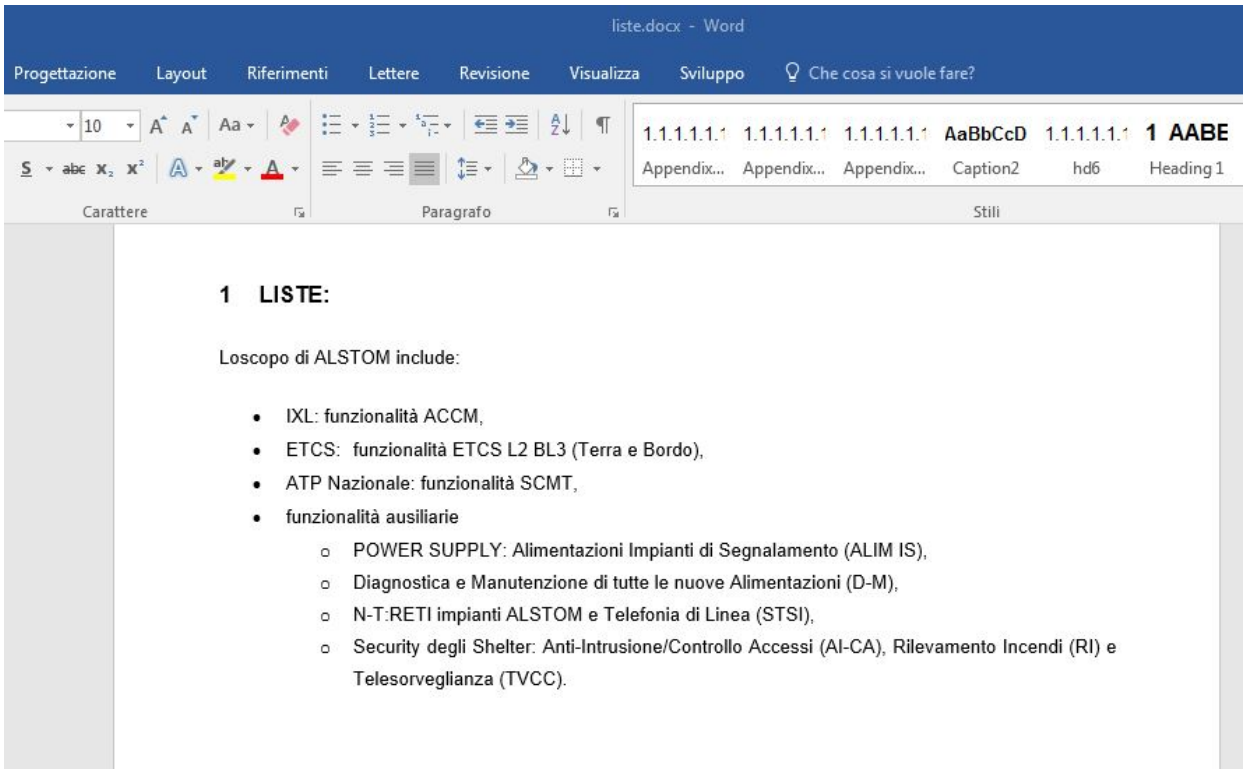


Figura 12: Lista prodotta da ADF2DOCX nel formato Office Open XML

Gli elementi che ho mostrato nella precedente lista sono tutti quelli che vengono correttamente convertiti nel formato Office Open XML ad opera di ADF2DOCX.

Capitolo 4

Dettagli implementativi di DOCX2ADF e ADF2DOCX

Nel presente capitolo, descrivo i dettagli implementativi dei convertitori DOCX2ADF e ADF2DOCX, soffermandomi sulle corrispondenze tra ADF e Office Open XML e viceversa. In particolare presento vari frammenti di ADF che descrivono paragrafi, tabelle, liste e li confronto con la loro traduzione in Office Open XML operata da ADF2DOCX. Eseguirò la stessa operazione per le traduzioni da Office Open XML a ADF operate da DOCX2ADF, solo nei casi in cui l'operazione non sia esattamente l'opposta da quella eseguita da ADF2DOCX. Per ogni conversione indicherò se la conversione viene eseguita solo in un verso piuttosto che in tutti e due. Per esempio in Office Open XML, le immagini possono essere rappresentate in due modi, nel processo di conversione verso ADF li gestisco entrambi mentre nell'altro verso utilizzo solo un metodo.
Prima di scendere nel merito delle conversioni farò una precisazione sui fogli di stile utilizzati da ADF2DOCX.

4.1 Precisazione sui fogli di stile utilizzati da ADF2DOCX

ADF2DOCX, utilizza tre fogli di stile denominati `numb.xsl`, `rel.xsl`, `fromadf2docx.xsl` per effettuare la conversione da ADF a Office Open XML. Ho scelto di utilizzare tre fogli di stile piuttosto che uno solo come nel caso di DOCX2ADF in quanto essendo un file Office Open XML composto da più parti XML risulta più conveniente dividere i compiti.
Il foglio di stile `numb.xsl` si occupa di creare il file `word/numbering.xml`, il quale, come ho avuto modo di mostrare nel paragrafo 2.1.3 al suo interno contiene un particolare elemento, contrassegnato da un id, univoco per ogni lista,

che contiene informazioni su ognuna di essa, come il fatto che sia ordinata o meno.
Il foglio di stile `rel.xsl` si occupa di gestire tutti i riferimenti a immagini o a siti web presenti all'interno del documento Office Open XML. Per ogni immagine crea un frammento come quello visualizzato nell'Esempio 3 a pagina 10.
Per ogni riferimento ad un sito web, viene creato il seguente frammento:

```
<Relationship Id="rId6"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relat
ionships/hyperlink" Target="http://www.google.com"
TargetMode="External"/>
```

