- Obiettivi: Comprendere cosa sia un documento elettronico accessibile. Come funziona, livelli logici e tecnologici, formati.
- Target: operatori tecnici e redattori
- Prerequisiti: nessuno
- Durata: 1 giorno
- Attrezzatura necessaria: un browser, Acrobat Reader, Acrobat Pro, Word, Active Accessibility 2.0 SDK Tools, Accessibility Probe, NDVA, CrossEyes, qad_doc2xml

L’accessibilità dei documenti elettronici

Cos’è un documento elettronico accessibile?

Innanzitutto, chiariamo la parola “documento”. Con documento si intende un file che possiede un contenuto, una struttura, una semantica e che può essere in relazione con altri documenti.
Un documento elettronico è un documento la cui rappresentazione fisica è in forma di bit all'interno di un sistema informatico.

Esempi di documenti:

- Testi debolmente strutturati: romanzi, racconti, poesie, saggi, articoli, eccetera.
- Testi fortemente strutturati: elenchi telefonici, schede cliniche, manuali, eccetera.
- Ipertesti: testi contenenti collegamenti interni che ne permettono una lettura non sequenziale (vedi più avanti).
- Non-testi: immagini, schemi, progetti, fotografie, filmati, animazioni, eccetera.

Un documento elettronico accessibile dovrà poter essere fruito da chiunque, incluse le persone portatrici di disabilità.
Poiché non è possibile conoscere a priori come sarà composta la platea degli utenti del documento, è necessario creare documenti che possiedano determinate caratteristiche che ne permettano la fruizione sulla più ampia gamma possibile di periferiche. Queste caratteristiche possono essere così riassunte:

1. La struttura logica del documento deve essere definita utilizzando stili di paragrafo che caratterizzino semanticamente i contenuti a cui vengono attribuiti.
2. L’ordine di lettura deve essere preservato anche quando il testo eventualmente suddiviso in blocchi o in colonne venga presentato in modo linearizzato.
3. Il documento deve essere dotato di un sommario navigabile che permetta il collegamento diretto ai corrispondenti contenuti e prevedere idonei collegamenti ipertestuali per il ritorno all’indice o ai contenuti alla fine di ciascuna sezione.
4. Gli elementi informativi a corredo del testo, tra i quali note e relativi rimandi e riquadri di approfondimento, devono essere dotati di collegamenti ipertestuali espliciti al punto o all'elemento corrispondente nel testo principale.
5. Evitare di utilizzare immagini o altri elementi grafici per rappresentare contenuti testuali.
6. Dotare le immagini, i grafici e le tabelle di didascalie esaurienti, che forniscano informazioni equivalenti commisurate alla funzione esercitata dall’oggetto originale nello specifico contesto.
7. Collegare esplicitamente le didascalie all’immagine a cui si riferiscono tramite numerazione sequenziale contestualizzata all’organizzazione del documento.
8. Garantire che i contenuti sottoposti a ingrandimento siano visualizzati nel rispetto dell’ordine di presentazione originale ed evitare che per la loro lettura si debba ricorrere alla barra di scorrimento orizzontale del programma di lettura utilizzato.

Purtroppo, i documenti elettronici di solito non possiedono una struttura definita utilizzando gli stili di paragrafo, non sono disponibili testi alternativi per le immagini, sommari ed indici non sono navigabili e non possono essere definiti “accessibili”. Questo accade di solito perché non si sa che è possibile fare diversamente e perché non si pensa all'accessibilità fin dalle fasi di progetto iniziale. I contenuti vengono disposti nei layout pensando esclusivamente alla “carta”, e in qualche caso ne risultano impaginazioni così complesse e confuse da essere di difficile fruizione anche nel formato di distribuzione tradizionale.
L’accessibilità obbliga ad agire su due livelli: redazionale e tecnico. Nella Parte Terza vedremo come sia possibile rispondere ai requisiti su esposti praticamente.
Ricordiamo però che in un flusso di lavoro tradizionale, dove il testo viene realizzato probabilmente con Word e poi importato in un programma di impaginazione dove viene assemblato in gabbie predefinite e vengono aggiunte immagini ed elementi accessori, per ora il lavoro di verifica ed eventuale correzione degli errori di accessibilità dovrà essere svolto sul file finale, sia che si tratti di un output in PDF sia in XHTML. La trasparenza e l’interoperabilità fra i programmi migliorano di versione in versione, ma non sempre il lavoro tecnico di accessibilità svolto, per esempio, in Word viene rispettato all’importazione del file nel programma di impaginazione.
Analogamente, il file preparato con cura nel programma di impaginazione non sempre viene accuratamente trasposto nel corrispondente formato di output: alcune caratteristiche si perdono, ed è necessario intervenire sul file di output.
In ogni caso, la situazione migliora vistosamente di versione in versione verso una più completa interoperabilità.
A questo punto è lecito chiedersi “D’accordo, ma come li realizzo questi documenti? Che strategie posso adottare per ottenere il massimo risultato senza dover ricorrere a ulteriori lavorazioni? Questa situazione mi obbliga a lavorare sul formato finale, e sappiamo che questa è la situazione più onerosa. Come ovviare?”.

