Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

L'(IN)ACCESSIBILITÀ DEGLI ARTICOLI SCIENTIFICI SUL WEB E L'USO DI RASH E EPUB

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Vincenzo Rubano

Sessione 1
Anno Accademico 2016-2017

È possibile leggere questa tesi in formato RASH
all'indirizzo

http://www.titengodocchio.it/tesi-triennale

A chi ancora oggi crede che un cieco non possa studiare informatica all'università.

Introduzione

L'accessibilità dei contenuti presenti sul Web è una tematica oramai molto discussa dalla letteratura. Il documento *Understanding WCAG 2.0* [Wor07a] illustra i quattro principi attorno a cui ruotano le *Web Content Accessibility Guidelines (WCAG) 2.0*, le linee guida dettate dal *W3C* a proposito dell'accessibilità delle pagine e delle applicazioni Web. Tali principi definiscono le fondamenta per garantire che chiunque, comprese le persone con disabilità, possa fruire di tali contenuti. In particolare, secondo le linee guida ogni utente deve poter avere contenuti:

percepibili Le informazioni e gli elementi delle interfacce grafiche devono essere presentabili all'utente in modi che egli possa percepire. Ciò significa che gli utenti devono poter percepire le informazioni presentate: esse non possono essere invisibili a tutti i loro sensi.

Utilizzabili Gli elementi delle interfacce grafiche e di navigazione dei documenti devono essere utilizzabili. Ciò significa che l'utente deve essere in grado di utilizzare l'interfaccia: utilizzarla non può richiedere esclusivamente azioni che l'utente non può eseguire.

Comprensibili Le informazioni e le modalità d'uso dell'interfaccia grafica devono essere comprensibili. Ciò significa che l'utente deve essere in grado di comprendere le informazioni così come le modalità di utilizzo dell'interfaccia grafica: i contenuti e le modalità di utilizzo non possono essere al di là della comprensione dell'utente.

Robusti I contenuti devono essere sufficientemente robusti da poter essere interpretati con affidabilità da parte di diversi user agent comprese le tecnologie assistive. Ciò significa che l'utente deve poter accedere ai contenuti via via che le tecnologie progrediscono: con l'evoluzione delle tecnologie e degli user agent i contenuti accessibili devono continuare ad essere tali.

Se un contenuto non rispetta anche uno solo di questi principi gli utenti con disabilità non saranno in grado di fruirne attraverso le tecnologie assistive.

Se negli ultimi anni vi sono dunque stati importanti progressi sul fronte dell'accessibilità dei contenuti Web in generale, non si può dire altrettanto in merito all'accessibilità degli articoli scientifici. Sebbene siano state condotte molteplici ricerche e siano emerse varie tecnologie per migliorare l'accessibilità di questo tipo di contenuti, infatti, essi restano per la maggior parte inaccessibili.

Le cause del problema sono da ricercare in molteplici fattori. I workflow utilizzati per la pubblicazione degli articoli scientifici, per esempio, possono introdurre problemi di accessibilità nelle varie fasi del processo. A volte, poi, gli autori non sono a conoscenza delle problematiche di questo tipo e delle modalità per poterle risolvere all'interno dei propri lavori. Non irrilevante, infine, è l'accettazione del formato `PDF` come standard "de facto" per la pubblicazione e la diffusione degli articoli scientifici sul Web.

Questo formato, infatti, fin dalle sue origini è stato pensato con una grande attenzione alla rappresentazione grafica dei contenuti piuttosto che al loro valore semantico. In un contesto simile diventa molto complicato ottenere una rappresentazione dei documenti che possa soddisfare i principi con cui garantire l'accessibilità dei contenuti. Il formato `PDF` è stato pubblicato per la prima volta dalla società statunitense *Adobe Systems* nel come formato proprietario; visto il suo successo è stato poi pubblicato come standard, seppur con una specifica priva di alcune funzionalità del formato, dalla *International Organization for Standardization (ISO)* nel . Allo stesso anno risale

la pubblicazione dello standard *PDF/UA*, il primo appositamente pensato per garantire l'accessibilità dei documenti in questo formato.

Le caratteristiche del formato `PDF`, la sua storia e la separazione degli standard che ne descrivono la specifica ed il supporto dell'accessibilità hanno fatto sì che molte implementazioni dello stesso non tengano conto degli sviluppi più recenti in merito all'accessibilità di questo tipo di documenti. Rendere un documento conforme allo standard `PDF/UA`, inoltre, non è un'operazione semplice e molti degli strumenti utilizzati nei workflow per la creazione degli articoli scientifici, LaTeX in primis, non lo supportano affatto, lo supportano parzialmente o non ne abilitano il supporto come impostazione predefinita.

Dopo aver fornito un esame approfondito dello scenario e del contesto, in questo lavoro di tesi viene proposta una alternativa basata sull'utilizzo del formato *Research Articles in Simplified HTML (RASH)* come mezzo per la produzione di articoli scientifici ed il formato `EPUB` come standard per la distribuzione dei documenti in tale formato. La combinazione di tali strumenti ha permesso di proporre un workflow per la produzione degli articoli scientifici che può competere con quelli più tradizionali e presenta molteplici vantaggi che ne giustificherebbero l'eventuale adozione, tra cui la possibile fruizione da parte di utenti con disabilità.

Il capitolo 1 fornisce una panoramica sul problema sintetizzandone le cause e gli effetti e illustra le soluzioni alternative emerse nel corso del tempo per provare a superarlo, evidenziandone vantaggi, svantaggi ed eventuali limiti. Sono dapprima analizzati brevemente i workflow principali che vengono utilizzati per produrre e pubblicare gli articoli scientifici e gli strumenti impiegati, per poi analizzare l'accessibilità di ciascuno di essi e gli aspetti che possano influenzarla. Dopo aver isolato nel formato `PDF` una causa di problemi comuni a tutti i workflow, sono illustrati in dettaglio la storia dell'accessibilità in questo formato e lo standard `PDF/UA` descrivendone brevemente i concetti fondamentali e le difficoltà di applicazione. Sono poi identificate le alternative più sostenibili per produrre articoli scientifici accessibili facendo in modo che implementarne l'accessibilità non abbia un impatto negativo sulla

produttività del workflow, ovvero non richieda sforzi aggiuntivi (o comporti effort minimi) da parte dell'autore per ottenere il risultato. Tali alternative sono identificate nei formati `HTML-Based` ed, in particolare, tra questi viene scelto l'utilizzo del formato `RASH` poiché presenta molteplici vantaggi dovuti al modo in cui è stato definito. Poiché gli articoli in formato `RASH` non possono essere distribuiti agevolmente all'interno di un unico file insieme alle risorse di cui necessitano per essere visualizzati correttamente, si propone l'uso del formato `EPUB` per colmare tale lacuna discutendo le ragioni di questa scelta.

Nel capitolo 2 viene invece introdotto lo strumento `rash2epub`, una utility che consente di convertire un documento in formato `RASH` nel suo equivalente in formato `EPUB` sviluppata durante questo lavoro di tesi. Dopo aver descritto il formato per la distribuzione di ebook `EPUB` ed averne brevemente illustrato la storia e le caratteristiche principali, è analizzato come esso permetta di creare pubblicazioni (anche a carattere scientifico) estremamente dinamiche ed accessibili. Viene poi fornita una panoramica sullo strumento `rash2epub` e sul processo di conversione evidenziandone i punti di forza e sottolineandone le caratteristiche significative dal punto di vista dell'accessibilità. Sono descritte infine le modalità di utilizzo dello strumento dal punto di vista dell'utente.

Nel capitolo 3 viene analizzata più in dettaglio l'implementazione dello strumento e le modalità con cui avviene il processo di conversione, nonché i principali problemi da esso derivanti e le soluzioni adottate per superarli. In particolare viene descritta l'architettura di `rash2epub` e successivamente sono discussi alcuni importanti problemi affrontati durante il processo di conversione quali:

- la gestione dei possibili errori;

- le modalità con cui vengono incluse le risorse esterne necessarie alla visualizzazione del documento, fornite dal framework `RASH` e non;

- la compressione del risultato in modo da ottenere un file conforme allo standard EPUB

Viene poi discusso approfonditamente uno dei più importanti problemi della conversione di un documento RASH in formato EPUB, ovvero la gestione delle formule matematiche in esso contenute. Il framework RASH, infatti, si avvale della libreria MathJax per consentire all'utente di utilizzare molteplici modalità di input per le formule, nonché per assicurare che esse vengano visualizzate correttamente indipendentemente dal dispositivo usato per leggere il documento. Integrare MathJax all'interno dei file EPUB generati da rash2epub non è un problema così banale come potrebbe sembrare: la libreria, infatti, ha una dimensione di 62 3MB ed includerla in ogni file EPUB generato da rash2epub creerebbe ebook di dimensioni troppo elevate. È stato dunque necessario individuare una strategia che è descritta dettagliatamente per includere la libreria negli ebook generati minimizzandone il footprint.

Nel capitolo 4 sono descritti i test eseguiti per assicurare il corretto funzionamento dello strumento rash2epub e se ne discuteranno le prestazioni. Viene, inoltre, ipotizzato un possibile workflow per la scrittura di articoli scientifici in formato RASH, che preveda l'uso di tale strumento, il quale è poi comparato a quelli descritti nel capitolo 1, evidenziandone i punti di forza rispetto a questi ultimi. Sono approfonditi in particolare i vantaggi del workflow ipotizzato dal punto di vista dell'accessibilità.

Nel capitolo 5 viene infine dato spazio ad alcune riflessioni a proposito del problema e del workflow ipotizzato; sono poi descritti alcuni possibili sviluppi futuri che potrebbero migliorare lo strumento rash2epub e la sua integrazione all'interno del framework RASH.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

L'(in)accessibilità degli articoli scientifici sul Web

Il tema dell'(in)accessibilità degli articoli scientifici presenti sul Web è molto ampio e si presta ad essere affrontato da vari punti di vista. In questo capitolo si cercherà di fornire una panoramica sul problema sintetizzandone le cause e gli effetti e si tenterà di illustrare le soluzioni alternative emerse nel corso del tempo per provare a superarlo, evidenziandone vantaggi, svantaggi ed eventuali limiti.

1.1 Workflow principali

Per comprendere quali possano essere le cause dei problemi di accessibilità degli articoli scientifici presenti sul Web occorre innanzitutto analizzare brevemente i workflow principali che vengono utilizzati per produrli e pubblicarli e gli strumenti impiegati.

1.1.1 Gli strumenti per scrivere documenti...

Una categoria di programmi usati molto spesso per la scrittura di documenti è costituita dai cosiddetti editor WYSIWYG (What You See Is What You Get), ovvero editor in cui è possibile vedere in tempo reale gli effetti di

tutte le modifiche applicate al documento. Tra gli editor wysiwyg più popolari si trovano Microsoft Word, Google Docs e Writer (parte del pacchetto LibreOffice). Questi editor si caratterizzano per la facilità d'uso e l'immediatezza con cui permettono di comprendere esattamente come verrà mostrato il documento.

Altre volte, invece, per la scrittura di documenti si utilizzano i formati `HTML` [Wor16] e `XHTML` [Wor02]. Questi formati sono nati per la redazione di contenuti da pubblicare su Internet (ad esempio pagine Web) e, quindi, si rivelano particolarmente adatti per questo scopo. Assieme ad altre tecnologie di supporto (come `CSS` [Wor15] e `JavaScript`) permettono di creare contenuti estremamente dinamici e con layout grafici molto sofisticati. Per redigere documenti con questi formati è necessario scrivere il documento seguendo il linguaggio scelto e sarà possibile visualizzarne il rendering grafico soltanto in una fase successiva. Per immettere formule matematiche nei documenti scritti in questi formati è possibile utilizzare il linguaggio `MathML` [Wor14c], appositamente pensato per rappresentare le notazioni matematiche in modo strutturato lasciando al browser il compito di fornirne il rendering grafico. Alcuni utenti potrebbero trovare la sintassi di `HTML` e `XHTML` "pedante" e troppo verbosa; le ragioni di questa complessità vanno ricercate nella storia delle origini di questi linguaggi.

Un'altro sistema molto popolare usato per la scrittura di documenti è LaTeX, il quale risulta estremamente diffuso negli ambienti accademici e più in generale nei contesti in cui è necessario ottenere la maggior qualità di stampa possibile. Come descritto in [Lam94], LaTeX prevede la scrittura di un documento di testo semplice seguendo un apposito linguaggio; tale documento viene poi interpretato da alcuni strumenti e, al termine del cosiddetto processo di *typesetting*, si ottiene un file in formato `DVI`, `PS` o, più spesso, `PDF`. Oltre alla elevata qualità di stampa dei file generati e alla grande flessibilità e potenza del linguaggio, un grande punto di forza di LaTeX risiede nella possibilità di inserire agevolmente formule matematiche utilizzando una sintassi testuale lineare; durante il processo di typesetting le formule vengono con-

vertite automaticamente in modo da essere visualizzate con il layout grafico bidimensionale tipico del loro rendering. La frazione $\frac{(x-1)^2}{(x+1)^3}$, per esempio, può essere scritta come `$\frac{(x-1)^2}{(x+1)^3}$` e sarà convertita nella sua rappresentazione grafica bidimensionale durante il processo di typesetting.

1.1.2 . . . e gli articoli scientifici

Se per scrivere un documento qualunque è possibile scegliere cosa utilizzare tra una vasta serie di strumenti, nel caso in cui si debba scrivere articoli scientifici è necessario che la scelta sia più oculata. Lo strumento scelto, infatti, deve offrire funzionalità aggiuntive (gestione avanzata di bibliografia, citazioni, formule matematiche e riferimenti incrociati, supporto per la gestione di float come tabelle e immagini, gestione semplice di layout multicolonna, etc.) rispetto a quelle necessarie per scrivere documenti generici. Inoltre, se si vuole che il proprio lavoro venga pubblicato, è necessario che lo strumento scelto supporti l'esportazione del documento in formato `PDF`: la maggior parte dei periodici, delle riviste, delle conferenze e dei convegni accettano infatti l'invio di articoli soltanto in questo formato.

Questi ulteriori vincoli rendono i formati `HTML` e `XHTML` meno adatti per la scrittura di articoli scientifici rispetto agli altri strumenti visti finora. Sebbene gli editor WYSIWYG ultimamente si siano evoluti molto, la maggior parte di essi ancora non offre tutti gli strumenti indispensabili per supportare la redazione degli articoli scientifici; non è una sorpresa, dunque, che lo strumento più usato per scrivere tali articoli sia ancora oggi LaTeX: oltre ad essere disponibile da moltissimi anni su varie piattaforme, la comunità dei suoi utilizzatori ha sviluppato innumerevoli pacchetti che ne estendono le funzionalità e lo rendono adatto agli scopi più disparati (si vedano ad esempio [MGB+04, lat]).

1.2 L'accessibilità nei workflow per la produzione di articoli scientifici

Dopo la breve panoramica sui principali workflow utilizzati per la creazione di articoli scientifici presentata nel paragrafo 1.1, si cercherà ora di analizzarne l'accessibilità, illustrando per ciascuno di essi gli aspetti positivi e negativi che la influenzano.

1.2.1 L'accessibilità nei workflow che usano `HTML` e `XHTML`

L'accessibilità dei documenti in formato `HTML` e `XHTML` è una tematica oramai molto nota e discussa nei minimi dettagli dalla letteratura. Al riguardo esistono sia riferimenti normativi in diverse nazioni (ad esempio [Par04] e [Ame98]) che vari standard internazionali. Il documento [Wor08b] fissa le linee guida che tutti i contenuti Web devono seguire per essere considerati accessibili ed i cosiddetti *success criteria*, ovvero criteri utilizzabili per testare la conformità dei documenti alle linee guida. Tali criteri sono organizzati in tre diversi livelli (A, AA e AAA) e, a seconda dei criteri che il documento soddisfa, la sua conformità alle [Wor08b] viene classificata in uno dei tre livelli.