Esempio 19: Elemento creato per ogni sito web

Il file `fromadf2docx.xsl` si occupa di gestire tutti gli elementi presenti all' interno del documento ADF, che analizzo nel prossimo paragrafo in dettaglio, in più inserisce alcuni elementi di default in ogni documento Office Open XML verso cui effettua la conversione, come gli indici delle sezioni, tabelle o figure. In ogni intestazione, vengono inseriti il numero di pagina corrente e il numero di pagine totali. I numeri di pagina e gli indici vengono inseriti come campi; i campi in Office Open XML sono un modo per inserire variabili all'interno del documento, sono essenzialmente dei "placeholders" per dati che possono cambiare.
Nelle prossime pagine presento gli elementi di default inseriti da ADF2DOCX e in seguito, vari frammenti XML deputati alla rappresentazione degli elementi presentati precedentemente nei paragrafi 3.1.3 e 3.2.3.

4.1.1 Elementi di default inseriti da ADF2DOCX

Il numero di pagina corrente e quello che rappresenta il numero di pagine totali vengono inseriti nel file `word/header1.xml`, deputato alla rappresentazione dell'intestazione; il frammento Office Open XML che li rappresenta è il seguente:

```
<w:p>
            <w:pPr>
                  <w:pStyle w:val="Intestazione"/>
                  <w:jc w:val="right"/>
            </w:pPr>
            <w:r>
                  <w:t>-</w:t>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="begin"/>
            </w:r>
            <w:r>
                  <w:instrText xml:space="preserve"> PAGE   \* MERGEFORMAT
</w:instrText>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="separate"/>
            </w:r>
            <w:r w:rsidR="00EA3C8E">
                  <w:rPr>
                        <w:noProof/>
                  </w:rPr>
                  <w:t>0</w:t>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="end"/>
            </w:r>
            <w:r>
                  <w:t>/</w:t>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="begin"/>
            </w:r>
            <w:r>
                  <w:instrText xml:space="preserve"> NUMPAGES   \* MERGEFORMAT
</w:instrText>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="separate"/>
            </w:r>
            <w:r w:rsidR="00EA3C8E">
                  <w:rPr>
                        <w:noProof/>
                  </w:rPr>
                  <w:t>0</w:t>
            </w:r>
            <w:r>
                  <w:fldChar w:fldCharType="end"/>
            </w:r>
            <w:r>
                  <w:t>-</w:t>
            </w:r>
            </w:r>
      </w:p>
```

Esempio 20: Frammento Office Open XML per rappresentare il numero di pagina corrente e il numero di pagine totali

L'indice delle sezioni il quale viene inserito nel file `document.xml` ha la seguente forma:

```
<w:sdt>
	<w:sdtPr>
		<w:id w:val="-651286457"/>
		<w:docPartObj>
			<w:docPartGallery w:val="Table of Contents"/>
			<w:docPartUnique/>
		</w:docPartObj>
	</w:sdtPr>
	<w:sdtEndPr>
		<w:rPr>
			<w:b/>
			<w:bCs/>
		</w:rPr>
	</w:sdtEndPr>
	<w:sdtContent>
		<w:p >
			<w:pPr>
				<w:pStyle w:val="Titolosommario"/>
				<w:rPr>
					<w:lang w:val="it-IT"/>
				</w:rPr>
			</w:pPr>
			<w:r>
				<w:rPr>
					<w:lang w:val="it-IT"/>
				</w:rPr>
				<w:t>Sommario</w:t>
			</w:r>
		</w:p>
		<w:p >
			<w:r>
				<w:fldChar w:fldCharType="begin"  />
			</w:r>
			<w:r>
				<w:instrText xml:space="preserve"> TOC \o "1-3" \h \z \u
</w:instrText>
			</w:r>
			<w:r>
				<w:fldChar w:fldCharType="separate"/>
			</w:r>
			<w:r>
				<w:rPr>
					<w:rFonts w:hAnsiTheme="minorHAnsi"/>
					<w:b/>
					<w:bCs/>
					<w:noProof/>
				</w:rPr>
				<w:t>Aggiorna il sommario cliccando qui, e in seguito, in alto,
sul pulsante aggiorna sommario</w:t>
			</w:r>
			<w:r>
				<w:rPr>
					<w:b/>
					<w:bCs/>
				</w:rPr>
				<w:fldChar w:fldCharType="end"/>
			</w:r>
		</w:p>
	</w:sdtContent>
</w:sdt>
```

Esempio 21: Frammento Office Open XML per rappresentare l'indice delle sezioni

L'indice delle figure che viene inserito nel file `document.xml`, insieme a quello delle tabelle ha la seguente forma:

```
<w:p >
     <w:pPr>
          <w:pStyle w:val="Titolosommario"/>
     </w:pPr>
     <w:r>
          <w:t>Indice Figure</w:t>
     </w:r>
</w:p>
<w:p >
     <w:r>
          <w:fldChar w:fldCharType="begin"/>
     </w:r>
     <w:r>
          <w:instrText xml:space="preserve"> TOC \h \z \c "Figura"
</w:instrText>
     </w:r>
     <w:r>
          <w:fldChar w:fldCharType="separate"/>
     </w:r>
     <w:r>
          <w:rPr>
               <w:rFonts w:hAnsiTheme="minorHAnsi"/>
               <w:b/>
               <w:bCs/>
               <w:noProof/>
          </w:rPr>
          <w:t>Aggiorna l'indice delle figure cliccando il tasto
destro del mouse e scegliendo aggiorna campo</w:t>
     </w:r>
     <w:r>
          <w:fldChar w:fldCharType="end"/>
     </w:r>
</w:p>
```

Esempio 22: Frammento Office Open XML per rappresentare l'indice delle figure

L'indice delle tabelle ha la stessa forma di quello per le figure, tranne il fatto che la stringa `<w:instrText xml:space="preserve"> TOC \h \z \c "Figura" </w:instrText>` viene sostituita con `<w:instrText xml:space="preserve"> TOC \h \z \c "Tabella" </w:instrText>`.