Problematiche generali

L’accessibilità dei documenti elettronici si svolge su più livelli, anche contemporaneamente.
Come spiegato nella Parte Prima, l’accessibilità non riguarda esclusivamente le disabilità fisiche permanenti, ma tutta una serie di situazioni che sono causa di limitazioni più o meno gravi della fruizione dei contenuti.
Se ci limitiamo alla “tecnica” (o al mugugno: XPress non mi esporta i testi alternativi, Indesign mi perde certe informazioni, cosa lo faccio a fare, e così via), stiamo cogliendo soltanto un parziale aspetto dell’accessibilità, e forse nemmeno il più importante visto che come già detto la trasparenza e l’interoperabilità migliorano di versione in versione dei programmi. L’accessibilità si svolge su più piani, che concorrono sinergicamente alla qualità ed economicità del formato finale di distribuzione prescelto.

Livelli di azione dell’accessibilità

Il primo livello di azione è progettuale, non ha nulla a che fare con la tecnologia e agisce direttamente sui contenuti. Progettare con l’accessibilità in mente presuppone la conoscenza di ciò che accadrà al termine del flusso di lavoro. Se utilizzare un particolare layout produrrà un florilegio di box slegati dal flusso principale del testo, senza alcun rimando esplicito (esempio tipico, tanti libri scolastici magari appaganti esteticamente ma incomprensibili da un punto di vista cognitivo, perché non si capisce quando e come quel box debba essere letto, quale sia la sua relazione con il testo principale) e magari affidando l’informazione al solo colore (quel bel fondo in tinta pastello potrebbe essere invisibile a chi soffre di cecità ai colori), quel testo sarà davvero difficile da rendere accessibile, qualsiasi tecnologia si utilizzi e indipendentemente dalla tipologia di utente.
È necessario comprendere con precisione come funzionano le tecnologie assistive, e capire per quale motivo determinate richieste (per esempio, la necessità di strutturare con gli stili di paragrafo i documenti) siano così importanti per ottenere documenti elettronici accessibili. Non è soltanto una

richiesta di “buone regole di redazione” (livello logico), ma una necessità pratica vera e propria (livello tecnologico).
Dicevamo “*separare contenuti e loro struttura dagli aspetti presentazionali*”. I contenuti devono essere fruibili indipendentemente dal loro aspetto grafico. Perché?
Ci concentreremo ora su alcuni aspetti tecnici, che non è importante comprendere a fondo dal punto di vista programmazione, ma che certamente saranno di aiuto nel capire “perché funziona”.
Il tentativo è capire dove la corretta progettazione si unisce sinergicamente alla tecnologia e come questa comprensione produca e migliori il flusso di lavoro fino alla sua conclusione con benefici sia sulla qualità dei contenuti sia sul versante economico del lavoro di editing.
Il livello di azione tecnologico si produce lato utente, coinvolge le periferiche utilizzate e quindi dipende dalla tecnologia. Osserviamo cosa accade per esempio in un sistema operativo molto diffuso, Windows, che da questo punto di vista è anche il più avanzato (anche se Mac OS X e Linux stanno procedendo molto velocemente a un rispettivo adeguamento sul versante accessibilità)

Esercizio 1: scaricare gli Active Accessibility 2.0 SDK Tools dal sito Microsoft (http://www.microsoft.com/downloads/details.aspx?familyid=3755582a-a707-460a-bf21-1373316e13f0&displaylang=en) e prendere confidenza con gli eventi generati dalle MSAA in Windows osservando cosa accade quando si eseguono semplici operazioni in vari programmi.

Esercizio 2: Leggere e commentare le specifiche di accessibilità di Windows a http://technet.microsoft.com/en-us/library/bb457128.aspx#mainSection, con particolare attenzione alla tabella Table H-2 Common User Difficulties and Solutions.