Il documento [Wor07a], invece, illustra dettagliatamente le ragioni dietro ogni linea guida ed elenca le tecniche disponibili per rendere i documenti conformi alle stesse; tali tecniche sono poi illustrate in modo approfondito, anche attraverso esempi pratici, nel documento [Wor08a].

Sebbene come visto nel paragrafo 1.1 sia possibile inserire formule matematiche in questo tipo di documenti utilizzando `MathML`, purtroppo la storia di questo linguaggio è molto complessa [Kra13]. Il risultato di queste vicissitudini è che nella maggior parte dei browser la sua implementazione risulta parziale, incompleta e assortita di bug quando non del tutto assente (come per esempio avviene nel motore `chromium` usato dal browser Google Chrome). Unito al fatto che la sintassi di `MathML` risulta eccessivamente verbosa, ciò a portato gli autori ad inserire molto spesso le formule matematiche nei docu-

menti HTML e XHTML sotto forma di immagini [ACI+14] generate attraverso programmi esterni. Questo workaround crea diversi problemi sia dal punto di vista dell'accessibilità, poiché il contenuto delle immagini non può essere letto dagli screen reader, sia dal punto di vista dell'usabilità, poiché risulta difficile modificare le dimensioni della formula per renderla più leggibile a seconda del dispositivo utilizzato per visualizzare il documento. Recentemente, tuttavia, sta emergendo come "best practice" quella di usare MathJax [Cer12]per l'inserimento di formule matematiche nei documenti di questo tipo: tale framework agisce sia come polifill, permettendo di visualizzare correttamente le formule immesse su qualunque browser indipendentemente dal supporto che esso offre per il linguaggio MathML, sia come helper, poiché consente di inserire le formule nei documenti utilizzando le sintassi AsciiMath o LaTeX che risultano molto più agevoli per la scrittura e la manipolazione delle formule. Durante la lettura di un documento in cui le formule matematiche siano espresse nei formati accessibili visti finora, l'utente potrà avvalersi di MathPlayer [Soi15], un plugin che consente di gestire in modo ottimale tali formule attraverso le tecnologie assistive in vari browser.

Occorre infine citare che in documenti in formato HTML e XHTML è possibile utilizzare evoGraphs [Sha16], un plugin che consente di generare grafici il cui contenuto è accessibile senza sforzi aggiuntivi da parte dell'autore, ed un sistema [SLW15] che consente di rappresentare la struttura delle molecole in modo tale che gli utenti di screen reader possano esplorarle e comprenderne la struttura.

1.2.2 L'accessibilità nei workflow basati su editor WYSIWYG

occorre fare un discorso leggermente più complesso per analizzare l'accessibilità dei workflow che prevedono l'uso di editor WYSIWYG. Sebbene molte volte i documenti creati con un editor possono essere letti e modificati con editor diversi, spesso vi sono differenze nell'implementazione di alcuni dettagli della specifica del formato che portano ad incompatibilità tra un editor e l'altro; tali incompatibilità possono alterare la struttura o la visualizzazione

del documento e, pertanto, anche se 2 editor usano un formato comune è sempre consigliabile usare lo stesso editor quando possibile. A questo punto, dunque, l'accessibilità dell'editor usato per creare il documento diventa condizione necessaria (ma non sufficiente) per l'accessibilità dei documenti creati con esso. Volendo spendere qualche parola sui 3 editor WYSIWYG citati nel paragrafo 1.1, l'esperienza di chi scrive suggerisce che:

- Microsoft Word fornisce un ottimo grado di accessibilità , anche se a volte è necessario attendere gli aggiornamenti di screen reader di terze parti che lo supportino prima di poterlo usare a pieno;

- Google Docs fornisce un buon supporto di accessibilità ([Goo], anche se l'esperienza d'uso tramite screen reader non è ottimale a causa delle limitazioni dovute alla natura di web-app del prodotto;

- Writer presenta molteplici problemi di accessibilità (si veda [Fre]) che lo rendono di fatto inutilizzabile con uno screen reader in vari contesti.

1.2.3 L'accessibilità nei workflow che usano LaTeX

Per parlare in modo esaustivo dell'accessibilità dei workflow che prevedono l'uso di documenti LaTeX, invece, occorre affrontare il problema da due punti di vista distinti: l'accessibilità del documento LaTeX vero e proprio (detto anche *sorgente LaTeX*) e l'accessibilità del risultato ottenuto attraverso la compilazione del sorgente LaTeX. Un documento LaTeX, per come è strutturato il linguaggio stesso ([Lam94]), è un documento di testo semplice in cui si trovano sia il contenuto che una serie di comandi per la sua formattazione; lo stesso principio vale anche per le formule matematiche che, pertanto, sono descritte attraverso una sintassi testuale lineare anche quando la loro rappresentazione grafica non lo sia (per esempio nel caso di frazioni e matrici). Nonostante ciò, la scelta dell'applicazione con cui leggere ed editare i sorgenti LaTeX resta un fattore rilevante: in certi casi, infatti, la sintassi del linguaggio può essere difficile da comprendere utilizzando uno screen reader

ed un display Braille e, perciò, alcuni strumenti offrono supporti più o meno sofisticati per aiutare l'utente ad aggirare il problema. Recentemente, per esempio, è stato presentato [Sor16], un'estensione all'ambiente Emacspeak [Ram97] che agevola l'editing e la lettura dei documenti LaTeX utilizzando le varie "dimensioni" del feedback audio (intonazione, enfasi, suoni, etc) e generando dinamicamente la pronuncia delle formule matematiche secondo le regole di lettura dei loro rendering [CKS16]. Un'altra soluzione simile nel principio e nel funzionamento è data da `Latex-Access`, la quale, però, funziona all'interno di qualunque editor di testo per Windows e solo con gli screen reader `JAWS` e `NVDA` che come mostrato nella figura 1.1 sono i 2 più usati.

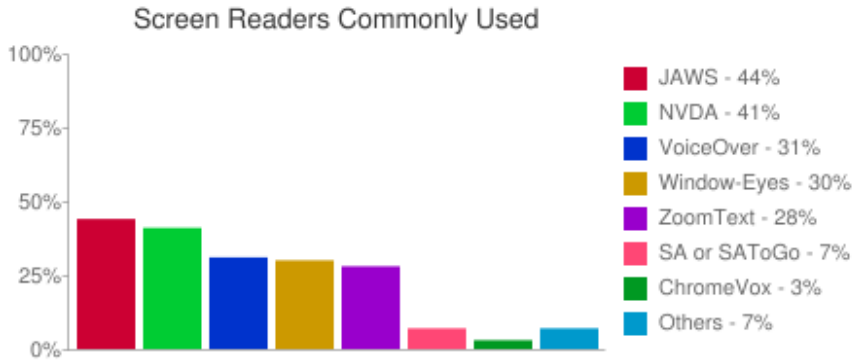


Figura 1.1: Grafico che mostra gli screen reader usati più comunemente. I dati rappresentati sono riportati nella tabella 1.1

Fonte: `http://webaim.org/projects/screenreadersurvey6/`

Nella stragrande maggioranza dei casi, però, purtroppo i sorgenti LaTeX di un documento non sono disponibili; viene invece distribuito il risultato ottenuto dalla compilazione di tali sorgenti. Il processo di compilazione restituisce in output un documento in formato `PDF` oppure un risultato intermedio costituito dal documento in formato `DVI`. Poiché tale file intermedio risulta utile per particolari tipi di processi tipografici finalizzati alla stampa ed in generale non è molto usato come formato per la distribuzione degli articoli scientifici, esso non sarà analizzato in dettaglio. Il processo di compilazione

di un sorgente LaTeX in formato PDF, detto anche *typesetting*, consiste nel produrre un file PDF contenente il rendering del sorgente LaTeX, ossia il documento esattamente come debba essere visualizzato e/o stampato. Ciò che viene compilato, però, non è il sorgente LaTeX vero e proprio; LaTeX, infatti, è definito come un insieme di macro in linguaggio TeX. Al contrario di LaTeX, TeX ha caratteristiche fortemente incentrate sulla formattazione e sul layout piuttosto che sulla semantica. Ciò influenza la generazione dell'output, poiché viene data una forte importanza all'aspetto visivo a volte "trascurando" le implicazioni semantiche delle scelte implementative compiute. Questo principio vale ovviamente anche per le formule matematiche le quali, pur essendo perfettamente fruibili nel sorgente LaTeX, diventano ingestibili con uno screen reader nel documento ottenuto compilando tale sorgente. Come si vedrà meglio nel paragrafo 1.3, originariamente il formato PDF non prevedeva alcun supporto specifico per l'accessibilità; il processo di compilazione dei sorgenti LaTeX restituisce ancora oggi documenti che non tengono conto dei più recenti aggiornamenti della specifica del formato in tal senso.

Occorre citare, infine, un approccio molto differente per migliorare l'accessibilità dei sorgenti LaTeX sperimentato da diversi anni; tale approccio consiste nel tentare di convertire i sorgenti in documenti HTML e/o XHTMLi quali possono essere letti più agevolmente tramite gli screen reader e i display Braille. Curiosamente, gli strumenti che implementano questo approccio non sono stati pensati con l'accessibilità come scopo principale, ma piuttosto per estrarre documenti più strutturati rispetto ai tradizionali file PDF che si ottengono compilando i sorgenti LaTeX. I principali strumenti che consentono di utilizzare questo approccio sono [DM99], [Mil16] e [Gur04]. L'esperienza di chi scrive suggerisce che l'ultimo strumento citato è il più completo e robusto e supporta un maggior numero di comandi LaTeX; tuttavia il futuro del progetto sembra quantomai incerto poiché dopo la morte del suo ideatore Eitan Gurari (avvenuta nel) non sono stati rilasciati aggiornamenti significativi dello strumento.

1.2.4 PDF: una causa del problema

Stando a quanto visto finora, dunque, ci si potrebbe chiedere come sia possibile che da workflow con i supporti di accessibilità descritti si generino tutti i problemi di accessibilità riscontrabili negli articoli scientifici. La causa principale del problema è da ricercare nel formato PDF, il quale è diventato lo standard di fatto per l'invio e la pubblicazione di tali articoli [BZB15]. Il grande successo di questo formato è da ricercarsi in due ragioni:

1. la garanzia di una visualizzazione identica del contenuto indipendentemente dal sistema operativo, dall'applicazione e dal dispositivo utilizzati;

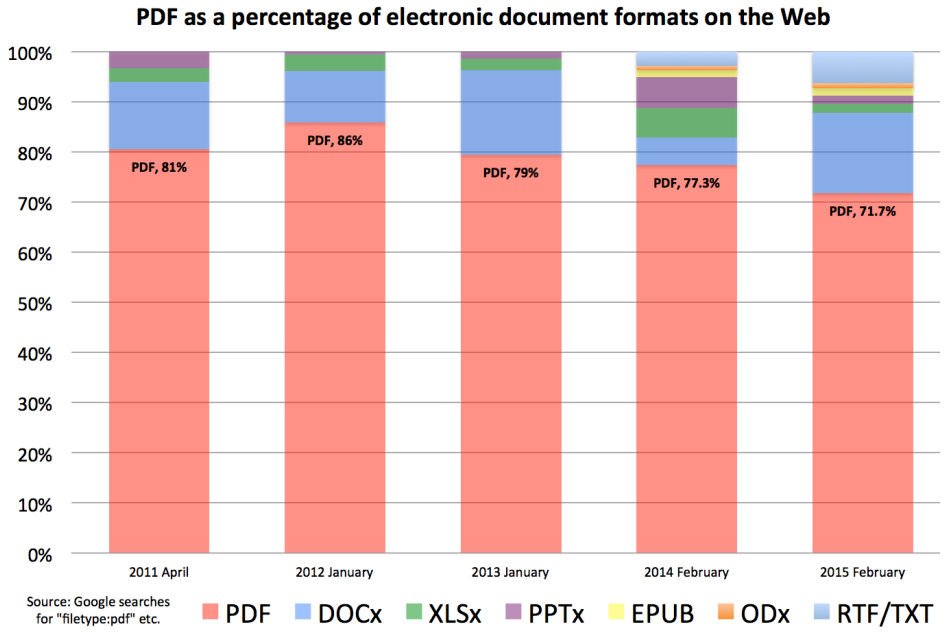
2. la "naturalità" con cui è possibile ottenere documenti in questo formato a partire da sorgenti LaTeX.

Uno studio [LAKM07] ha dimostrato che l'inaccessibilità dei documenti in formato PDF è una delle principali cause di frustrazione quando i non vedenti navigano il Web. Ciò è altrettanto vero nel caso degli articoli scientifici. Spesso essi sono disponibili in questo formato e non sono accessibili per mancanza di conoscenze da parte degli autori, del non ritenerne l'accessibilità una priorità [BZB15] o, come si vedrà meglio nel paragrafo 1.3 semplicemente per ragioni strutturali legate al modo in cui l'accessibilità è stata pensata ed implementata in questo formato.

1.3 L'accessibilità nei documenti in formato PDF

Verrebbe spontaneo chiedersi come sia possibile che il formato PDF, il più diffuso per lo scambio di documenti (vedi 1.2 e lo standard de facto per la pubblicazione degli articoli scientifici [BZB15] possa creare tanti problemi dal punto di vista dell'accessibilità, anche quando i documenti vengono prodotti a partire da workflow che per loro natura genererebbero documenti accessibili.

Per provare a dare una risposta a questa domanda occorre innanzitutto ripercorrere brevemente la storia di questo formato poiché, come sempre ac-



Il grafico mostra la percentuale di ricerche di documenti in formato PDF comparata alla percentuale delle ricerche di documenti in altri formati. Si evidenzia una netta predominanza della percentuale delle prime, con una leggera inflessione intorno al a causa dell'aggiunta da parte dell'autore di altri formati. I dati rappresentati nel grafico sono disponibili nella tabella 2.1.

Fonte: `http://duff-johnson.com/2015/02/12/the-8-most-popular-document-formats-on-the-web-in-2015/`

Figura 1.2: Ricerche di documenti in vari formati

cade, le vicende storiche permettono di comprendere meglio le ragioni che portano a determinate scelte piuttosto che ad altre. Il formato PDF nasce nei primi anni novanta per consentire lo scambio di documenti che possano contenere all'interno dello stesso file testo, immagini ed istruzioni per la formattazione dell'output e viene pubblicato per la prima volta dall'azienda statunitense *Adobe Systems* nel . Nel corso degli anni la specifica del formato subisce diverse evoluzioni che aggiungono varie funzionalità interattive tra cui la gestione di audio e video all'interno del documento, la possibilità di includere moduli compilabili dall'utente e la creazione di contenuti dinamici attraverso l'uso di un'estensione proprietaria di JavaScript. Fin da subito il formato PDF viene adottato nei workflow di pubblicazione più utilizzati dagli editori [Wikb]. In seguito, visto il grande successo del formato, varie componenti della specifica vengono rilasciate come standard dalla International Organizations for Standardization (ISO); il primo standard che ne descriva una specifica completa è [Int14d].

Il formato PDF è caratterizzato da tre componenti principali:

- un sottoinsieme del linguaggio di programmazione per la descrizione di pagine PostScript [1], utilizzato per la descrizione del layout e di tutti gli aspetti grafici del documento;
- un meccanismo per la gestione dei font, per consentire di integrarli e distribuirli nel documento che li usa;
- un meccanismo strutturato per la memorizzazione dei contenuti necessari per soddisfare i punti precedenti ed ogni altra risorsa (ad esempio immagini e testo) del documento.