4.2 Corrispondenze tra ADF e Office Open XML

Nel presente paragrafo mostro i risultati dei convertitori DOCX2ADF e ADF2DOCX , attraverso i frammenti XML prodotti da entrambi. In particolare, esaminerò frammenti di ADF che rappresentano elementi come paragrafi e tabelle e li affiancherò alla loro traduzione, operata da ADF2DOCX, nel formato Office Open XML. Se no è espressamente indicato la conversione si intende valida anche nell'altro verso ovvero da Office Open XML ad ADF.

4.2.1 Struttura generale

La struttura ad alto livello di un documento ADF rispecchia quella di Office Open XML come si evince dai frammenti riportati in sequenza.

```
<html xmlns="http://www.w3.org/1999/xhtml"  >
   <head>
      <link    ...  />
      <script  ...> </script>
      <script  type="application/ld+json"   id="doc-metadata">
         <--   Metadata  about     the  document  -->
      </script>
   </head>
   <body>
      <section id="frontmatter"    role="doc-frontmatter"
     class="doc-ro">
        <!-- contenuto frontmatter -->
      </section>
      <section id="content"   role="doc-content">
         <section>
            <h1>
               <span role="pdx-hcounter"> </span> Titolo
            </h1>
            <p>Hello, world.</p>
         </section>
      </section>
   </body>
</html>
```

Esempio 23: Struttura ad alto livello documento ADF

L'elemento `<span role="pdx-hcounter"> </span>` viene impiegato per la numerazione automatica delle sezioni in un documento ADF.

```
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/
main">
   <w:body>
      <w:p>
         <w:pPr>
            <w:pStyle w:val="Titolo1"/>
         </w:pPr>
         <w:r>
            <w:t xml:space="preserve">Titolo </w:t>
         </w:r>
      </w:p>
      <w:p>
         <w:r>
            <w:t>Hello, world.</w:t>
         </w:r>
      </w:p>
   </w:body>
</w:document>
```

Esempio 24: Struttura ad alto livello documento Office Open XML

4.2.2 Riferimenti

Nei prossimi due esempi viene mostrato come vengono gestiti i riferimenti esterni e interni.

4.2.2.1 Riferimenti esterni

I riferimenti esterni in Office Open Xml sono composti da due parti, una parte in cui viene segnalata la presenza del riferimento, presente all'interno del file `document.xml` e una parte che contiene l'informazione riguardo al riferimento, all'interno del file `document.xml.rels.` Le due informazioni precedenti, sono collegate per mezzo di un id. I riferimenti esterni in Office Open XML sono un tipico esempio di riferimenti espliciti come ho spiegato nel paragrafo 2.1.3 a pagina 4.

```
<p>Un riferimento a <a href=”http://www.google.it”> Google </a><p>
```

Esempio 25: Riferimento esterno in ADF

```
<w:p>
   <w:r>
      <w:t>Un riferimento a:</w:t>
   </w:r>
   <w:hyperlink r:id="rId4">
      <w:r>
         <w:rPr>
            <w:rStyle w:val="Hyperlink"/>
         </w:rPr>
         <w:t>Google</w:t>
      </w:r>
   </w:hyperlink>
</w:p>
```

Esempio 26: Riferimento esterno in Office Open XML, presente all'interno del file document.xml

```
<Relationship Id="rId4" Type="http://. . ./hyperlink"
Target="http://www.google.com/" TargetMode="External"/>
```

Esempio 27: Riferimento esterno in Office Open XML, presente all'interno del file document.xml.rels

4.2.2.2 Riferimenti interni

I riferimenti a qualsiasi paragrafo, figura o immagine del documento vengono gestiti solo nel processo di conversione dal formato ADF ad Office Open XML ad opera di ADF2DOCX.

Quando si va ad effettuare la conversione dal formato Office Open XML ad ADF invece i riferimenti interni vengono gestiti solamente se diretti a sezioni, sottosezioni , elementi di una lista numerata, immagini o tabelle. Le immagini o le tabelle devono contenere una didascalia se sono l'obbiettivo di un riferimento.

```
<p id="àncora"> Paragrafo con un segnalibro </p>

<p>In questo paragrafo si ha un link al paragrafo <a
href="#àncora"> più sopra </a></p>
```

Esempio 28: Riferimento interno ADF

```
<w:document >
   <w:body>

      <w:p >
         <w:bookmarkStart w:id="0" w:name="àncora"/>
         <w:r>
            <w:t>Paragrafo con un segnalibro</w:t>
         </w:r>
      </w:p>
      <w:bookmarkEnd w:id="0"/>


      <w:p >
         <w:r>
            <w:t xml:space="preserve">In questo paragrafo si ha un
link al paragrafo </w:t>
         </w:r>
         <w:r>
            <w:fldChar w:fldCharType="begin"/>
         </w:r>
         <w:r>
            <w:instrText xml:space="preserve"> REF àncora \p \h
</w:instrText>
         </w:r>
         <w:r>
            <w:fldChar w:fldCharType="separate"/>
         </w:r>
         <w:r>
            <w:t xml:space="preserve">più sopra</w:t>
         </w:r>
         <w:r>
            <w:fldChar w:fldCharType="end"/>
         </w:r>
      </w:p>
   </w:body>
</w:document>
```

Esempio 29: Riferimenti interni in Office Open XML

4.2.3 Liste

Le liste in Office Open XML, sono specificate in due file distinti.