Esercizio 3: leggere e commentare notizie selezionate sull’accessibilità di Mac OS su http://atmac.org/

Esercizio 4: leggere e commentare notizie selezionate sull’accessibilità di Linux a http://larswiki.atrc.utoronto.ca/wiki

Leggendo le varie specifiche dei sistemi operativi, è semplice comprende come l’interfacciamento fra tecnologie assistive e programmi avvenga perché i sistemi operativi prevedono le necessarie librerie e puntatori agli elementi dell’interfaccia.
Per esempio, in Windows l’interazione fra un pulsante e uno screen reader provocherà la seguente cascata di eventi: il client (lo screen reader) interroga le Microsoft Active Accessibility per trovare un puntatore all’oggetto **IAccessible** con il metodo `AccessibleObjectFromWindow`, `AccessibleObjectFromEvent`, e così via. Microsoft Active Accessibility chiede al server (il pulsante) un puntatore all’implementazione di **IAccessible**. Poiché si tratta di un controllo standard e non possiede una propria implementazione di **IAccessible**, viene restituito il valore 0. In questo caso, Microsoft Active Accessibility per restituire metodi e valori della proprietà usa un proxy. Per `IAccessible::get_accName`, Microsoft Active Accessibility restituisce il valore di `GetWindowText()`.
Questa complessa tecnologia agisce sia sull’interfaccia sia sui documenti. Per esempio, nel caso di una tabella l’oggetto IAccessible presenta una gerarchia così fatta:

```
ROLE_SYSTEM_TABLE
  |- ROLE_SYSTEM_ROW
  |   |- ROLE_SYSTEM_ROWHEADER
  |   |- ROLE_SYSTEM_COLUMNHEADER
  |   |- ROLE_SYSTEM_COLUMNHEADER
  |   |- ROLE_SYSTEM_COLUMNHEADER
  |       '- ..
  |- ROLE_SYSTEM_ROW
  |   |- ROLE_SYSTEM_ROWHEADER
  |   |- ROLE_SYSTEM_CELL
```

```
|   |- ROLE_SYSTEM_CELL
|   |- ROLE_SYSTEM_CELL
|    '- ..
|- ROLE_SYSTEM_ROW
|   |- ROLE_SYSTEM_ROWHEADER
|   |- ROLE_SYSTEM_CELL
|   |- ROLE_SYSTEM_CELL
|   |- ROLE_SYSTEM_CELL
|    '- ..
 '- ..
```

Un oggetto **Table** è una collezione di uno o più oggetti **Row**, dove ciascun oggetto **Row** è il genitore di uno o più oggetti figli. Un oggetto **Row** può contenere uno o più oggetti **Cell**. Tuttavia, tutti gli oggetti **Row** devono contenere lo stesso numero di oggetti **Cell**. Anche se nella gerarchia di **IAccessible** della tabella non viene mostrato, un oggetto **Cell** è a sua volta un elemento che può contenere una gerarchia arbitrariamente profonda di oggetti **IAccessible** con qualsiasi ruolo valido. Per esempio, una tabella annidata viene rappresentata posizionando un oggetto **Table** all'interno di un oggetto **Cell**. I metodi e le proprietà di IAccessibile sono moltissime, e non siamo dei programmatori. Chi volesse approfondire l'argomento può trovare tutte le informazioni del caso all'URL http://msdn.microsoft.com/en-us/library/ms971325.aspx.
In questo momento, l'obiettivo è rendere chiaro perché le tecnologie assistive riescono a interpretare i documenti elettronici.
Nel caso dei documenti elettronici accessibili, a fare da puntatore è il nostro pilastro basilare: gli stili di paragrafo. Dovrebbe essere chiaro che se gli stili non sono presenti, il software cercherà di interpretare a modo suo il documento, "inventando" autonomamente una struttura ipotetica.
Nel caso di documenti lineari, è possibile ottenere dei buoni risultati. Ma se il documento ha un layout appena complesso, e se il flusso del contenuto non è lineare ed esplicitamente collegato (per esempio, nei programmi di impaginazione incatenando correttamente i box), il lavoro del software che interpreta la struttura è affidato a degli algoritmi, la cui efficacia non è per nulla certa.
Sono gli stili di paragrafo a costituire gli elementi primari *sia* della struttura logica del documento *sia* della sua accessibilità tecnologica.
A livello di maggiore dettaglio, se una tabella è dotata di elementi header, come visto in precedenza la tecnologia assistiva potrà individuarli tramite `ROLE_SYSTEM_ROWHEADER`, e così via. È evidente che se questa struttura non è stata esplicitata, sarà molto difficile per la tecnologia assistiva interpretarla, e si avranno errori e malfunzionamenti fino ad arrivare in certi casi al blocco del computer.