Fin dalle sue origini, quindi, il formato PDF è stato fortemente votato alla "rappresentazione" grafica delle informazioni più che alla loro "semantica". Questa situazione era incompatibile con l'esigenza sempre più pressante di dover rendere i documenti in questo formato accessibili agli utenti di tecnologie assistive; sebbene i primi tentativi di supportare l'accessibilità di questo

[1] https://en.wikipedia.org/wiki/PostScript

tipo di documenti risalgano al , è stato possibile superare tale incompatibilità (per lo meno dal punto di vista delle specifiche) soltanto con la pubblicazione di uno standard ISO apposito detto PDF/UA [Int14c] che ha aperto nuovi scenari impensabili fino ad allora per l'accessibilità di questo formato [DE12]. Come illustrato in [Oet13] lo standard fissa regole:

- sul come debbano essere fornite informazioni e strutture aggiuntive che consentano alle tecnologie assistive di ottenere informazioni semantiche sul documento;
- sul come le soluzioni per il rendering di documenti PDF debbano esporre tali informazioni alle tecnologie assistive in esecuzione quando il documento viene visualizzato;
- sul come le tecnologie assistive debbano fare uso di tali informazioni semantiche aggiuntive.

Questo standard si prefigge come scopo quello di applicare i principi dettati dalle WCAG 2.0 ai documenti PDF [Drü12]. Più nel dettaglio, per rendere i documenti accessibili con lo standard PDF/UA occorre aggiungere una struttura XML basata su appositi tag: i documenti resi accessibili in questo modo, perciò, sono anche detti *tagged* o *taggati*. La struttura di alcuni tag è molto simile a quella dei tag HTML (per esempio, per contrassegnare un frammento di testo come titolo di livello uno è necessario racchiuderlo nel tag h1), ma ciò non deve trarre in inganno: le similitudini si limitano ai nomi dei tag e al modo in cui le tecnologie assistive debbano gestirli. In questo standard non è previsto alcun meccanismo specifico per rendere accessibili le formule matematiche.

Se lo strumento per rendere i documenti PDF accessibili esiste, dunque, perché ci ritroviamo nella situazione attuale, "circondati" da file PDF prodotti dopo la pubblicazione dello standard PDF/UA che, però, sono inaccessibili [SM14]? In primo luogo lo standard PDF/UA è completamente separato da quello che descrive il formato PDF vero e proprio [Int14d]; ciò significa che

possono esistere implementazioni della specifica del formato PDF che non supportano lo standard PDF/UA. In secondo luogo, sebbene esista uno strumento open source per verificare la conformità dei documenti allo standard PDF/UA [UBR14], l'unico programma che consente di apportare le modifiche necessarie a correggere gli eventuali problemi rilevati è *Adobe Acrobat Reader Pro*, la cui licenza ha un costo significativo.

In secondo luogo va detto che tutti i workflow per la produzione di documenti visti finora supportano male (o non supportano affatto) lo standard PDF/UA. Gli editor WYSIWYG di solito consentono di esportare i documenti sotto forma di PDF taggati, ma l'opzione non è abilitata di default e spesso è nascosta in finestre di configurazione complesse che l'utente potrebbe anche non visualizzare mai; è questo il caso di Microsoft Word e Writer. Altre volte, invece, l'esportazione sotto forma di PDF accessibile è semplicemente non supportata: è questo il caso di LaTeX che nonostante diversi tentativi compiuti nel tempo [Moo09, Cli] ad oggi non offre un'implementazione realmente funzionante dello standard all'interno del processo di compilazione dei sorgenti. Infine, nel caso dei workflow che prevedono l'uso di documenti in formato HTML e XHTML, spesso l'esportazione in formato PDF non è prevista.

1.4 Alternative sostenibili: i formati HTML per documenti

Da quanto visto, dunque, il processo per creare un articolo scientifico accessibile in formato PDF si può riassumere nei seguenti punti:

1. Utilizzare un editor WYSIWYG o LaTeX per scrivere l'articolo.
2. Se si è usato un editor WYSIWYG ed esso lo supporta, individuare l'opzione per esportare il documento come PDF taggato e selezionarla, assicurandosi che essa sia selezionata ogni volta che si esporta un documento.
3. Esportare l'articolo in formato PDF.

4. Controllare l'accessibilità del risultato con strumenti automatici come lo strumento *accessibility checker* integrato in *Adobe Acrobat Reader Pro* o *PAC 2* [UBR14].

5. Correggere i problemi segnalati utilizzando Adobe Acrobat Reader Pro, l'unico software disponibile sul mercato in grado di apportare le modifiche necessarie;

6. Verificare manualmente il risultato e correggere eventuali problemi emersi utilizzando Adobe Acrobat Reader Pro, ancora una volta l'unico strumento in grado di apportare le eventuali modifiche necessarie.

A ciò va aggiunto che la modifica dei tag all'interno di Adobe Acrobat Reader Pro è un'operazione non banale che richiede un'approfondita conoscenza sia della materia che dell'applicazione stessa e che, allo stato attuale, lo standard non prevede ufficialmente alcun modo per rendere accessibili le formule matematiche. Viene spontaneo chiedersi, dunque, se non sia possibile individuare delle alternative più sostenibili per produrre articoli scientifici accessibili magari facendo in modo che implementarne l'accessibilità non abbia un impatto negativo sulla produttività del workflow, ovvero non richieda sforzi aggiuntivi (o comporti effort minimi) da parte dell'autore per ottenere il risultato.

Stando a quanto detto nel paragrafo 1.2 verrebbe spontaneo cercare una soluzione che in qualche modo racchiuda tutti i vantaggi dei formati `HTML` e `XHTML` ma che, nello stesso tempo, permetta di superare i limiti di questi ultimi. Recentemente, per esigenze diverse da quelle di accessibilità, nella comunità scientifica stanno emergendo diversi formati pensati per la scrittura di articoli scientifici che si basano proprio su tecnologie Web quali `HTML`, `CSS`, `XML`, `JavaScript`, ed altre. Si cercherà dunque di analizzare i principali formati emersi finora e si proverà ad individuarne i punti di forza e di debolezza legati all'accessibilità, illustrando come l'uso di uno di questi possa raggiungere l'obiettivo prefissato all'inizio di questo paragrafo.

1.4.1 I formati HTML-based per la scrittura di articoli scientifici

Sebbene non sia un formato propriamente HTML-based, merita una citazione ScholarlyMarkdown [LB15], una versione di Markdown [Gru04] pensata appositamente per la scrittura di articoli scientifici. Rispetto alla sintassi markdown originale essa aggiunge varie funzionalità utili per la scrittura di articoli scientifici, come la gestione delle figure e dei riferimenti; un documento scritto con questa sintassi viene processato da un particolare fork dell'applicazione pandoc [Mac] chiamato *scholdoc*, ottenendo in output il suo equivalente in formato HTML o XHTML.

Scholarly HTML [SMRF$^+$], invece, si pone come un formato intermedio per la distribuzione di articoli scientifici; esso non è definito da una grammatica formale, ma piuttosto da una serie di regole che consentono di specificare un sottoinsieme di tag HTML sufficiente a descrivere i metadati ed il contenuto di un articolo. Un approccio simile viene adottato da PubCSS [Par15] il quale fornisce un numero limitato di fogli di stile CSS per formattare documenti HTML secondo alcuni dei layout più richiesti nelle conferenze in ambito informatico; anche in questo caso, dunque, non esiste una grammatica formale che descriva la sintassi.

Un altro formato HTML-based è HTMLBooks [Ore]. Esso è un formato pensato specificamente per la scrittura di libri (anche a carattere tecnico e scientifico) che poi possano essere sia stampati che pubblicati in forma digitale agevolmente. Viene definito come un sottoinsieme di XHTML 5 ma, al è ancora considerato un *draft* dai suoi stessi ideatori; ne deriva che la documentazione non è dettagliata e la grammatica non è ancora ufficiale. Un ulteriore formato HTML-based piuttosto maturo va sotto lo stesso nome di un formato già visto in precedenza: ScholarlyHtML [sci15]; tale formato consente di scrivere articoli scientifici utilizzando un sottoinsieme piuttosto ampio dei tag del linguaggio HTML e da una serie di annotazioni schema.org [2] utiliz-

[2] http://schema.org

zate per descrivere i metadati e la struttura del documento. Va sottolineata, inoltre, l'esistenza di un gruppo di lavoro all'interno del W3C denominato *ScholarlyHTML* [3] che mira alla creazione di uno standard ufficiale basato sulle tecnologie Web per agevolare lo scambio di articoli scientifici.

`RASH (Research Articles in Simplified HTML)` [PODI+16], infine, è un formato descritto da una grammatica formale e definito come un sottoinsieme molto ristretto (composto da soli 31 elementi) di tag del linguaggio `XHTML 5` [Wor16]. La grammatica di `RASH`, inoltre, consente di inserire nel documento annotazioni `RDFA` [Wor], `Turtle` [Wor14d], `RDF/XML` [Wor14e] e `JSON-LD` [Wor14b].

1.4.2 Perché `RASH`?

A questo punto è logico chiedersi perché è stato scelto di utilizzare il formato `RASH` piuttosto che uno degli altri formati `HTML-Based` per la pubblicazione di articoli scientifici.

Come detto poc'anzi, `RASH` utilizza soltanto 31 tag `XHTML` [Per16]. Tali tag, inoltre, sono stati scelti in modo da consentire all'autore di utilizzarli con naturalezza a seconda del loro scopo [4] ed esprimere un chiaro valore semantico non ambiguo [5]. La grammatica di `RASH`, quindi, è stata pensata con particolare attenzione al valore semantico dei tag ammessi; la struttura dei documenti fa anche largo uso dei ruoli e degli attributi previsti dalla specifica `ARIA` [Wor14a], garantendo così una loro lettura ottimale per gli utenti di tecnologie assistive.

`RASH`, poi, è pensato in modo da consentire all'autore di focalizzare l'attenzione sul contenuto dell'articolo piuttosto che sul suo aspetto grafico. Oltre al formato, infatti, `RASH` si compone di un framework che si occupa proprio di garantire una visualizzazione conforme alle aspettative dell'autore ed adatta ad ogni tipo di dispositivo, nonché una serie di strumenti che consen-

[3] https://www.w3.org/community/scholarlyhtml/

[4] per definire una sezione di un documento `RASH`, per esempio, si usa il tag `section`

[5] Ciò avviene, per esempio, in `HTML` e `XHTML` quando si usa il tag `div`

tono di esportare gli articoli nei più comuni formati (come per esempio PDF) utilizzando il layout che si desidera [6].

Va infine sottolineato come RASH, al contrario degli altri formati, presti particolare attenzione alla gestione delle formule matematiche. Non solo è possibile immetterle utilizzando il linguaggio MathML, ma anche LaTeX e la notazione AsciiMath [7], le quali possono essere manipolate molto più facilmente dagli autori rispetto a MathML; il rendering di tutte le notazioni viene poi garantito dal framework attraverso l'uso dellla libreria MathJax [Cer12]. È inoltre disponibile un'estensione del framework per integrare il plugin evographs [MPR+17] all'interno dei documenti RASH.

Dulcis in fundo, Un bonus certamente apprezzabile è che la semplicità della grammatica di RASH consente di creare molto agevolmente convertitori da e per questo formato verso e a partire da altri formati.

1.5 L'uso di RASH e EPUB

Ora che si è individuato nel formato RASH un valido alleato per superare i problemi di accessibilità degli articoli scientifici di cui si è lungamente parlato in questo capitolo, non resta che analizzare il modo in cui possono essere distribuiti i documenti in questo formato e trovare il miglior modo possibile per farlo senza perdere alcuno dei vantaggi visti finora.

Come accennato verso la fine del paragrafo 1.4, è molto semplice scrivere convertitori dal formato RASH ad altri formati; il framework RASH fornisce un convertitore che consente di generare un documento PDF equivalente al documento RASH di partenza a cui viene applicato un certo layout. L'implementazione di questo convertitore, però, internamente si avvale di LaTeX e

[6] Il framework fornisce i layout richiesti dalle principali conferenze in ambito informatico, ma chiunque può sviluppare un layout che soddisfi esigenze tipografiche non soddisfatte dai layout esistenti

[7] Il supporto per la notazione AsciiMath è stato implementato dal sottoscritto in una fase precedente a questo lavoro di tesi. Maggiori informazioni su questa sintassi sono disponibili all'URL http://asciimath.org/

non è quindi una valida alternativa per tutti i problemi già discussi; si avrebbe, insomma, un altro workflow che a partire da un documento accessibile genera un risultato inaccessibile.

C'è però uno standard per la distribuzione di libri in formato elettronico che recentemente sta ottenendo sempre più successo ossia EPUB la cui specifica è attualmente alla versione 3.0.1. Il formato EPUB [Int14a, Int14b] consente di distribuire all'interno di un unico file un documento HTML o XHTML assieme a tutte le risorse di cui esso necessita per poter essere visualizzato correttamente, nonché una serie di metadati sul documento stesso e su tali risorse; le risorse inserite possono essere elementi multimediali come file audio, video e immagini, ma anche fogli di stile CSS e script in JavaScript. All'interno dei documenti distribuiti con questo standard, inoltre, è possibile inserire formule matematiche utilizzando il linguaggio MathML.

Poiché la grammatica di RASH è definita su un sottoinsieme ristretto di tag XHTML, un documento RASH è automaticamente un documento XHTML. I componenti del framework che si occupano della visualizzazione del documento, inoltre, sono implementati proprio attraverso fogli di stile CSS e script JavaScript. Di conseguenza non vi sono ostacoli teorici all'utilizzo del formato EPUB come formato per la distribuzione dei documenti RASH.

Come parte del lavoro di tesi, quindi, è stato sviluppato uno strumento che potesse convertire un documento in formato RASH in un documento in formato EPUB. Occorrerebbe discutere sul fatto che tale processo di conversione non deteriori l'ottima accessibilità dei documenti RASH di partenza, ma l'assenza di peggioramenti da questo punto di vista è banalmente garantita. Infatti si ha che:

- all'interno del file EPUB è presente lo stesso markup del documento RASH di partenza in cui sono stati aggiornati, se necessario, i percorsi da cui caricare le risorse;

- nel file EPUB sono presenti le stesse risorse del framework RASH necessarie alla corretta visualizzazione del documento in un browser;

- le formule matematiche sono tradotte in `MathML` indipendentemente dal formato in cui erano state immesse nel documento originale.

Da ciò deriva che, salvo problemi nell'implementazione dei lettori di ebook `EPUB`, l'accessibilità dell'output del processo di conversione di un documento `RASH` in formato `EPUB` non viene peggiorata in alcun modo rispetto a quella del documento di partenza. Nel capitolo 2 si discuterà in maggior dettaglio tale processo di conversione e lo strumento che lo implementa.

Capitolo 2

Conversione dei documenti da RASH a EPUB

Parte di questo lavoro di tesi ha riguardato lo sviluppo dello strumento `rash2epub`, una utility che consente di convertire un documento in formato `RASH` nel suo equivalente in formato `EPUB`. Dopo aver brevemente introdotto il formato per la distribuzione di ebook `EPUB` ed averne brevemente descritto le caratteristiche, in questo capitolo verrà fornita una panoramica ad alto livello sullo strumento e sul processo di conversione evidenziandone i punti di forza e sottolineandone le caratteristiche significative dal punto di vista dell'accessibilità.

2.1 Il formato EPUB

Prima di procedere ad una panoramica sullo strumento `rash2epub` e sul processo di conversione da esso implementato, è opportuno fornire qualche informazione in più sullo standard `EPUB` che consenta di comprendere perché esso sia particolarmente indicato per raggiungere gli scopi prefissati nel capitolo 1.