1. Il contenuto è specificato nel file `document.xml`; un elemento di una lista è distinto da un normale paragrafo, in quanto all'interno del campo dedicato alle proprietà contiene l'elemento `<w:numpr>`, che a sua volta contiene gli elementi :

- `<w:ilvl>` : specifica il livello di un elemento della lista;
- `<w:numId>` : inserisce un id univoco per ogni lista.

2. Le informazioni riguardo alla lista come il fatto che sia ordinata o meno sono inserite all'interno del file `numbering.xml`, ogni lista viene distinta dalle altre attraverso un id univoco.

```
<ol>
   <li><p>Primo elemento</p></li>
   <li><p>Secondo elemento</p></li>
</ol>
```

Esempio 30: Lista numerata nel formato ADF

```
<w:p>
   <w:pPr>
      <w:pStyle w:val="Paragrafoelenco"/>
      <w:numPr>
         <w:ilvl w:val="0"/>
         <w:numId w:val="1"/>
      </w:numPr>
   </w:pPr>
   <w:r>
      <w:t>Primo elemento</w:t>
   </w:r>
</w:p>
<w:p>
   <w:pPr>
      <w:pStyle w:val="Paragrafoelenco"/>
      <w:numPr>
         <w:ilvl w:val="0"/>
         <w:numId w:val="1"/>
      </w:numPr>
   </w:pPr>
   <w:r>
      <w:t>Secondo elemento</w:t>
   </w:r>
</w:p>
```

Esempio 31: Lista ordinata con due elementi nel formato Office Open XML, specificata all'interno del file document.xml

```
<w:numbering >
   <w:abstractNum w:abstractNumId="0" >
   <w:nsid w:val="0F101145"/>
   <w:multiLevelType w:val="hybridMultilevel"/>
   <w:tmpl w:val="835AB472"/>
   <w:lvl w:ilvl="0" w:tplc="0409000F">
      <w:start w:val="1"/>
      <w:numFmt w:val="decimal"/>
      <w:lvlText w:val="%1."/>
      <w:lvlJc w:val="left"/>
      <w:pPr>
         <w:ind w:left="720" w:hanging="360"/>
      </w:pPr>
   </w:lvl>
   <w:num w:numId="1">
      <w:abstractNumId w:val="0"/>
   </w:num>
</w:numbering>
```

Esempio 32: Frammento Office Open XML inserito all'interno del file word/numbering.xml

4.2.4 Tabelle

Le tabelle nel formato ADF sono racchiuse all'interno dell'elemento `<figure>` che mediante il tag `<figcaption>` specifica la didascalia.

```
<figure data-pdx-family="tables">
   <table>
      <tr>
         <th> Titolo 1 </th>
         <th> Titolo 2 </th>
      </tr>
      <tr>
         <td> Cella 1 </td>
         <td> Cella 2 </td>
      </tr>
   </table>
   <figcaption>
      Tabella  <span     role="pdx-incrementer"   data-pdx-
family="tables"> </span> Didascalia
   </figcaption>
</figure>
```

Esempio 33: Tabella con didascalia nel formato ADF

```
<w:tbl>
   <w:tblPr>
      <w:tblStyle w:val="Grigliatabella"/>
      <w:tblW w:w="0" w:type="auto"/>
      <w:tblLook w:val="04A0" w:firstRow="1" w:lastRow="0"
w:firstColumn="1" w:lastColumn="0" w:noHBand="0" w:noVBand="1"/>
   </w:tblPr>
   <w:tblGrid>
      <w:gridCol w:w="4814"/>
      <w:gridCol w:w="4814"/>
   </w:tblGrid>
   <w:tr >
      <w:tc>
         <w:p>
            <w:r>
               <w:t>Titolo 1</w:t>
            </w:r>
         </w:p>
      </w:tc>
      <w:tc>
         <w:p >
            <w:r>
               <w:t>Titolo 2</w:t>
            </w:r>
         </w:p>
      </w:tc>
   </w:tr>
   <w:tr>
      <w:tc>
         <w:p>
            <w:r>
               <w:t>Cella 1</w:t>
            </w:r>
         </w:p>
      </w:tc>
      <w:tc>
         <w:p >
            <w:r>
               <w:t>Cella 2</w:t>
            </w:r>
         </w:p>
      </w:tc>
   </w:tr>
</w:tbl>
```

Esempio 34: Tabella nel formato Office Open XML senza didascalia

```
<w:p>
   <w:pPr>
      <w:pStyle w:val="Didascalia"/>
   </w:pPr>
   <w:r>
      <w:t>Tabella </w:t>
   </w:r>
   <w:r>
      <w:fldChar w:fldCharType="begin"/>
   </w:r>
   <w:r>
      <w:instrText xml:space="preserve"> SEQ Tabella \* ARABIC
</w:instrText>
   </w:r>
   <w:r>
      <w:fldChar w:fldCharType="separate"/>
   </w:r>
   <w:r>
      <w:t>1</w:t>
   </w:r>
   <w:r>
      <w:fldChar w:fldCharType="end"/>
   </w:r>
   <w:r>
      <w:t xml:space="preserve"> Didascalia</w:t>
   </w:r>
</w:p>
```

Esempio 35: Didascalia di una tabella nel formato Office Open XML

4.2.5 Immagini

```
<figure   id="figure_4"  data-pdx-family="images">
   <p><img src="img/figure4.png" alt="figura 4" /></p>
   <figcaption>
      Figura <span role="pdx-incrementer" data-pdx-
family="images">4</span>Esempio
   </figcaption>
</figure>
```

Esempio 36: Immagine nel formato ADF

La sintassi per la didascalia di un'immagine nel formato Office Open XML è uguale a quella impiegata per le tabelle nell'Esempio 35 a pagina 57.

L'unica differenza riguarda l'uso dell'elemento `<w:instrText xml:space="preserve">SEQ Figura \* ARABIC</w:instrText>` al posto di `<w:instrText xml:space="preserve"> SEQ Tabella \* ARABIC </w:instrText>`.

```
<w:p>
   <w:r>
      <w:drawing>
         <wp:inline distT="0" distB="0" distL="0" distR="0">
            <wp:extent cx="1685925" cy="1264531"/>
            <wp:docPr id="4" name="Esempio"/>
            <wp:cNvGraphicFramePr>
               <a:graphicFrameLocks noChangeAspect="1"/>
            </wp:cNvGraphicFramePr>
            <a:graphic>
               <a:graphicData
uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                  <pic:pic>
                     <pic:nvPicPr>
                        <pic:cNvPr id="4" name="Esempio"/>
                        <pic:cNvPicPr/>
                     </pic:nvPicPr>
                     <pic:blipFill>
                        <a:blip r:embed="rId5" cstate="print" />
                        <a:stretch>
                           <a:fillRect/>
                        </a:stretch>
                     </pic:blipFill>
                     <pic:spPr>
                        <a:xfrm>
                           <a:off x="0" y="0"/>
                           <a:ext cx="1692806" cy="1269692"/>
                        </a:xfrm>
                        <a:prstGeom prst="rect">
                           <a:avLst/>
                        </a:prstGeom>
                     </pic:spPr>
                  </pic:pic>
               </a:graphicData>
            </a:graphic>
         </wp:inline>
      </w:drawing>
   </w:r>
</w:p>
```

Esempio 37: Immagine nel formato Office Open XML

Come nel caso dei riferimenti la locazione di un'immagine è specificata nel file `_rels/document.xml.rels`. La dimensione di un immagine è specificata dall'elemento `<wp:extent cx="1685925" cy="1264531"/>` , cx rappresenta la larghezza e cy l'altezza. La larghezza e l'altezza sono espresse in English Metric Units[7], 1cm= 3600 English Metric Units.

```
<Relationships>
   <Relationship Id="rId5"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relat
ionships/image" Target="media/Esempio.png"/>
</Relationships>
```

Esempio 38: Frammento di Office Open XML il quale indica la locazione dell'immagine con id rId5

4.2.5.1 Modo alternativo per la rappresentazione delle immagini nel formato Office Open XML

Nell'esempio che segue è presente un frammento di Office Open XML impiegato per specificare un'immagine, di norma è usato il metodo specificato nel paragrafo precedente, perciò questo caso viene gestito solo nel processo di conversione da Office Open XML ad ADF.

```
<w:p>
   <w:r>
      <w:object w:dxaOrig="13845" w:dyaOrig="7575">
         <v:shape id="_x0000_i1026" type="#_x0000_t75"
style="width:279.75pt;height:532.5pt" o:ole="">
            <v:imagedata r:id="rId5" o:title="" croptop="6377f"
cropbottom="1962f" cropleft="20932f" cropright="28178f"/>
         </v:shape>
      </w:object>
   </w:r>
</w:p>
```

Esempio 39: Modo alternativo per la rappresentazione delle immagini nel formato Office Open XML

7 http://archive.oreilly.com/pub/post/what_is_an_emu.html

Capitolo 5

Scelte implementative e valutazioni

Dopo aver discusso delle caratteristiche generali e tecniche dei convertitori realizzati, procedo nell'analisi dei motivi che mi hanno portato a scegliere Java e XSLT per realizzare i convertitori presentati in questa dissertazione.
In conclusione farò un analisi del processo di sviluppo e dei limiti dei convertitori.

5.1 Scelte implementative

Ho deciso di utilizzare XSLT [13] [14] nella sua versione 2.0 in quanto dovendo trasformare file XML in altri file XML, e essendo questo lo scopo di XSLT la scelta è stata naturale.
Per redigere il foglio di stile del convertitore DOCX2ADF, mi sono basato su un precedente lavoro di tesi, per la laurea triennale in informatica [12], che svolgeva il compito di convertire documenti dal formato Office Open XML a RASH.
Dopo averne studiato il codice ho provveduto ad adattarlo per la conversione verso il formato ADF.
Riguardo la parte XSLT di ADF2DOCX, ho sviluppato il codice da zero in quanto in letteratura non ho trovato materiale su cui potessi basarmi.
Ho deciso di utilizzare il linguaggio Java [15] per la parte dedicata ad applicare i fogli di stile ai documenti Office Open XML, e ADF, in quanto era già presente nel framework di RASH il codice sorgente[8] di un programma in grado di convertire documenti dal formato OpenDocument a RASH.