Il formato `EPUB`, contrazione di *Electronic Publication*, è un formato per la distribuzione di libri elettronici (ebook) la cui prima versione è stata ri-

lasciata nel dal *International Digital Publishing Forum (IDPF)*. EPUB è lo standard aperto basato su XML per la distribuzione di ebook più supportato [Wika]: oltre che su un elevatissimo numero di dispositivi hardware, infatti, è possibile leggere i libri in questo formato con varie applicazioni disponibili per tutti i sistemi operativi in circolazione. Il successo di EPUB è testimoniato anche dalla sua larga adozione all'interno dell'industria editoriale per la distribuzione di testi di vario genere. La versione più recente dello standard è la 3.0.1, rilasciata dal IDPF nel .

Il formato EPUB si basa su tecnologie Web preesistenti. Un file EPUB, infatti, contiene:

- uno o più documenti XHTML, che definiscono il contenuto vero e proprio della pubblicazione;
- uno o più fogli di stile CSS, che definiscono il layout per la visualizzazione del contenuto della pubblicazione;
- ogni altra risorsa (ad esempio le immagini) necessaria alla visualizzazione del contenuto;
- alcuni file XML che contengono l'indice ed una serie di metadati sulla pubblicazione e sulle risorse che essa contiene, nonché l'ordine di lettura dei documenti XHTML inclusi nell'ebook.

Tutte queste informazioni sono poi incluse in un unico file sotto forma di archivio compresso informato zip rinominato per avere l'estensione epub.

Potendo contare sulle principali tecnologie Web (compresa una buona parte del linguaggio JavaScript) si possono creare ebook in formato EPUB estremamente dinamici ed sofisticati, come per esempio avviene nel caso della realizzazione di pubblicazioni con layout cosiddetti *responsive*, ovvero che si adattano per fornire la miglior esperienza di lettura a seconda del dispositivo utilizzato dall'utente, oppure di ebook che forniscono esperienze didattiche interattive.

Un'importante novità introdotta con la versione 3.0 dello standard riguarda la possibilità di inserire formule matematiche all'interno delle pubblicazioni utilizzando il linguaggio MathML. Per supportare lo standard EPUB, quindi, tutti i lettori di questo formato dovrebbero essere in grado di effettuare il rendering delle formule matematiche inserite utilizzando MathML; qualora questo linguaggio non fosse supportato opportunamente, però, è possibile l'utilizzo di polifill come MathJax poiché come detto in una pubblicazione EPUB è possibile inserire ed eseguire codice JavaScript.

Si può dunque dedurre come il formato EPUB sia particolarmente adatto per la distribuzione di ebook accessibili. Poiché questo standard si basa sul formato XHTML vale quanto detto nel capitolo 1 a proposito dei workflow che prevedono l'uso di documenti HTML e XHTML e della loro accessibilità. Potendo contare sulla presenza del linguaggio MathML, poi, è possibile garantire anche l'accessibilità delle formule matematiche presenti all'interno delle pubblicazioni. EPUB, perciò, costituisce il candidato ideale per la distribuzione di documenti RASH poiché consente di mantenere tutti i vantaggi di quest'ultimo, permettendo nello stesso tempo di distribuire in un unico file sia il documento che le risorse necessarie affinché esso venga visualizzato correttamente.

2.2 Una panoramica su rash2epub

Dopo aver esaminato il formato EPUB e come esso permetta di distribuire pubblicazioni (anche a carattere scientifico) estremamente dinamiche ed accessibili, si introdurrà ora lo strumento rash2epub, un tool a riga di comando che consente di generare un file EPUB contenente un documento in formato RASH e tutte le risorse necessarie per visualizzarlo correttamente, comprese quelle fornite dal framework. Lo strumento genera dinamicamente tutti i metadati sulle risorse da includere, ricavando laddove necessario (per esempio nel caso delle immagini) le informazioni dal documento da convertire. Poi genera le strutture XML richieste dal formato EPUB in cui vengono inseriti i

metadati così ricavati, copia i file del framework RASH indispensabili per la corretta visualizzazione del documento ed infine "impacchetta" il tutto in un unico file compresso come da specifica [Int14a, Int14b].

Per svolgere il suo compito lo strumento ha bisogno di alcune dipendenze: prima di avviare il processo di conversione, dunque, esso verifica che queste siano disponibili e mostra opportuni messaggi d'errore qualora non lo fossero indicando all'utente come risolvere il problema. Oltre che per l'assenza delle dipendenze, il processo di conversione potrebbe fallire per vari motivi tra cui:

- la non conformità del documento alla grammatica del formato RASH: è piuttosto logico predisporre un processo di conversione che funzioni per i documenti conformi alla grammatica del formato e che fallisca qualora essi non lo siano, informando l'utente del problema; un documento non conforme alla grammatica di RASH, in ogni caso, avrebbe problemi ben più seri (per esempio nella visualizzazione in un browser) ancora prima della sua conversione;

- l'impossibilità di recuperare tutte le risorse necessarie per la visualizzazione del documento: questo problema è particolarmente rilevante nel caso delle immagini. rash2epub, infatti, tenta di ottenere le risorse sia dal filesystem che da eventuali URL di rete a seconda dei percorsi da cui esse vengono caricate dal documento originale ma, se uno di questi tentativi non va a buon fine, il processo di conversione fallisce informando l'utente del problema. Non è possibile generare file EPUB incompleti poiché si violerebbe la specifica e si modificherebbe il contenuto del documento di partenza, cosa che rash2epub tenta di fare solo quando strettamente necessario ed in situazioni estremamente controllate;

- la presenza di una formula matematica non convertibile in MathML: ciò di solito accade quando nel contenuto della formula vi sono evidenti errori di sintassi. In questi casi il processo di conversione fallisce e l'utente viene informato del problema; questo accorgimento serve a garantire la

correttezza del contenuto del file EPUB generato, come richiesto dallo standard.

I file EPUB generati da rash2epub possono essere considerati a tutti gli effetti dei file per la distribuzione di documenti in formato RASH, poiché le modifiche al contenuto del documento di partenza apportate dallo strumento si limitano all'aggiornamento dei percorsi da cui caricare le risorse esterne, così che esse vengano caricate dall'interno del file EPUB anzicché dal percorso originale, e all'eventuale traduzione delle formule matematiche nel loro equivalente in MathML. Tale processo di traduzione delle formule potrebbe in effetti creare dei problemi, ma la sua implementazione è stata effettuata in modo tale da prevenire questo rischio: rash2epub, infatti, si limita ad utilizzare opportunamente i meccanismi forniti dalla libreria MathJax esattamente come farebbe il framework RASH durante la fase di visualizzazione del documento all'interno di un browser. Da quanto detto ne deriva che il processo di conversione conserva tutte le caratteristiche positive del formato RASH, incluse quelle relative all'accessibilità come discusso nel paragrafo 1.5 e preserva ogni altra proprietà del documento originale.

Oltre a complementare un workflow per la generazione di articoli scientifici in formato accessibile, quindi, rash2epub contribuisce a colmare una mancanza nel framework emersa già da tempo [1].

2.3 RASH2EPUB, lato utente

Dopo avere fornito una breve panoramica sull'utilità e gli scopi dello strumento rash2epub, ne verrà ora descritto l'utilizzo dal punto di vista dell'utente.

[1] Per maggiori dettagli al riguardo vedere
Identification of a "normalised" form for RASH documents - https://github.com/essepuntato/rash/issues/60 e
From RASH to EPUB - https://github.com/essepuntato/rash/issues/61

La figura 2.1 mostra l'output fornito quando si invoca all'interno di un terminale il tool `rash2epub` passando come argomento il flag "-h", ossia chiedendo di mostrare le opzioni disponibili inserendo il comando `rash2epub -h`.

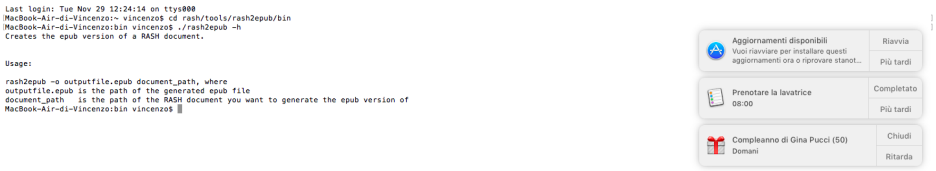


Figura 2.1: La figura mostra la schermata d'aiuto che si ottiene in un terminale invocando lo strumento `rash2epub` passandogli come opzione il flag "-h". Il contenuto dello screenshot è riportato nell'appendice C.

innanzitutto è facile notare che l'output del comando viene restituito in inglese; ciò avviene perché lo strumento è pensato per l'inclusione all'interno del framework `RASH` che viene distribuito a livello internazionale. Non è quindi prevista alcuna localizzazione in italiano dell'output e degli eventuali messaggi di errore.

Il tool `rash2epub` deve essere invocato seguendo la sintassi

```
rash2epub -o nome.epub percorso-documento
```

dove:

- *-o* è l'opzione con cui indicare il nome del file `EPUB` da generare,
- *nome.epub* è il nome da assegnare al file `EPUB` generato e

- *percorso-del-documento* è il percorso (assoluto o relativo) del documento in formato `RASH` che si desidera convertire in formato `EPUB`.

Durante l'esecuzione `rash2epub` mostra vari messaggi informativi per indicare all'utente l'attività eseguita. Se il processo di conversione viene completato con successo il file `EPUB` generato viene salvato nella directory corrente, ossia la directory in cui si trovava l'utente quando è stato impartito il comando per avviare la conversione, (in altre parole l'output del comando `pwd`). Se invece dovesse verificarsi un errore il processo di conversione viene terminato ed un opportuno messaggio informativo contenente una spiegazione e dei suggerimenti per la risoluzione del problema sarà mostrato all'utente. Il processo di conversione si interrompe ad ogni messaggio d'errore: essendo composto da varie fasi dipendenti l'una dall'altra, infatti, proseguire nonostante un errore potrebbe portare a conseguenze difficili da prevedere e, soprattutto, alla creazione di file `EPUB` non conformi alle specifiche del formato.

Ma come si svolge effettivamente il processo di conversione "under the hood" (sotto il cofano)? E come si comporta `rash2epub` in aspetti delicati quali la gestione delle formule matematiche? Tutto ciò che concerne l'implementazione dello strumento sarà discusso nei minimi dettagli nel capitolo 3.

Capitolo 3

`Rash2epub`: implementazione

Come si è visto nel paragrafo 2.3, lo strumento `rash2epub` prende in input il percorso di un documento `RASH` ed il nome del file `EPUB` da generare e restituisce in output la conversione in formato `EPUB` di quel documento. In questo capitolo sarà analizzata più in dettaglio l'architettura dello strumento e le modalità con cui avviene il processo di conversione, nonché i principali problemi da esso derivanti e le soluzioni adottate per superarli.

3.1 L'architettura del tool

La figura 3.1 mostra il diagramma di sequenza che descrive il processo di conversione di un documento `RASH` in formato `EPUB` svolto dallo strumento `rash2epub`. Nell'appendice D è possibile trovare una descrizione testuale del diagramma ed alcuni dettagli tecnici su come esso sia stato realizzato.

Il componente principale dello strumento `rash2epub` è costituito da uno script `BASH` il quale, dopo aver preso in input il percorso del documento `RASH` da convertire ed il nome del file `EPUB` da generare, esegue il processo di conversione attraverso varie fasi. È stata valutata l'idea di realizzare lo script principale in linguaggio `python`, ma vista la necessità di invocare diversi strumenti esterni a riga di comando scrivere uno script `BASH` è stato più conveniente poiché ha consentito di ottenere il risultato con un minor

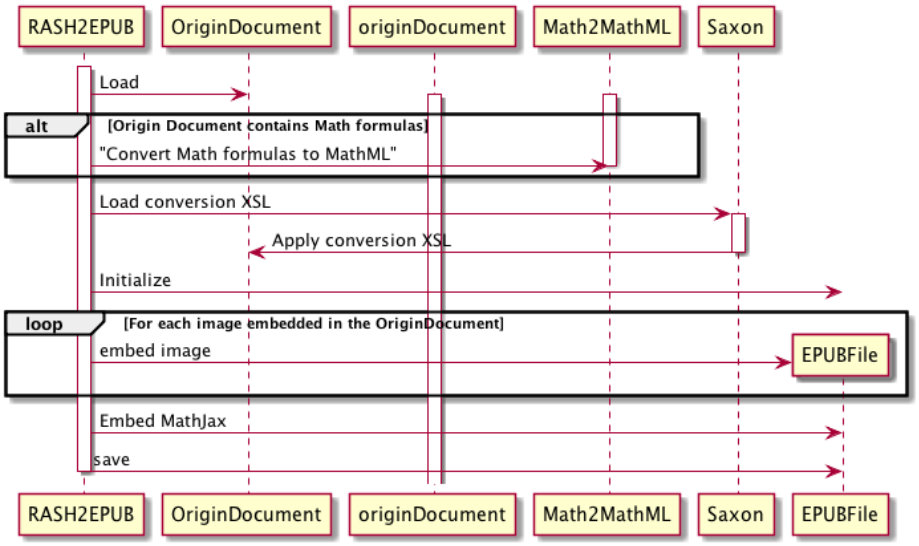


Figura 3.1: Diagramma di sequenza che descrive il processo di conversione di un documento RASH in formato EPUB eseguito da RASH2EPUB.

numero di righe di codice. Il processo di conversione svolto dallo script si può schematizzare nelle seguenti fasi:

- verifica delle dipendenze;
- conversione delle formule matematiche;
- applicazione di un foglio di stile XSL [Wor07b];
- generazione del file EPUB.

La conversione delle formule matematiche sarà analizzata approfonditamente nel paragrafo 3.3, mentre di seguito si esamineranno le altre fasi del processo di conversione.

3.1.1 Verifica delle dipendenze

In questa fase lo script verifica che nel sistema in cui viene eseguito siano presenti tutte le dipendenze di cui esso necessita per svolgere il suo compito; in particolare, viene verificata la presenza dei tool:

- `curl`, utilizzato per il download di eventuali immagini caricate da remoto dal documento originale;

- `nodeJS`, necessario per l'esecuzione dello strumento `math2mathml`;

- `saxon`, necessario per l'applicazione del foglio di stile `XSL`;

- una versione del comando `sed` che supporti le estensioni GNU;

- `zip`, necessario per completare la generazione del file `EPUB`.

Alcune note particolari meritano i controlli eseguiti per verificare la presenza di due dipendenze: `sed` e `saxon`.

Nel primo caso, oltre a verificare la presenza dello strumento `sed`, è necessario assicurarsi che esso supporti alcune estensioni GNU che invece non sono supportate dalla versione descritta nello standard POSIX (cf. [Fre04] e [IEE16]); in particolare, le estensioni GNU del comando `sed` di cui necessita `rash2epub` riguardano la possibilità di eseguire una ricerca in modo case-insensitive e di poter sostituire le occorrenze di un'espressione regolare con un'altra espressione regolare all'interno del testo di uno stesso file. Sebbene soddisfare questa dipendenza può sembrare oneroso per l'utente, va detto che tutti i sistemi GNU/Linux integrano di default la versione del comando `sed` che supporta tutte le estensioni GNU; inoltre, installare una versione di sed che supporti tali estensioni sui sistemi che non la contengono (come quelli che derivano da Free BSD, tra i quali spicca Mac OS) di solito è un'operazione molto semplice.

Nel secondo caso, invece, si è scelto di usare `saxon` come processore `XSLT` poiché supporta completamente `XSLT 2.0` e `XPath 2.0` (si veda [Kay15]). Anche la procedura di installazione di `saxon` sui sistemi operativi più diffusi è molto semplice.

3.1.2 Applicazione di un foglio di stile XSL

A questo punto lo script `rash2epub` procede con l'applicazione di un foglio di stile `XSL` al documento `RASH` utilizzando il processore `XSLT saxon`. Il foglio di stile genera i principali elementi richiesti dal formato `EPUB` ([Int14b], ossia:

- il file *mimetype*, che, come prevsito dallo standard `EPUB` specifica il mimetype "application/epub+zip" per la pubblicazione;
- il file *container.xml*, che dichiara la posizione del file content.opf all'interno dell'ebook;
- il file *content.opf*, che contiene:
  1. i metadati del file `EPUB` tra cui un identificatore univoco, la lingua, la data di ultima modifica, gli autori ed il titolo del documento;
  2. la sezione *manifest*, che dichiara tutte le risorse che fanno parte del file `EPUB`;
  3. la sezione *spine*, che indica le risorse a partire dalle quali è possibile raggiungere tutte le altre ed il loro ordine di lettura;
- il file *document.xhtml*, che contiene il documento `RASH` modificato attraverso i passaggi precedenti del processo di conversione;
- il file *toc.ncx*, che contiene l'indice della pubblicazione; tale indice elenca tutte le sezioni del documento a qualunque livello gerarchico esse si trovino ed espone la gerarchia come previsto dallo standard `EPUB`, lasciando al sistema di lettura il compito di decidere quante informazioni visualizzare e come;
- il file *toc.xhtml*, che contiene lo stesso indice visto al punto precedente ma in formato `XHTML`; sebbene non sia richiesto dalla versione 3.0 dello standard `EPUB`, esso è stato incluso per fornire retrocompatibilità ai sistemi di lettura che non supportano la gestione dell'indice in formato `NCX`;

Durante questa fase viene generato un ID univoco per le sezioni del documento che ne sono sprovviste, così da garantire che anche esse possano essere raggiunte attraverso l'indice.

3.1.3 Generazione del file EPUB

In questa fase viene portato a termine il processo di conversione del documento RASH nel suo equivalente in formato EPUB. In particolare:

- vengono copiati vari file di supporto necessari per la corretta visualizzazione del file EPUB da parte dei sistemi di lettura, provenienti sia dal framework RASH che da alcune librerie (bootstrap e JQuery) da cui dipende il suo funzionamento;

- vengono copiati i file che compongono la versione ad oc della libreria MathJax (si veda il paragrafo 3.3 per maggiori dettagli);

- viene generato un archivio in formato zip secondo le regole dettate dallo standard EPUB, il quale viene poi rinominato utilizzando il nome del file fornito in input dall'utente all'inizio del processo di conversione e l'estensione epub.

Il processo genera vari file temporanei i quali a questo punto vengono eliminati poiché inutili.

3.2 Problemi della conversione

Stando a quanto finora descritto il processo di conversione di un documento RASH in formato EPUB sembrerebbe piuttosto lineare e privo di problemi. In realtà quanto detto nel paragrafo 3.1 non tiene conto di alcune problematiche che verranno analizzate in dettaglio di seguito.

3.2.1 Gestione degli errori

Per prima cosa è doveroso dire che tutte le fasi del processo descritte nel paragrafo 3.1 potrebbero fallire per diversi motivi. Pertanto lo script `rash2epub` verifica che la fase precedente sia stata terminata con successo prima di procedere alla seguente; inoltre, in caso di errore informa l'utente e, quando possibile, fornisce indicazioni sulle cause ed eventuali soluzioni del problema.

3.2.2 Inclusione delle risorse esterne

All'interno di un file `EPUB` devono essere presenti tutte le risorse di cui esso ha bisogno per poter essere visualizzato da parte di un sistema di lettura. I documenti in formato `RASH` necessitano di alcune risorse esterne [Per16] per essere visualizzati correttamente e, perciò, `rash2epub` deve assicurarsi che esse siano incluse nell'ebook derivante dalla conversione del documento e che i percorsi da cui esse vengono caricate siano aggiornati in modo da fare riferimento alla loro posizione all'interno del file `EPUB` e non alla loro posizione originale. L'aggiornamento dei percorsi alle risorse esterne viene eseguito durante l'applicazione del foglio di stile `XSL`, mentre l'inserimento di tali risorse nell'ebook viene eseguito successivamente dallo script `BASH`. Alcune risorse indispensabili alla visualizzazione del documento vengono copiate dalla cartella del framework `RASH` in cui è contenuto il tool `rash2epub`: è pertanto consigliabile che un documento venga convertito con la versione di `rash2epub` distribuita nella versione del framework `RASH` usata per scriverlo onde evitare problemi di compatibilità.

Una particolare categoria di risorse esterne utilizzabili all'interno di un documento `RASH`è costituita dalle immagini, le quali devono essere ovviamente incluse nel file `EPUB` generato. Il percorso di un immagine caricata all'interno di un documento `RASH` può essere di tre tipi, ognuno dei quali viene gestito in modo diverso da rash2epub:

- un percorso assoluto ad un file presente nel filesystem: in questo caso `rash2epub` si limita a copiare l'immagine dal percorso originale ad una cartella temporanea chiamata *img*;

- un percorso relativo ad un file presente nel filesystem: in questo caso `rash2epub` risolve il percorso relativo nell'equivalente percorso assoluto e copia l'immagine dal percorso ottenuto alla sottocartella *img*;

- un URL ad una risorsa esterna: in questo caso `rash2epub` scarica l'immagine nella cartella *img*.

Come si può notare facilmente in tutti e 3 i casi lo script posiziona le immagini in una cartella chiamata *img*. Non si tratta di una coincidenza, ma piuttosto di un comportamento voluto: in questo modo per integrare tutte le immagini nel file `EPUB` è sufficiente aggiungere all'ebook proprio la cartella *img* così ottenuta. La modifica dei percorsi da cui vengono caricate le immagini che avviene durante l'applicazione del foglio di stile `XSL` tiene conto di questo aspetto.

3.2.3 Compressione

In questa fase viene creato un archivio compresso in formato `zip`, poi rinominato con il nome appropriato e l'estensione `epub`. Questa fase sarebbe piuttosto banale, ma per osservare la specifica dello standard `EPUB` [Int14a, Int14b] è necessario adottare alcuni accorgimenti che ne complicano l'implementazione. La compressione viene eseguita attraverso l'utility `zip`, uno strumento a riga di comando che crea archivi compressi nell'omonimo formato.

Innanzitutto il file *mimetype* deve essere il primo file in assoluto presente nell'archivio; ad esso, inoltre, non devono essere applicate le alterazioni che vengono eseguite sui file quando diventano parte di un archivio in formato `zip`. Tale risultato si ottiene passando il flag `-0` all'utility zip quando viene aggiunto il file *mimetype* all'archivio creato.

Per come è strutturato il processo di conversione dei documenti può capitare di dover aggiungere intere directory al file `zip` generato come risultato intermedio; pertanto, quando vengono aggiunti tutti gli altri files e directory all'archivio viene passato all'utility `zip` il flag `-r`, che indica di aggiungere e comprimere ricorsivamente tutti gli elementi presenti in una directory inserita nell'archivio. Viene anche passato il flag `-9`, che indica all'utility `zip` di usare il massimo livello di compressione possibile così da minimizzare le dimensioni del file generato.

La gestione del filesystem di alcuni sistemi operativi prevede che nelle cartelle create siano inseriti automaticamente alcuni file e sottocartelle nascosti all'utente ma utilizzati dallo stesso sistema operativo. L'inclusione di tali file e sottocartelle nel file `EPUB` generato non ha alcuna utilità pratica, oltre a costituire una potenziale violazione dello standard. Pertanto vengono passati altri due parametri all'utility `zip`:

- il flag `-X`, che indica di non includere nell'archivio generato tutte le sottocartelle non visibili all'utente;
- il parametro `-x` con valore `*.DS_Store`, che indica di non aggiungere all'archivio il file ".DS_Store" inserito automaticamente in ogni sottocartella dal sistema operativo Mac OS.

Questi ultimi due accorgimenti si rendono necessari poiché i sistemi operativi potrebbero applicare le loro alterazioni tipiche del filesystem anche quando i file e le sottodirectory vengono create programmaticamente tramite gli strumenti a riga di comando.

3.3 Gestione delle formule matematiche

Uno dei più importanti problemi della conversione di un documento `RASH` in formato `EPUB` è costituito dalla gestione delle formule matematiche in esso contenute. Il framework `RASH`, infatti, si avvale della libreria `MathJax` per consentire all'utente di utilizzare molteplici modalità di input per le formule,

nonché per assicurare che esse vengano visualizzate correttamente indipendentemente dal dispositivo usato per leggere il documento. Ad una prima analisi superficiale, dunque, si potrebbe pensare che si stia parlando di un non-problema: basterebbe infatti integrare `MathJax` all'interno dei file `EPUB` generati da `rash2epub` per ottenere lo stesso risultato. Ed in effetti la direzione da percorrere è proprio questa, ma la faccenda è un po' più delicata. `MathJax`[1], infatti, ha una dimensione di $62,3MB$. All'interno del framework `RASH` la dimensione di `MathJax` non è un problema, poiché la libreria viene prelevata dinamicamente dal CDN ufficiale e, sfruttandone l'elevata modularità, vengono caricati soltanto i componenti strettamente necessari per visualizzare il documento. Includere l'intera libreria in ogni file `EPUB` generato da `rash2epub` creerebbe ebook di dimensioni troppo elevate. È stato dunque necessario individuare una strategia per includere la libreria negli ebook generati minimizzandone il footprint.

3.3.1 Conversione delle formule matematiche

Come accennato in precedenza all'interno di un documento `RASH` è possibile inserire formule matematiche in tre modi diversi:

- utilizzando il linguaggio `MathML`;
- utilizzando il linguaggio LaTeX;
- utilizzando la sintassi `ASCIIMath`;

Di questi tre modi soltanto il primo è supportato ufficialmente nella specifica del formato `EPUB` [Int14a]. Pertanto, se il documento `RASH` da convertire contiene formule matematiche scritte con la sintassi LaTeX e/o `ASCIIMath`, esse vengono convertite nel loro equivalente in sintassi `MathML`. Tale operazione è resa possibile dallo strumento `math2mathml`, uno script per `nodeJS`

[1] Al momento in cui è stato sviluppato `rash2epub` l'ultima versione stabile di `MathJax` disponibile è la versione 2.7.0; tutti i dati e gli sviluppi descritti fanno riferimento a tale versione.

creato dal sottoscritto in una fase precedente a questo lavoro di tesi. Lo script `math2mathml` prende in input il percorso del documento che contiene le formule matematiche da convertire ed il percorso in cui salvare il documento risultante dopo la conversione e converte le formule matematiche sfruttando la stessa libreria `MathJax` usata dal framework `RASH`. Perciò, la sostituzione delle formule con il loro equivalente in `MathML` effettuata da `math2mathml` non altera le stesse rispetto a ciò che vedrebbe l'utente visualizzando il documento originale in un browser.

3.3.2 Integrazione di `MathJax` nei file generati

Se il mondo reale fosse un mondo perfetto il problema della gestione delle formule matematiche sarebbe risolto banalmente con la loro conversione in `MathML` che, in quanto ufficialmente incluso nello standard `EPUB`, dovrebbe essere supportato da tutti i sistemi di lettura che gestiscono questo formato di ebook. Anche in questo caso, però, la situazione è analoga a quella dei browser descritta in precedenza; per garantire la corretta visualizzazione degli ebook generati, perciò, è necessario che essi includano la libreria `MathJax` da usare come polifill. La tabella 3.1 mostra le operazioni compiute per ridurre le dimensioni della libreria `MathJax` e quanto esse contribuiscano a diminuirla.

Una volta scompattato l'archivio contenente la libreria `MathJax` si ottiene una cartella le cui dimensioni sono di $62.3MB$. Le operazioni mostrate nella tabella 3.1 modificano la struttura di tale cartella rimuovendo i componenti non necessari al funzionamento di `MathJax` all'interno dei file generati da `rash2epub` e saranno illustrate più in dettaglio di seguito.

La rimozione della documentazione e degli esempi d'uso consiste nella rimozione degli elementi

- *.gitignore, .npmignore e .travis.yml*, file di supporto per la distribuzione della libreria;
- *bower.json, package.json e composer.json*, file che consentono di installare la libreria rispettivamente tramite gli strumenti `bower`, `NPM` (fornito

Operazione	Riduzione di dimensione	Dimensione complessiva
Rimozione della documentazione e degli esempi d'uso	$103KB$	$62,3MB$
Rimozione del supporto per la localizzazione	$1MB$	$61,3MB$
Rimozione del supporto per debugging	$16,8MB$	$44,5MB$
Scelta della configurazione MML_SVG-full	$7,7MB$	$36,8MB$
Rimozione dei file di supporto a funzionalità non usate | $25,7MB$	$11,1MB$	
Rimozione dei file per l'output in formato SVG già inclusi nella configurazione | $101KB$	$11MB$	
Rimozione dei font SVG non utilizzati dalla configurazione	$9,5MB$	$1,5MB$

Tabella 3.1: Operazioni compiute per ridurre le dimensioni della libreria `MathJax`

da `nodeJS`) e `composer`;

- *CONTRIBUTING.md*, file contenente indicazioni sulle modalità per contribuire allo sviluppo del progetto;
- *docs*, cartella contenente la documentazione;
- *README.md*, file contenente informazioni di base e istruzioni per usare la libreria;
- *test*, cartella contenente vari esempi d'uso;

La rimozione del supporto per la localizzazione, invece, consiste nell'eliminare la cartella *localization* contenente vari file che consentono alla libreria di mostrare gli eventuali messaggi d'errore nella lingua usata dall'utente. In assenza del supporto per la localizzazione `MathJax` mostrerà tali messaggi in inglese.

La cartella della libreria `MathJax` contiene al suo interno 2 versioni di alcuni componenti: la prima è ottimizzata per la distribuzione, mentre la seconda fornisce un maggiore supporto per il debugging di eventuali problemi. Questa seconda versione di tali componenti si trova nella cartella *unpacked*, che quindi è stata eliminata.

`MathJax` ha inoltre molteplici funzionalità ,le quali possono essere abilitate a seconda delle proprie esigenze. Per farlo essa fornisce vari profili di configurazione pronti all'uso. Poiché lo strumento `math2mathml` converte le formule di un documento `RASH` in `MathML`, per visualizzare correttamente le formule nei file `EPUB` generati da `rash2epub` è sufficiente che `MathJax` supporti questa modalità di input. La libreria permetterebbe di utilizzare anche diverse modalità di output, ma di queste la modalità `SVG` è quella che fornisce i migliori risultati dal punto di vista della visualizzazione. Pertanto è stato scelto di usare il profilo di configurazione `MML_SVG` che abilita solo le modalità di input e di output necessarie. Inoltre, ogni profilo di configurazione viene distribuito in due varianti: una detta "standard", che contiene soltanto i parametri per abilitare le funzionalità necessarie, ed una detta "full", che contiene sia i parametri per abilitare le funzionalità richieste che le estensioni necessarie. Pertanto scegliendo la variante "full" della configurazione "MML_SVG" è stato possibile eliminare la cartella *extensions* oltre che tutti i file presenti nella cartella *config* tranne *"MML_SVG-full.js"* che contiene proprio il profilo di configurazione nella variante utilizzata.

Una volta scelta la configurazione *MML_SVG-full* è possibile eliminare anche tutti file della libreria che supportano modalità di input e di output non usate dalla configurazione, ossia:

- la cartella *fonts*, contenente definizioni di font non usate con la modalità

di output `SVG`;

- la cartella *input*, contenente i file di supporto per le varie modalità di input; i file che supportano l'input in linguaggio `MathML` sono già inclusi nel file di configurazione scelto in precedenza;

- tutte le sottocartelle della cartella *output* esclusa la cartella *SVG*, che contengono il supporto alle modalità di output non utilizzate;

Infine è possibile ridurre ulteriormente le dimensioni della libreria `MathJax` eliminando i file di supporto per il formato di output `SVG` già inclusi nel file di configurazione e i font non utilizzati. Si arriva così ad ottenere una versione di `MathJax` che contiene tutte e sole le funzionalità necessarie a garantire una visualizzazione corretta delle formule matematiche presenti negli ebook generati da `rash2epub`, ma con una dimensione di $1,5MB$ che, una volta compressa, aggiunge un footprint ai file `EPUB` di circa $700KB$. Tale footprint è decisamente minore rispetto a quello che avrebbe avuto l'inclusione della versione standard di `MathJax` (circa 30 MB) ed è più che accettabile.