8 https://github.com/essepuntato/rash/blob/master/sources/odt2rash/src/main/java/it/unibo/cs/rash/odt2rash/ODT2RASH.java

5.1.1 Processo di sviluppo

Avendo svolto da solo lo sviluppo dei convertitori, ho adottato un processo di sviluppo del tipo: "*edit-compile-debug*", come mostrato nella seguente figura:

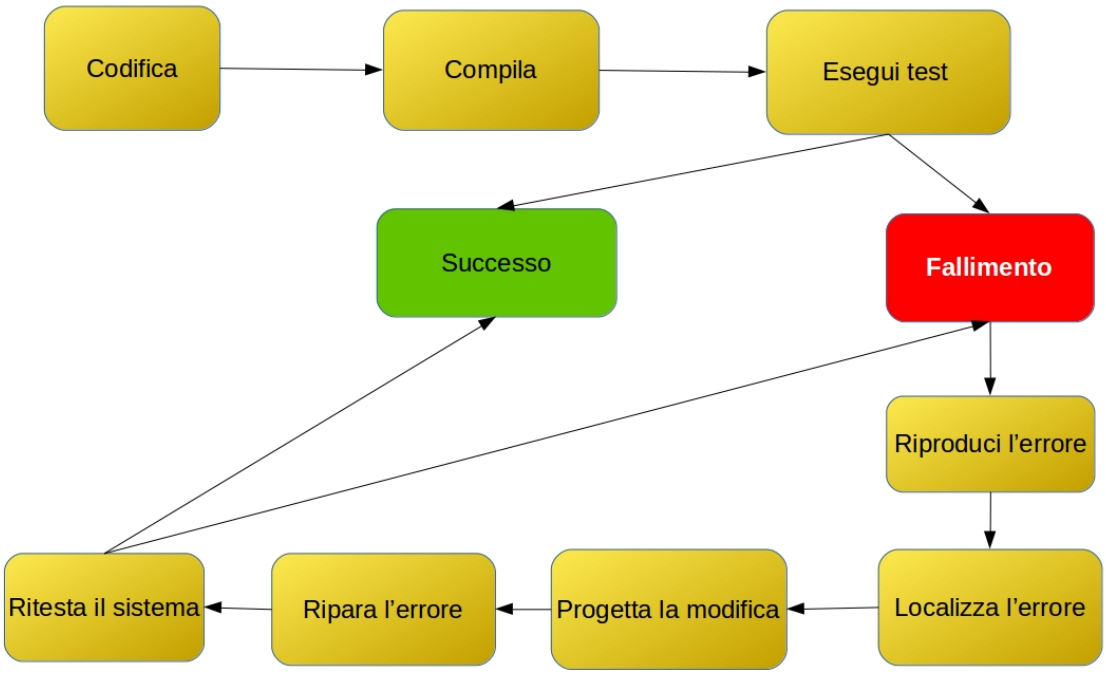


Figura 13: Processo di sviluppo dei convertitori ADF2DOCX e DOCX2ADF

I test che ho eseguito per i software DOCX2ADF e ADF2DOCX sono stati essenzialmente l'ispezione dei risultati prodotti.
Per il software DOCX2ADF ho esaminato visivamente il documento ADF prodotto, mentre per ADF2DOCX mi assicuravo che il documento Office Open XML, risultato della conversione, venisse correttamente visualizzato da MS Word.
Un tipico test che ho eseguito è stato quello di convertire nel formato ADF i documenti in formato Office Open XML originali dell'Azienda ALSTOM, ed in seguito riconvertire i documenti ADF nel formato Office Open XML, analizzandone le differenze.
La figura seguente illustra il test precedente.