La versione di `MathJax` così ottenuta è presente tra i file di supporto dello strumento `rash2epub` nella cartella denominata *MathJax-lite*. Essa viene inserita in tutti i file `EPUB` generati da `rash2epub` e i file che la compongono sono dichiarati all'interno della sezione *manifest* del file *content.opf* generato durante l'applicazione del foglio di stile `XSL`. La generazione di tali dichiarazioni è stata automatizzata attraverso uno script denominato *genmanifest.sh* che prende in input la lista dei file che compongono *MathJax-lite* e restituisce il frammento della sezione del file *content.opf* che dichiara tutti i file presenti nella lista.

Va infine notato che tutte le operazioni descritte potrebbero essere eseguite usando lo strumento `MathJax-grunt-cleaner` [Mat16] che, al di là delle molteplici dipendenze, non sembra svolgere il suo compito come descritto dalla (troppo scarsa) documentazione. È stato perciò necessario implementare la strategia descritta manualmente piuttosto che utilizzando tale strumento.

Capitolo 4

Valutazione

Dopo aver descritto nel dettaglio l'implementazione dello strumento `rash2epub`, in questo capitolo saranno descritti i test eseguiti per assicurare il corretto funzionamento dello strumento `rash2epub`. Si cercherà inoltre di ipotizzare un possibile workflow per la scrittura di articoli scientifici in formato `RASH` che preveda l'uso di tale strumento, il quale sarà poi comparato a quelli descritti nel capitolo 1 evidenziandone i punti di forza.

4.1 Test eseguiti

Per assicurare il corretto funzionamento dello strumento `rash2epub` ed il rispetto delle proprietà che esso doveva garantire secondo le specifiche illustrate finora sono stati eseguiti diversi test che verranno documentati in questo paragrafo.

Per verificare il corretto funzionamento dello strumento `rash2epub` è stato dapprima seguito un approccio *divide et impera*, ovvero è stato verificato il corretto funzionamento su documenti `RASH` molto semplici che contenessero soltanto una specifica caratteristica del formato `RASH` da gestire durante l'esportazione del documento in formato `EPUB`. Lo strumento è stato perciò testato su:

- un documento contenente solo elementi testuali;

- un documento contenente solo una tabella;
- un documento contenente solo una figura;
- un documento contenente solo un listato di codice;
- per ognuna delle tre sintassi (`ASCIIMath`, LaTeX e `MathML`) utilizzabili per l'immissione delle formule nel formato `RASH`, un documento contenente solo una formula matematica;
- un documento contenente soltanto riferimenti bibliografici ed un frammento di testo in cui essi venivano citati;
- un documento contenente soltanto note a piè di pagina ed un frammento di testo che facesse riferimento ad esse.

Tutti i documenti utilizzati per eseguire i test erano ovviamente conformi alla grammatica del formato `RASH`. I test dei documenti contenenti le formule scritte con la sintassi `ASCIIMath` e LaTeX assumevano il corretto funzionamento dello strumento `math2mathml` che era già stato verificato in una fase precedente a questo lavoro di tesi.

Una volta verificato il corretto funzionamento di `rash2epub` su documenti `RASH` così elementari, si è proceduto al testing su documenti più complessi che contenessero varie combinazioni degli elementi testati in precedenza. È stato poi verificato il corretto funzionamento dello strumento su documenti `RASH` contenenti varie sezioni e sottosezioni e si è verificato che la generazione del file `EPUB` esportasse correttamente i metadati del documento.

Infine è stato utilizzato lo strumento `rash2epub` per convertire in formato `EPUB` vari documenti `RASH` disponibili in rete.

La correttezza dei risultati generati è stata verificata sia con controlli automatizzati che con test manuali. I controlli automatizzati miravano a verificare:

- la validità degli ebook generati da `rash2epub` secondo lo standard `EPUB`; tali controlli sono stati eseguiti utilizzando lo strumento `epubcheck` [Int] fornito per questo scopo da *IDPF*;

- la conformità dell'output generato da `rash2epub` alla grammatica di `RASH`; tale controllo è stato effettuato per garantire che le eventuali modifiche ai percorsi da cui caricare le risorse apportate durante il processo di conversione rispetto al documento originale non ne alterassero in alcun modo la conformità alla grammatica di `RASH`.

Infine sono state simulate le varie condizioni di errore in cui il processo di conversione dovesse fallire:

- assenza di una o più dipendenze necessarie al funzionamento di `rash2epub`;
- tentativo di conversione di un documento non conforme alla grammatica del formato `RASH`;
- Tentativo di conversione di una formula matematica non traducibile in `MathML`.

ed è stato verificato che effettivamente questo fallisse e lo strumento `rash2epub` mostrasse il messaggio d'errore appropriato.

4.2 Profiling

Dopo aver verificato il corretto funzionamento dello strumento `rash2epub` come descritto, si è cercato di comprenderne le prestazioni, valutando il tempo impiegato per convertire alcuni documenti `RASH` in formato `EPUB`. Il *profiling* effettuato non è certamente esaustivo e completo, ma i dati ricavati permettono di avere indicazioni di massima sull'ordine di tempo necessario per eseguire la conversione di un documento. I dati sono riportati nella tabella 4.1; le misurazioni sono state eseguite utilizzando un macbook Air mid con unità SSD da 128GB, processore Intel Core I7 e 8GB di memoria RAM.

Tutti i documenti di cui è stato misurato il tempo di conversione sono distribuiti come documentazione, template o esempi d'uso all'interno del framework `RASH`; alcuni di questi (`RASH` in `DOCX` e `RASH` in `ODT`) sono stati prima convertiti in formato `RASH` dal loro formato originale attraverso strumenti

Documento	Lunghezza (in parole)	Figure	Tabelle	Formule	Listati	Tempo impiegato (in secondi)
Documentazione di RASH [Per16]	4651	2	1	2	1	0.96
Evaluating citation functions in CiTO...	6522	1	3	0	0	0.67
RASH in DOCX	2902	15	2	1	1	1.05
RASH in ODT	3027	15	2	1	1	0,83
Template per documenti	302	2	1	2	1	1

Tempo impiegato per la conversione di alcuni documenti in formato RASH in formato EPUB utilizzando rash2epub

Tabella 4.1: Dati profiling rash2epub

anch'essi inclusi nel framework. Dai dati rilevati non è possibile ricavare relazioni tra grandezze dei documenti (come ad esempio lunghezza in parole, numero di tabelle, numero di figure e numero di listati) e tempo di conversione, ma rianalizzando l'implementazione di rash2epub discussa nel capitolo 3 è possibile isolare alcuni fattori che possono incidere significativamente sulla quantità di tempo impiegata per convertire un documento RASH in formato EPUB:

- il numero e la dimensione delle immagini caricate da remoto: per completare il processo di conversione esse dovranno essere scaricate e, perciò, a seconda della velocità della connessione di rete il processo impiegherà una certa quantità di tempo per completare l'operazione;
- il numero di figure presenti nel documento: le immagini hanno dimensioni significative e, perciò, la loro copia e compressione all'interno del file EPUB può richiedere una quantità di tempo significativa;

- il numero di formule matematiche da convertire in `MathML`: sebbene lo strumento `math2mathml` sia piuttosto efficiente nello svolgere il suo compito, se il numero di formule da convertire è elevato la quantità di tempo impiegata per svolgere quest'operazione diventa significativa.

Alla luce di quanto detto si può provare a stimare l'ordine di grandezza della quantità di tempo impiegata per convertire un documento in formato `RASH` in formato `EPUB` utilizzando `rash2epub`. Trascurando casi pessimi, ovvero documenti che caricano da remoto un gran numero di immagini da convertire su macchine con connessioni di rete particolarmente lente, è possibile stimare il tempo di conversione nell'ordine dei secondi; nei casi pessimi, invece, il tempo di conversione potrebbe arrivare all'ordine dei minuti.

4.3 Ipotesi di workflow

Avendo descritto nel dettaglio il formato `RASH` e tutti i vantaggi in termini di accessibilità che il suo uso comporta ed avendo illustrato approfonditamente l'uso e l'implementazione dello strumento `rash2epub` e come esso preservi tali vantaggi, è ora necessario provare a strutturare un workflow per la scrittura degli articoli scientifici che permetta di usufruire di tutti i benefici visti finora.

Come ovvio un workflow così strutturato dovrà ruotare attorno alla scrittura di un documento in formato `RASH`. Il workflow si può schematizzare in tre fasi:

- l'editing del documento vero e proprio, che sarà definito anche *sorgente*;
- il rendering del sorgente;
- l'esportazione del risultato.

Per scrivere il sorgente di un documento in formato `RASH` è possibile utilizzare un qualunque editor di testi semplici scegliendolo tra un vasto numero di alternative: poiché `RASH` è implementato come un sottoinsieme di `XHTML`

editor con funzionalità pensate in modo specifico per l'editing di codice sorgente in questo linguaggio potrebbero offrire un supporto migliore attraverso l'implementazione di caratteristiche quali la gestione dell'indentazione o l'evidenziazione della sintassi con colori diversi (*syntax highlighting*) per favorirne la lettura.

Per la fase di rendering, invece, è possibile avvalersi dei browser più comuni. Il framework `RASH`, infatti, è strutturato in modo tale da poter funzionare correttamente all'interno di qualunque browser purché esso offra un buon supporto per i linguaggi `XHTML`, `CSS` e `JavaScript`, cosa che avviene nella maggior parte delle applicazioni di questo tipo essendo questi linguaggi utilizzati per la creazione dei contenuti Web ormai da moltissimo tempo. Visualizzare il rendering del documento in un'applicazione diversa da quella utilizzata per scriverlo potrebbe essere un inconveniente: diversi editor per documenti `HTML` e `XHTML`, perciò, consentono di visualizzare tale rendering all'interno della stessa applicazione avvalendosi degli engine utilizzati dai browser più comuni; a volte tali editor offrono persino un aggiornamento in tempo reale del rendering quando vengono apportate modifiche al sorgente. Poiché come detto più volte `RASH` usa proprio tecnologie per il Web, è possibile utilizzare tali editor per i documenti in questo formato senza alcun problema. Se si preferisse un approccio WYSIWYG, invece, al momento è possibile scrivere i documenti utilizzando il programma Microsoft Word seguendo alcune linee guida e convertirli in formato `RASH` tramite lo strumento `docx2rash` distribuito con il framework; analogamente è possibile utilizzare l'applicazione Writer del pacchetto Libre Office ed esportare il risultato in formato `RASH` tramite lo strumento `odt2rash` anch'esso distribuito nel framework. Inoltre è in fase di sviluppo un editor wysiwyg per documenti `RASH` che potrà funzionare all'interno del browser e, quindi, sarà disponibile per tutti i principali sistemi operativi.

Per la fase di esportazione del documento, infine, è possibile utilizzare diversi approcci. Il primo e più vantaggioso è sicuramente quello di utilizzare lo strumento `rash2epub`: in questo modo, come già visto in precedenza, si

otterrà un unico file contenente sia il documento RASH che tutte le risorse indispensabili per visualizzarlo correttamente indipendentemente dal dispositivo utilizzato. In alternativa un documento RASH può essere agevolmente pubblicato su un sito Web, assicurandosi che esso contenga anche le risorse necessarie al funzionamento del framework e alla corretta visualizzazione del documento. Qualora fosse strettamente necessario ottenere un file in formato PDF, per esempio a causa di vincoli imposti dagli editori per esigenze tipografiche, è infine possibile esportare un documento RASH in tale formato utilizzando gli strumenti integrati nel framework.

4.4 Discussione...

A questo punto non resta che discutere i benefici del workflow ipotizzato nel paragrafo 4.3 ed analizzarne alcuni dettagli non ancora approfonditi.

4.4.1 ...dell'editing del sorgente del documento RASH

Poiché il sorgente di un documento RASH è costituito da un documento di testo semplice che contiene sia il contenuto e i tag che definiscono la struttura semantica del documento, esso può essere modificato attraverso molteplici editor; l'accessibilità del sorgente, dunque, dipenderà fortemente da quella dell'editor utilizzato per scriverlo e leggerlo. Tuttavia, al contrario di quanto avveniva nel caso degli editor WYSIWYG, a meno di problemi di codifica dei caratteri non vi sono rischi di incompatibilità tra editor diversi e, perciò, si potrà utilizzare l'editor che risulta più accessibile. Oltre che per l'accessibilità, però, questo aspetto ha un'altra implicazione positiva: viene infatti agevolato l'editing dello stesso documento da parte di più persone; vista la non-ambiguità dei tag utilizzabili in un documento in formato RASH, infatti, oltre a regole relative ai contenuti sarà necessario concordare soltanto dettagli minori quali stile dell'indentazione del sorgente e l'encoding dei caratteri, anche se recentemente nei documenti XhTML si tende ad assumere l'encoding dei caratteri UTF8.

Il sottoinsieme di tag XHTML ammessi dalla grammatica del formato RASH è tale che il sorgente del documento, al contrario di quanto avviene nei workflow che utilizzano LaTeX, può essere letto agevolmente dagli utenti di screen reader; l'unico accorgimento riguarda l'immissione delle formule matematiche poiché, se queste sono immesse nel sorgente in MathML, la loro lettura del sorgente tramite uno screen reader può essere complicata. È possibile però aggirare il problema in due modi:

- utilizzando la sintassi ASCIIMath (fortemente consigliata) o LaTeX per la scrittura delle formule;
- leggendo la formula dal rendering del sorgente.

Va infine sottolineato che non è affatto obbligatorio editare direttamente il sorgente di un documento RASH manualmente; come visto nel paragrafo 4.3, infatti, è possibile generare documenti in questo formato anche utilizzando vari editor WYSIWYG; una volta convertiti tali documenti in formato RASH essi presenteranno gli stessi vantaggi dei documenti scritti "nativamente" in formato RASH.

4.4.2 . . . del rendering del sorgente

Come visto nel paragrafo 4.3 in questa fase è possibile visualizzare la rappresentazione grafica del documento generata dal framework RASH. Si potrebbe paragonare questa fase al typesetting di un documento LaTeX, ma sebbene concettualmente i due processi appaiano simili tecnicamente sono molto differenti; mentre il processo di typesetting restituisce un file di output completamente diverso dal sorgente, infatti, il rendering di un documento RASH costruisce dinamicamente la rappresentazione grafica del documento sorgente e, perciò, la rappresentazione grafica riflette sempre il contenuto del sorgente. Il rendering grafico generato dal framework RASH, inoltre, è perfettamente accessibile per gli utenti di screen reader poiché tutti i tag semantici vengono rappresentati seguendo i principi dettati dalle WCAG [Wor08b]; la gestione

delle formule matematiche tramite MathJax, poi, sfrutta tutti i meccanismi che la libreria offre per rendere accessibili agli utenti di screen reader.

Gli unici fattori che possono influire negativamente sull'accessibilità del rendering di un documento RASH, quindi, sono fattori esterni e non riconducibili al formato del documento; tra questi si trovano per esempio:

- il browser (o l'engine del browser usato da un'applicazione di altro tipo) per costruire il rendering;
- il supporto dello screen reader per le più recenti API di accessibilità;
- il supporto del sistema operativo per lam appatura dei più recenti standard per l'accessibilità delle pagine Web alle API di accessibilità.

L'utente ha il pieno controllo su tali fattori e, perciò, può scegliere liberamente di agire per risolvere i problemi causati da essi.

4.4.3 ... dell'esportazione del risultato

Resta infine da discutere la fase di esportazione del documento RASH prevista dal workflow ipotizzato. Come descritto nel paragrafo 4.3 essa può avvenire secondo tre modalità:

- la conversione in formato EPUB utilizzando lo strumento rash2epub;
- la pubblicazione del documento su un sito Web;
- l'esportazione del documento in formato PDF.

Il risultato ottenuto esportando il documento in formato EPUB utilizzando rash2epub sarà pienamente accessibile per gli utenti di tecnologie assistive e non presenterà particolari problemi di usabilità; inoltre, poiché all'interno dei file EPUB generati da rash2epub è presente l'intero framework RASH sarà possibile modificare dall'interno dell'ebook il layout da usare per la visualizzazione (ed eventualmente la stampa) del risultato. Poiché durante la lettura di un file EPUB generato da rash2epub viene di fatto eseguito il rendering del

documento RASH in esso contenuto, inoltre, valgono anche tutti i vantaggi esaminati a proposito del rendering di un documento RASH.

La visualizzazione di un documento RASH distribuito attraverso un sito Web, invece, è del tutto paragonabile alla visualizzazione dello stesso documento in locale; l'unica differenza sostanziale, infatti, è che sia il documento che il framework vengono caricati da remoto piuttosto che da percorsi locali. Ne deriva che valgono tutti i benefici visti a proposito dell'accessibilità della fase di rendering del documento.

È infine possibile esportare un documento RASH in formato PDF. In questo caso il risultato presenterà tutti i problemi di accessibilità tipici dei documenti PDF generati da LaTeX di cui si è ampiamente discusso nel capitolo 1; tale possibilità viene offerta per soddisfare particolari esigenze tipografiche che richiedano esplicitamente l'uso di questo formato. Pubblicare su Internet il risultato ottenuto esportando un documento RASH in formato PDF è quindi fortemente sconsigliato, poiché distribuire il documento in questo formato non presenta alcun vantaggio rispetto alla sua pubblicazione in formato EPUB o in formato originale, presentando al contrario molteplici problematiche. In ogni caso, qualora per qualsiasi motivo si necessiti di distribuire il formato PDF di un documento RASH è fortemente consigliato renderne disponibile una versione nei due formati visti in precedenza, inserendo all'interno del file PDF le indicazioni per ottenerla in modo che esse siano facilmente raggiungibili e ben visibili (per esempio inserendole nella prima pagina).

Capitolo 5

Conclusione e sviluppi futuri

Come si è visto nel capitolo 1 la situazione attuale relativa all'accessibilità degli articoli scientifici è piuttosto critica. Nella maggior parte dei casi, infatti, essi vengono distribuiti in formato `PDF` non rispettando gli standard di accessibilità previsti da questo formato per tutte le ragioni lungamente discusse in questo lavoro di tesi; come si è visto spesso si giunge a questo risultato anche utilizzando dei workflow che, invece, non presenterebbero tali problemi di accessibilità. Questa situazione genera molteplici difficoltà alle persone con disabilità che volessero leggere tali articoli ed "accedere" alle informazioni in essi contenute.

Per come sono stati concepiti fin dalle loro origini, sia il formato `RASH` che il framework ad esso correlato, costituiscono delle valide fondamenta per la produzione di articoli scientifici non solo accessibili, ma anche aderenti al movimento della cosiddetta "Linked Research" che mira a rendere il contenuto di tali articoli fruibile attraverso le tecnologie del "Web semantico". Con l'aggiunta dello strumento `rash2epub`, che consente di distribuire in un unico file i documenti in formato `RASH` e tutte le risorse necessarie per visualizzarli correttamente in modo molto agevole, è possibile creare un vero e proprio workflow per la produzione e la distribuzione degli articoli scientifici in tale formato. Tale workflow non solo può competere con quelli più tradizionali, ma anche aspirare in un certo senso a sostituirli; come descritto nel capito-

lo 4, i vantaggi che esso produce, infatti, sono tali e tanti da giustificarne l'adozione.

Ma lo strumento `rash2epub`, come ogni altro software, può essere sicuramente migliorato e, perciò, di seguito si esporranno alcune idee che potrebbero essere implementate per farlo. La maggior parte di tali miglioramenti riguarda aspetti non indispensabili per il corretto funzionamento dello strumento o, nel caso di quelli importanti, che non hanno un impatto significativo nella maggior parte dei casi in cui esso può essere utilizzato e non sono stati implementati per motivi temporali. Altri miglioramenti, invece, riguardano puramente la qualità del codice; altri, infine, sono relativi ad una maggiore integrazione di `rash2epub` nel resto del framework `RASH`.

Un primo possibile miglioramento di `rash2epub` riguarda la gestione della lingua del documento. Essa infatti deve essere dichiarata nei metadati del file `EPUB` generato; attualmente lo strumento tenta di ricavare il valore dal documento originale ma, se essa non è specificata, assume di default l'inglese. Sarebbe auspicabile, invece, fornire un'opzione per fare in modo che sia l'utente a selezionare il valore da utilizzare nella circostanza in cui la lingua del documento non sia dichiarata.

Un altro possibile miglioramento strettamente correlato al precedente riguarda la gestione della localizzazione di alcune stringhe inserite nel file `EPUB` generato qualora il documento `RASH` non contenga un valore appropriato per l'elemento in cui esse vengono utilizzate: è questo il caso della sezione delle note a piè di pagina che, secondo la grammatica di `RASH`, non richiede l'uso esplicito di un titolo per la sezione il quale invece è necessario per a corretta generazione dell'indice del file `EPUB`. Al momento `rash2epub` in questi casi utilizza delle stringhe in inglese "hard-coded" all'interno dei componenti del tool, poiché queste situazioni devono essere gestite durante l'applicazione del foglio di stile `XSL` e la localizzazione delle stringhe in questa tecnologia è un problema non banale da risolvere.

Un altro possibile miglioramento per il foglio di stile `XSL` utilizzato dallo strumento `rash2epub` consiste nell'annotare le variabili e le funzioni ausilia-

rie definite all'interno del foglio di stile con opportune dichiarazioni di tipo fornite dallo schema `XSD`. Tale miglioramento non avrebbe conseguenze visibili per l'utente finale, ma si rivelerebbe uno strumento molto utile per agevolare gli sviluppi futuri di `rash2epub`: durante l'applicazione del foglio di stile `XSL`, infatti, il processore di questo linguaggio potrebbe eseguire gli opportuni controlli sui tipi rispetto all'input fornito e segnalare gli eventuali errori.

Un ulteriore miglioramento che si potrebbe apportare allo strumento `rash2epub` riguarda l'implementazione di una gestione dei collegamenti ipertestuali presenti nel documento. Al momento, infatti, una volta generato il formato `EPUB` del documento originale saranno accessibili soltanto i collegamenti ipertestuali che puntano ad URL remoti assoluti; sarebbe auspicabile, invece, gestire anche i collegamenti ipertestuali che puntano a risorse locali. Tale gestione si potrebbe comportare in due modi diversi:

- aggiungere al file `EPUB` la risorsa a cui fa riferimento il collegamento ipertestuale e modificare la destinazione del collegamento, così che questa venga caricata dalla sua posizione all'interno del file;

- generare un messaggio informativo che avverta l'utente della presenza di un tale collegamento ipertestuale.

La prima implementazione sarebbe ovviamente più complessa; occorrerebbe infatti copiare la risorsa, ma anche verificare che essa possa essere effettivamente inclusa in un ebook in formato `EPUB` secondo lo standard e generare i metadati appropriati. La seconda implementazione, invece, sarebbe ovviamente più semplice e, in ogni caso, permetterebbe di informare l'utente della presenza di tali collegamenti ipertestuali che potrebbero non funzionare se cliccati dall'interno del file `EPUB` generato.

Altri possibili miglioramenti per lo strumento `rash2epub`, infine, riguardano la gestione della libreria `MathJax`. Il primo di questi consiste nel verificare se sia possibile ridurre ulteriormente il footprint della stessa all'interno dei file `EPUB` generati; per farlo occorrerebbe analizzare in dettaglio il co-

dice sorgente della libreria, comprenderne approfonditamente l'architettura e, probabilmente, crearne una versione "ad-hoc" con modifiche sostanziali. In realtà, non è detto che un miglioramento del genere sia possibile, vista la strategia già adottata per minimizzare il footprint della libreria descritta nel paragrafo 3.3, ma allo stato attuale non è nemmeno possibile escluderne con certezza la fattibilità. Il secondo miglioramento relativo alla gestione di `MathJax`, invece, consiste nell'includere la libreria all'interno del file `EPUB` solo ed esclusivamente se il documento originale contiene almeno una formula matematica al contrario di quanto accade attualmente, visto che `rash2epub` aggiunge la libreria ai file generati indipendentemente da ciò. Implementare questo miglioramento richiede alcune modifiche sia al foglio di stile `XSL` che allo script `BASH`.

Un altro aspetto su cui si potrebbero concentrare futuri miglioramenti è l'integrazione dello strumento `rash2epub` con il resto del framework `RASH`. Attualmente, infatti, `rash2epub` può essere invocato soltanto a riga di comando passando come parametri il nome del file da generare ed il percorso del documento da convertire; tuttavia, l'usabilità dello strumento sarebbe decisamente migliore se, per esempio, fosse possibile invocarlo direttamente dalla visualizzazione del rendering di un documento `RASH` all'interno di un browser cliccando un apposito pulsante. Tale pulsante potrebbe essere aggiunto alla barra degli strumenti inserita alla fine di ogni documento dal framework in cui si trovano, per esempio, le statistiche (numero di parole, numero di figure, numero di listati, numero di formule) sul documento ed i pulsanti per modificare il layout con cui esso viene visualizzato.

Avendo discusso approfonditamente la problematica dell'(in)accessibilità degli articoli scientifici sul Web, fornito una panoramica prima e descritto l'implementazione poi per lo strumento `rash2epub`, avendo ipotizzato un possibile workflow che ne faccia uso ed espresso valutazioni da diversi punti di vista sullo stesso e sugli strumenti da esso utilizzati ed avendo infine descritto alcuni possibili sviluppi futuri al riguardo, è possibile formulare una conclusione per questo lavoro di tesi. E per farlo si intende portare il letto-

re ad una riflessione su un aspetto non propriamente tecnico del problema. L'inaccessibilità degli articoli scientifici sul Web, infatti, può essere considerata da un punto di vista molto pragmatico oltre che da quello tecnico: la pubblicazione di un articolo, infatti, è per sua natura mirata a raggiungere il maggior numero di persone possibile per informarle a proposito del proprio lavoro di ricerca o per portarle a conoscenza di informazioni che si ritengono importanti. Stando al "rapporto sulla disabilità nel mondo" [Wor11] pubblicato nel 2011 dall'Organizzazione Mondiale della Sanità (OMS), le persone affette da disabilità nel mondo sono oltre un miliardo, circa il 15% della popolazione mondiale; pubblicando un articolo scientifico inaccessibile tutte queste persone vengono potenzialmente escluse dal target che potrà leggerlo e, perciò, si sta automaticamente limitando l'impatto dell'articolo. L'accessibilità degli articoli scientifici, come del resto l'accessibilità delle informazioni in generale, dovrebbe essere dunque un interesse di tutti e non solo delle persone con disabilità che lottano per vedere riconosciuto quello che, se ci si pensa, è soltanto un proprio diritto. Il diritto di sapere, di poter aver accesso alla cultura; un diritto talmente importante che viene persino riconosciuto dalla convenzione dei diritti delle persone con disabilità adottata dall'Organizzazione delle Nazioni Unite (ONU).

Appendici

Appendice A

Dati sull'utilizzo degli screen reader

Il grafico mostrato nella figura 1.1 rappresenta i dati relativi agli screen reader più utilizzati raccolti nel sondaggio *Screen Reader Survey* condotto periodicamente da *WebAIM*, nota società impegnata da vari anni nel settore dell'accessibilità. Tali dati sono riportati nella tabella che segue in modo da renderli accessibili anche a chi non potesse "guardare l'immagine".

Screen Reader	# of Respondents	% of Respondents
JAWS	1098	43.7%
NVDA	1040	41.4%
VoiceOver	778	30.9%
Window-Eyes	745	29.6%
ZoomText	691	27.5%
System Access or System Access To Go	173	6.9%
ChromeVox	71	2.8%
Other	163	6.5%

Tabella 1.1: Dati sull'utilizzo degli screen reader più comuni.

Fonte: `http://webaim.org/projects/screenreadersurvey6/`

Appendice B

Dati sulle ricerche di documenti in vari formati

Il grafico mostrato nella figura 1.2 rappresenta i dati relativi alle ricerche di documenti nei formati più diffusi diversi da `HTML` e `XHTML` rilevati attraverso Google Trends in varie date. Tali dati sono riportati nella tabella 2.1 in modo da renderli accessibili anche a chi non potesse "guardare l'immagine".

Data detection time	`PDF`	`DOCx`	`XLSx`	`PPTx`	`EPUB`	`ODx`	`TXT/RTF`
	81%	13%	3%	3%	?	?	?
	86%	10%	3%	1%	?	?	?
	79%	17%	2%	1%	?	?	?
	77.3%	5.5%	6.0%	6.1%	1.4%	0.8%	2.9%
	71.7%	16.1%	1.8%	1.6%	1.6%	1.0%	6.3%

Tabella 2.1: La tabella mostra i dati sulle ricerche di documenti in vari formati in percentuale.

Fonte: `http://duff-johnson.com/2015/02/12/the-8-most-popular-document-formats-on-the-web-in-2015/`

Appendice C

Schermata d'aiuto di `rash2epub`

La figura 2.1 contiene uno screenshot che mostra l'output che si ottiene invocando lo strumento `rash2epub` passando il flag "-h" in un terminale, ovvero "chiedendo" di visualizzare le informazioni sull'uso dello strumento. Il contenuto di tale screenshot viene riportato di seguito in formato testuale, così da renderlo comprensibile anche a chi non potesse "guardare l'immagine".

```
MacBook-Air-di-Vincenzo:~ vincenzo$ cd rash/tools/rash2epub/bin
MacBook-Air-di-Vincenzo:bin vincenzo$
MacBook-Air-di-Vincenzo:bin vincenzo$ ./rash2epub -h
Creates the EPUB version of a RASH document.


Usage:

rash2epub -o outputfile.epub document_path, where
outputfile.epub is the path of the generated EPUB file
document_path is the path of the RASH document you want to generate the EPUB vers
```

Appendice D

Diagramma di sequenza per rash2epub

Il diagramma mostrato nella figura 3.1 è un diagramma di sequenza UML che è stato realizzato tramite il software PlantUML, uno strumento che consente di realizzare diagrammi UML a partire da semplici descrizioni testuali; il tool interpreta la descrizione testuale del diagramma fornita dall'utente e disegna il diagramma ad essa corrispondente, salvando il risultato in un'immagine nel formato richiesto. Di seguito è riportata la descrizione testuale del diagramma mostrato in figura 3.1; tale descrizione consente di comprendere il contenuto del diagramma anche a chi non potesse "guardare l'immagine".

```
@startuml
activate RASH2EPUB
RASH2EPUB -> OriginDocument: Load
activate originDocument

alt Origin Document contains Math formulas
    activate Math2MathML
    RASH2EPUB -> Math2MathML: "Convert Math formulas to MathML"
    deactivate Math2MathML
end
```

```
RASH2EPUB -> Saxon: Load conversion XSL
activate Saxon
Saxon -> OriginDocument: Apply conversion XSL
deactivate Saxon

RASH2EPUB -> EPUBFile: Initialize
create EPUBFile

loop For each image embedded in the OriginDocument
    RASH2EPUB -> EPUBFile: embed image
end

deactivate OriginDocument

RASH2EPUB -> EPUBFile: Embed MathJax
RASH2EPUB -> EPUBFile: save

deactivate RASH2EPUB

@enduml
```

Bibliografia

[ACI+14] Tiziana Armano, Anna Capietto, Marco Illengo, Nadir Murru, and Rosaria Rossini. An overview on ict for the accessibility of scientific texts by visually impaired students. *Atti del convegno SIREM–SIE–L, Perugia, Italy*, pages 119–122, 2014.