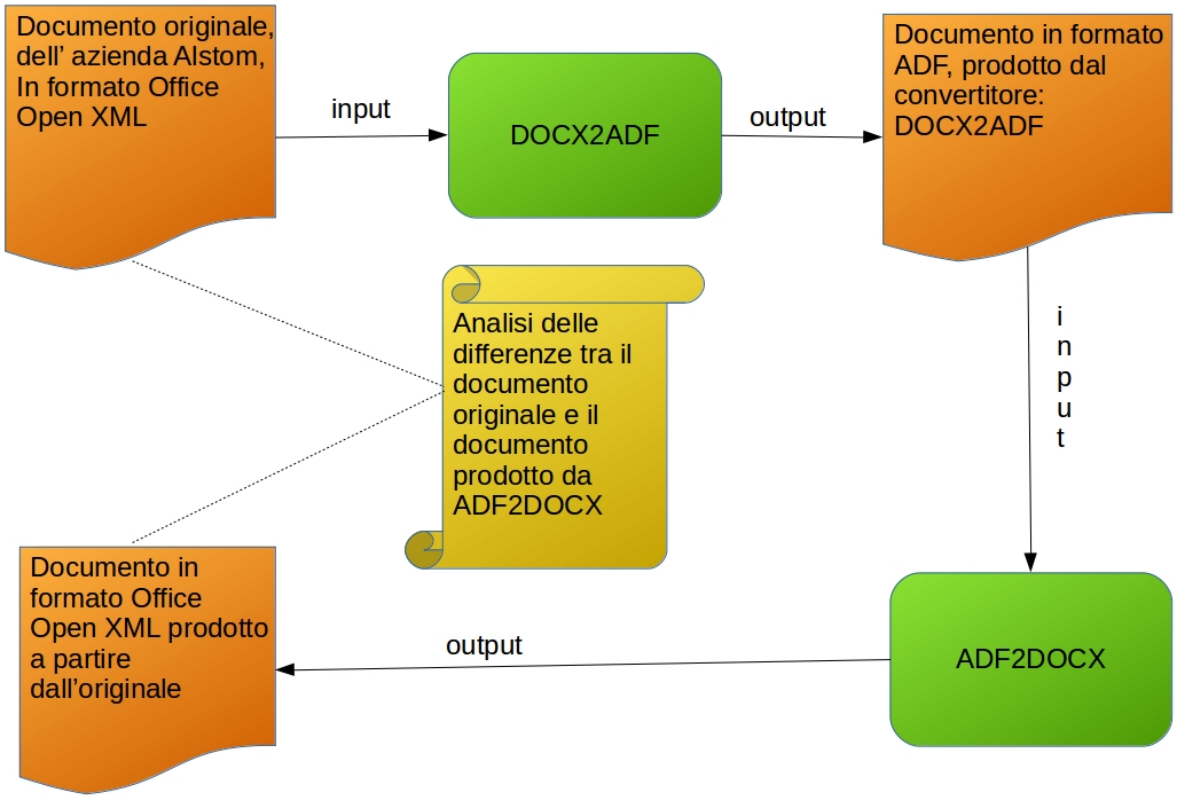


Figura 14: Tipico test eseguito per i convertitori DOCX2ADF e ADF2DOCX

Nel seguito mostro i risultati della conversione che riguardano la copertina che è anche la parte più critica in quanto contiene molte figure posizionate non in linea con il testo, le quali, come spiegherò nel prossimo paragrafo non vengono correttamente gestite.
La Figura 15 mostra la copertina così come la si trova nei documenti tecnici dell'azienda ALSTOM, mentre la Figura 16 mostra la copertina tradotta nel formato ADF, infine nella Figura 17 viene mostrata la copertina convertita nel formato Office Open XML.

Figura 15: Copertina originale dei documenti ALSTOM nel formato Office Open XML

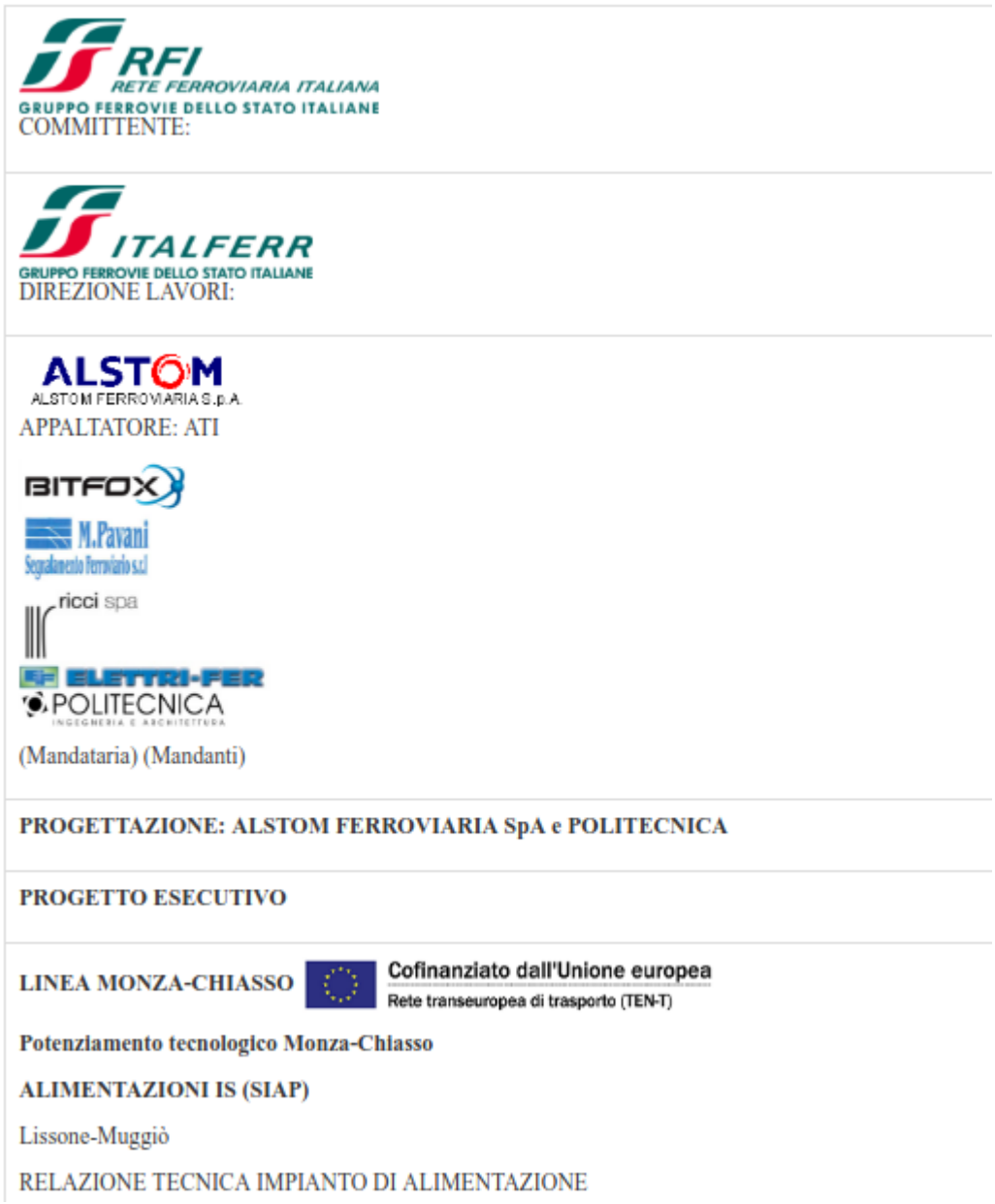


Figura 16: Copertina tradotta nel formato ADF da DOCX2ADF

Figura 17: Copertina nel formato Office Open XML prodotta da ADF2DOCX.

5.2 Limiti dei convertitori

Di seguito elenco gli elementi che non vengono gestiti nel processo di conversione da Office Open XML ad ADF ad opera di DOCX2ADF; e viceversa da ADF a Office Open XML mediante ADF2DOCX.