[Ame98] U.S. Rehabilitation Act Amendments. Section 508, 1998. URL: `http://www.webaim.org/standards/508/checklist`.

[BZB15] Erin Brady, Yu Zhong, and Jeffrey P Bigham. Creating accessible pdfs for conference proceedings. In *Proceedings of the 12th Web for All Conference*, page 34. ACM, 2015.

[Cer12] Davide Cervone. Mathjax: a platform for mathematics on the web. *Notices of the AMS*, 59(2):312–316, 2012.

[CKS16] Davide Cervone, Peter Krautzberger, and Volker Sorge. Employing semantic analysis for enhanced accessibility features in mathjax. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1129–1134. IEEE, 2016.

[Cli] Andy Clifton. Latex accessibility_meta package. URL: `https://github.com/NREL/latex_editing/blob/master/documents/accessibility-meta.sty`.

[DE12] Olaf Drümmer and Markus Erle. Pdf/ua–a new era for document accessibility: Understanding, managing and implementing the iso

standard pdf/ua (universal accessibility): Introduction to the special thematic session. In *International Conference on Computers for Handicapped Persons*, pages 585–586. Springer, 2012.

[DM99] Nikos Drakos and Ross Moore. The latex2html translator, March 1999.

[Drü12] Olaf Drümmer. Pdf/ua (iso 14289-1)–applying wcag 2.0 principles to the world of pdf documents. In *International Conference on Computers for Handicapped Persons*, pages 587–594. Springer, 2012.

[Fre] Free Document Foundation. [META] Accessibility (a11y) bugs and enhancements. URL: `https://bugs.documentfoundation.org/show_bug.cgi?id=101912`.

[Fre04] Free Software Foundation, INC. sed a stream editor, 2004. URL: `https://www.gnu.org/software/sed/manual/sed.html`.

[Goo] Google, INC. Edit documents with a screen reader - Docs editors Help. URL: `https://support.google.com/docs/answer/1632201?hl=en`.

[Gru04] John Gruber. Markdown, 2004. URL: `https://daringfireball.net/projects/markdown/`.

[Gur04] Eitan M Gurari. Tex4ht: Html production. *TUG-boat*, 25(1):39–47, 2004.

[IEE16] IEEE. Standard for information technology–portable operating system interface (posix(r)) base specifications, issue 7. *IEEE Std 1003.1, 2016 Edition (incorporates IEEE Std 1003.1-2008, IEEE Std 1003.1-2008/Cor 1-2013, and IEEE Std 1003.1-2008/Cor 2-2016)*, pages 3216–3225, Sept 2016. `doi:10.1109/IEEESTD.2016.7582338`.

[Int] International Digital Publishing Forum (IDPF). EpubCheck a validation tool for EPUB. URL: `https://github.com/IDPF/epubcheck`.

[Int14a] International Digital Publishing Forum (IDPF). EPUB Content Documents 3.0.1, June 2014. URL: `http://www.idpf.org/epub/301/spec/epub-contentdocs.html`.

[Int14b] International Digital Publishing Forum (IDPF). EPUB Publications 3.0.1, June 2014. URL: `http://www.idpf.org/epub/301/spec/epub-publications.html`.

[Int14c] International Organization for Standardization (ISO). ISO 14289-1:2014 - Document management applications – Electronic document file format enhancement for accessibility – Part 1: Use of ISO 32000-1 (PDF/UA-1). Standard, International Organization for Standardization (ISO), 2014. URL: `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64599&ICS1=35&ICS2=240&ICS3=30`.

[Int14d] International Organization for Standardization (ISO). ISO 32000-1:2008 - Document management – Portable document format – Part 1: PDF 1.7. Standard, International Organization for Standardization (ISO), 2014. URL: `http://www.iso.org/iso/catalogue_detail.htm?csnumber=51502`.

[Kay15] Michael H. Kay. SAXON, the XSLT and XQuery processor, 2015. URL: `http://saxon.sourceforge.net`.

[Kra13] Peter Krautzberger. MathML forges on the standard for mathematical content in publishing work flows, technical writing, and math software, November 2013. URL: `http://radar.oreilly.com/2013/11/mathml-forges-on.html`.

[LAKM07] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of human-computer interaction*, 22(3):247–269, 2007.

[Lam94] Leslie Lamport. *Latex*. Addison-Wesley, 1994.

[lat] LaTeX. Wikibooks, open books for an open world. URL: `https://en.wikibooks.org/wiki/LaTeX`.

[LB15] Tim T. Y. Lin and Graham Beales. ScholarlyMarkdown Syntax Guide, January 2015. URL: `http://scholarlymarkdown.com/Scholarly-Markdown-Guide.html`.

[Mac] John MacFarlane. Pandoc: a universal document converter. URL: `http://pandoc.org`.

[Mat16] MathJax. MathJax-grunt-cleaner a grunt file to reduce the footprint of a mathjax installation, 2016. URL: `https://github.com/mathjax/MathJax-grunt-cleaner`.

[MGB+04] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX companion*. Addison-Wesley Professional, 2004.

[Mil16] Bruce Miller. Latexml: Alatex to xml converter, September 2016. URL: `http://dlmf.nist.gov/LaTeXML/,seenSeptember2006`.

[Moo09] Ross Moore. Ongoing efforts to generate "tagged pdf" using pdftex. *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*, pages 125–131, 2009.

[MPR+17] Silvia Mirri, Silvio Peroni, Vincenzo Rubano, Paola Salomoni, and Fabio Vitali. Towards accessible graphs in HTML-based

scholarly articles. In *14th IEEE Annual Consumer Communications & Networking Conference, CCNC 2017*, 2017. Accepted for publication.

[Oet13] Alexandra Oettler. *PDF/UA in a nutshell*. PDF Association, August 2013. URL: `http://www.pdfa.org/publication/pdfua-in-a-nutshell/`.

[Ore] Oreilly Media. HTMLBook: Let's write books in HTML. URL: `https://github.com/oreillymedia/HTMLBook/`.

[Par04] Italian Parliament. Law nr. 4 – 01/09/2004 the stanca act, January 2004. URL: `http://www.camera.it/parlam/leggi/04004l.htm`.

[Par15] Thomas Park. pubcss: Format academic publications in HTML & CSS, January 2015. URL: `https://github.com/thomaspark/pubcss/`.

[Per16] Silvio Peroni. RASH: Research Articles in Simplified HTML Documentation - Version 0.5, 2016. URL: `https://rawgit.com/essepuntato/rash/master/documentation/index.html`.

[PODI+16] Silvio Peroni, Francesco Osborne, Angelo Di Iorio, Andrea Giovanni Nuzzolese, Francesco Poggi, Fabio Vitali, and Enrico Motta. Research Articles in Simplified HTML: a Web-first format for HTML-based scholarly articles. Technical report, PeerJ Preprints, 2016. URL: `https://peerj.com/preprints/2513`.

[Ram97] TV Raman. Emacspeak-an audio desktop. In *Compcon'97. Proceedings, IEEE*, pages 229–231. IEEE, 1997.

[sci15] science.ai. scienceai/scholarly.vernacular.io: A vernacular of HTML for scholarly publishing, 2015. URL: `https://github.com/scienceai/scholarly.vernacular.io`.

[Sha16] Ather Sharif. evographs - a jQuery Plugin to create Web Accessible graphs. In *13th IEEE Annual Consumer Communications & Networking Conference, CCNC 2016*, 2016.

[SLW15] Volker Sorge, Mark Lee, and Sandy Wilkinson. End-to-end solution for accessible chemical diagrams. In *Proceedings of the 12th Web for All Conference*, page 6. ACM, 2015.

[SM14] Adam Spencer and Karen McCall. A strategic approach to document accessibility: Integrating pdf/ua into your electronic content. In *International Conference on Computers for Handicapped Persons*, pages 202–204. Springer, 2014.

[SMRF+] Peter Sefton, Peter Murray-Rust, Martin Fenner, Brian McMahon, Sam Adams, Dan Hagon, Claudia Koltzemburg, and Egon Willighagen. Scholarly HTML, Guidelines and tools for web based research and scholarship. URL: `http://scholarlyhtml.org`.

[Soi15] Neil Soiffer. Browser-independent accessible math. In *Proceedings of the 12th Web for All Conference*, page 28. ACM, 2015.

[Sor16] Volker Sorge. Supporting visual impaired learners in editing mathematics. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 323–324. ACM, 2016.

[UBR14] Andreas Uebelbacher, Roberto Bianchetti, and Markus Riesch. PDF Accessibility Checker (PAC 2): The First Tool to Test PDF Documents for PDF/UA Compliance. In *International Conference on Computers for Handicapped Persons*, pages 197–201. Springer, 2014.

[Wika] Wikipedia. EPUB - From Wikipedia, the free encyclopedia. URL: `https://en.wikipedia.org/wiki/EPUB`.

[Wikb] Wikipedia. Portable Document Format - From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Portable_Document_Format.

[Wor] World Wide Web Consortium (W3C). RDFa Core 1.1 - Third Edition syntax and processing rules for embedding rdf through attributes. W3C Recommendation 17 March 2015. URL: http://www.w3.org/TR/rdfa-syntax/.

[Wor02] World Wide Web Consortium (W3C). XHTML™ 1.0 - The Extensible HyperText Markup Language (Second Edition), 2002. W3C Recommendation 26 January 2000, revised 1 August 2002. URL: https://www.w3.org/TR/xhtml1/.

[Wor07a] World Wide Web Consortium (W3C). Understanding WCAG 2.0, a guide to understanding and implementing Web Content Accessibility Guidelines 2.0, 2007. URL: https://www.w3.org/TR/UNDERSTANDING-WCAG20/.

[Wor07b] World Wide Web Consortium (W3C). XSL Transformations (XSLT) Version 2.0, 2007. W3C Recommendation 23 January 2007, revised 1 August 2002. URL: https://www.w3.org/TR/xhtml1/.

[Wor08a] World Wide Web Consortium (W3C). Techniques for WCAG 2.0, techniques and failures for Web Content Accessibility Guidelines 2.0, 2008. URL: https://www.w3.org/TR/WCAG20-TECHS/.

[Wor08b] World Wide Web Consortium (W3C). Web content accessibility guidelines (wcag) 2.0, 2008. URL: http://www.w3.org/TR/WCAG20/.

[Wor11] World Health Organization (WHO). World report on disability, 2011. URL: http://www.who.int/disabilities/world_report/2011/report/en/.

[Wor14a] World Wide Web Consortium (W3C). Accessible Rich Internet Applications (WAI-ARIA) 1.0, 2014. W3C Recommendation 20 March 2014. URL: https://www.w3.org/TR/wai-aria/.

[Wor14b] World Wide Web Consortium (W3C). JSON-LD 1.0 - A JSON-based Serialization for Linked Data, 2014. W3C Recommendation 16 January 2014. URL: http://www.w3.org/TR/json-ld/.

[Wor14c] World Wide Web Consortium (W3C). Mathematical Markup Language (MathML) Version 3.0 2nd Edition, 2014. W3C Recommendation 10 April 2014. URL: https://www.w3.org/TR/MathML/.

[Wor14d] World Wide Web Consortium (W3C). RDF 1.1 Turtle terse rdf triple language, 2014. W3C Recommendation 25 February 2014. URL: https://www.w3.org/TR/turtle/.

[Wor14e] World Wide Web Consortium (W3C). RDF 1.1 XML Syntax, 2014. W3C Recommendation 17 February 2014. URL: http://www.w3.org/TR/rdf-syntax-grammar/.

[Wor15] World Wide Web Consortium (W3C). CSS Snapshot 2015, 2015. W3C Working Group Note, 13 October 2015. URL: https://www.w3.org/TR/2015/NOTE-css-2015-20151013/.

[Wor16] World Wide Web Consortium (W3C). HTML 5.1 - W3C Recommendation, November 2016. URL: https://www.w3.org/TR/html/.

Ringraziamenti

Desidero a questo punto ringraziare tutti coloro che hanno contribuito in qualche modo alla realizzazione di questo lavoro di tesi, prendendo in considerazione anche quanti hanno reso possibile un sereno svolgimento del mio percorso di studi universitario in questi tre anni. Senza il loro apporto, infatti, probabilmente questo lavoro di tesi non esisterebbe.

Ringrazio innanzitutto il relatore prof. Fabio vitali ed i correlatori prof.ssa Silvia Mirri e prof. Silvio Peroni che con i loro suggerimenti ed indicazioni tecniche mi hanno permesso di migliorare notevolmente la qualità di questa tesi.

Ringrazio poi tutto il personale della segreteria didattica e dell'ufficio disabili che in questi anni mi hanno supportato nel superare le inaccessibilità che inevitabilmente un percorso di studi universitario comporta per un non vedente; sembra assurdo, ma ancora nel 2016 la burocrazia ne crea davvero tante. Senza il loro apporto questi tre anni non sarebbero trascorsi così serenamente.

Ringrazio inoltre tutti i docenti del corso di laurea in informatica; la presenza di un non vedente in aula spesso "costringe" a riconsiderare vari aspetti quali l'accessibilità del materiale didattico e delle stesse lezioni; Ho sempre trovato grande disponibilità da parte di tutti nel comprendere le esigenze al riguardo, anche quando è stato necessario reperire materiali in formati alternativi (per esempio sorgenti LaTeX invece degli onnipresenti `PDF`) e mettere in discussione aspetti complessi (come per esempio l'individuazione di rappresentazioni testuali adeguate per alberi e grafi). Senza questa disponi-

bilità tutt'altro che scontata il mio percorso di studi sarebbe stato molto più complesso!

Ma la qualità di un mobile si comprende osservando la qualità del pannello posteriore, quello che nessuno mai pensa di guardare. Perciò ritengo doveroso ringraziare i miei genitori Carmen e Alfonso che, fin dai miei primi anni di vita, mi hanno permesso di crescere con la convinzione che avere una disabilità non fosse una vergogna e che molti dei problemi comportati dalla cecità spesso possono essere aggirati. Se non fossi cresciuto con questi valori e queste convinzioni forse oggi non avrei potuto raggiungere tanti traguardi e superare tante sfide per me così importanti, compresa questa laurea.

Ed ovviamente non mi sono dimenticato di mio fratello Manuel, che pur essendo arrivato 5 anni dopo di me ha subito compreso "come stavano le cose" e interiorizzato questi valori! E nemmeno di Raffaella, Eliana, della signora Enza, delle maestre, dei professori e di tutte le altre figure che mi hanno supportato nel mio percorso di studi precedente.

Infine permettetemi di ringraziare tutti coloro che non hanno esitato a giudicarmi come un pazzo quando ho detto loro di voler studiare informatica all'università pur essendo cieco, sostenendo che volevo fare una cosa semplicemente impossibile. A loro è dedicato questo lavoro; perché evidentemente ero non solo tanto pazzo per volerci provare nonostante i loro pareri, ma anche per riuscirci. O forse i loro pareri non erano poi così fondati. A loro è dedicato questo lavoro di tesi con la speranza che le prossime volte possano esprimere giudizi più oculati, specialmente quando ciò sia richiesto dalla loro posizione sociale e professionale.