5.2.1 Elementi non gestiti da DOCX2ADF

- **Posizionamento delle immagini**: Le immagini in un file Office Open XML possono essere posizionate in qualunque punto del documento o seguire il normale flusso di testo. Nel momento in cui viene effettuata la conversione verso il formato ADF vengono correttamente convertite solo le immagini che seguono il normale flusso di testo altrimenti verranno inserite nel momento in un cui sono fisicamente posizionate nel file `document.xml`.
- **Formule matematiche**: le formule matematiche in questa prima versione del convertitore non vengono gestite. Anche ADF non prevede, nella versione corrente, un modo per inserire formule matematiche.
- **Note a piè di pagina**
- **Stili:** tutti gli stili che si applicano al documento che non siano testo in grassetto, corsivo, sottolineato oppure inserito come apice o pedice vengono persi.
- **Intestazioni o piè di pagina:**

5.2.2 Elementi non gestiti da ADF2DOCX

- **Dimensione immagini e posizionamento**: Tutte le immagini convertite tramite ADF2DOCX sono posizionate seguendo il normale flusso di testo e una dimensione fissa.
- **Note a piè di pagina**
- **Stili**: Come nel caso di DOCX2ADF gli stili che non siano grassetto, corsivo, sottolineato oppure inserito come apice o pedice vengono persi. Tutti i documenti prodotti da il convertitore DOCX2ADF hanno uno stile predefinito.

Capitolo 6

Conclusioni

I convertitori presentati in questa tesi, non realizzano perfettamente la conversione dal formato Office Open XML ad ADF e viceversa, in quanto come ho avuto modo di dimostrare nelle precedenti pagine, i due formati presentano grosse differenze. Si è deciso di sviluppare nuovi software in quanto quelli presenti sul mercato sono troppo generali e non effettuano la conversione correttamente da e per il formato ADF. Le sottosezioni che in Office Open XML vengono specificate mediante titoli di livello maggiore di uno, se tradotti con i convertitori presentati nei paragrafo 2.3 verrebbero resi in ADF tramite gli elementi non permessi `<h2>` … `<h6>`. L'implementazione di DOCX2ADF e ADF2DOCX può in futuro essere portata avanti man mano che nuove funzionalità vengono aggiunte ad ADF come il supporto alle formule matematiche. Un' importante funzionalità che può essere aggiunta ad entrambi i convertitori è il posizionamento delle immagini, in questa prima versione del convertitore le immagini vengono posizionate all'interno del flusso di testo senza avere una posizione fissa. Per quanto riguarda il convertitore ADF2DOCX può essere aggiunta è la possibilità di recuperare dal documento ADF le informazioni che riguardano lo stile in modo da riportarle nel documento Office Open XML. La componente Java di entrambi i convertitori, che si occupa principalmente di applicare i fogli di stile ai documenti da convertire, oltre che ad effettuare alcune semplici operazioni sui file, andrebbe riscritta in Javascript. La scelta di cambiare linguaggio è dettata in gran parte al principale utilizzo dei convertitori che viene effettuato attraverso un'applicazione web mediante la piattaforma Node.js.

Bibliografia

1: ECMA-376, Part 1 - Fundamentals And Markup Language Reference, 2016

2: Fabio Vitali, An HTML Vocabulary for ALSTOM documents, 2017

3: Peroni, S., Osborne, F., Di Iorio, A., Nuzzolese, A. G., Poggi, F., Vitali, F., Motta, E., Research Articles in Simplified HTML: a Web-first format for HTML-based scholarly articles, 2016

4: Richard Cyganiak, Markus Lanthaler, David Wood, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, 2014, https://www.w3.org/TR/rdf11-concepts/

5: Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, Niklas Lindström, JSON-LD 1.0 - A JSON-based Serialization for Linked Data, W3C Recommendation, 2014, https://www.w3.org/TR/json-ld/

6: Fabien Gandon, Guus Schreiber, RDF 1.1 XML Syntax W3C Recommendation, 2014, https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

7: David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, Gavin Carothers, RDF 1.1 Turtle, Terse RDF Triple Language, W3C Recommendation, 2014, https://www.w3.org/TR/turtle/

8: Ben Adida, Mark Birbeck, Shane McCarron, Ivan Herman, RDFa Core 1.1 - Third Edition, Syntax and processing rules for embedding RDF through attributes, , 2015, https://www.w3.org/TR/rdfa-syntax/

9: Markus Gylling, Matt Garrish, Tzviya Siegman, Shane McCarron, Digital Publishing WAI-ARIA Module 1.0, W3C Candidate Recommendation , 2016, https://www.w3.org/TR/2016/CR-dpub-aria-1.0-20161215/

10: James Craig, Joanmarie Diggs, Michael Cooper, Shane McCarron, Accessible Rich Internet Applications (WAI-ARIA) 1.1, W3C Candidate Recommendation, 2016, https://www.w3.org/TR/wai-aria-1.1/

11: Erika Doyle Navara, Robin Berjon, Silvia Pfeiffer, Steve Faulkner, Ian Hickson, Travis Leithead, Theresa O'Connor, HTML5, W3C Recommendation, 2014, https://www.w3.org/TR/html5/

12: Alberto Nicoletti, Conversione di documenti DOCX in formato RASH, 2016

13: Michael Kay, XSL Transformations (XSLT) Version 2.0, W3C Recommendation, 23 January 2007

14: Michael Kay, XSLT 2.0 e XPath 2.0, programmer's Reference, 2008

15: K. Arnold, J. Gosling, D. Holmes, Il Linguaggio Java 4/Ed., 2006