Esercizio 1:

- scaricare e installare Accessibility Probe da http://www.eclipse.org/actf/downloads/tools/accprobe/index.php,
- scaricare NVDA da http://www.nvda-project.org/ (screen reader)

Aprire i due programmi e puntare un browser a una pagina conosciuta. Osservare in Accessibility Probe gli eventi provocati dall'interazione con il computer e ascoltare il rendering audio di NVDA.

Fare la stessa cosa con un documento PDF in Acrobat, e un documento in Word.

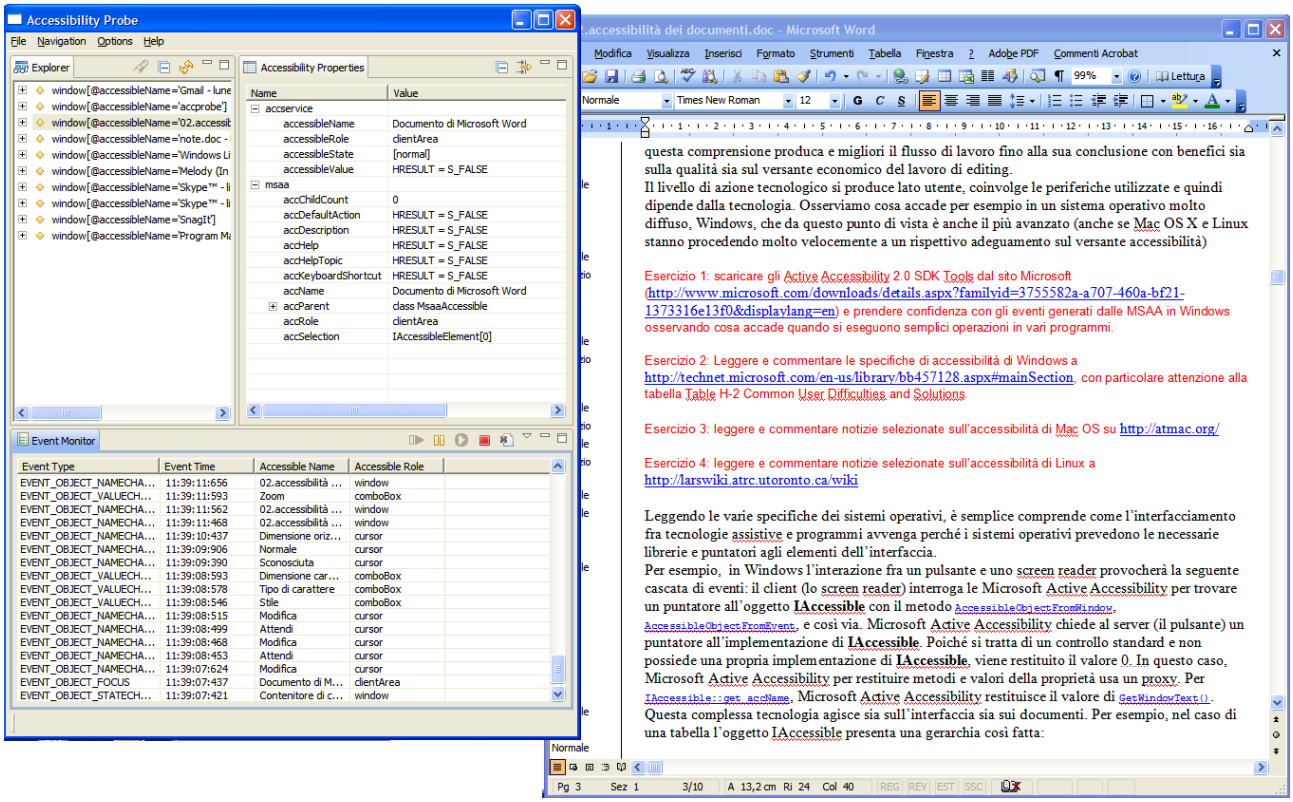

Figura 1. Accessibility Probe all'opera

In una pagina HTML, o un file PDF con tag, uno screen reader o un ingranditore di schermo riusciranno ad interpretare la struttura presente e a gestirla perché questa struttura è esplicita e ben delineata, poiché i formati HTML e PDF sono di libero dominio e tutti i programmatori possono conoscerne le specifiche.

Esercizio 2: installare CrossEyes (http://www.levitjames.com/crosseyes/CrossEyes.html), aprire un documento .doc ed osservare la sua struttura. Se questa non esistesse, applicare stili di paragrafo a blocchi di testo ed osservare i cambiamenti.

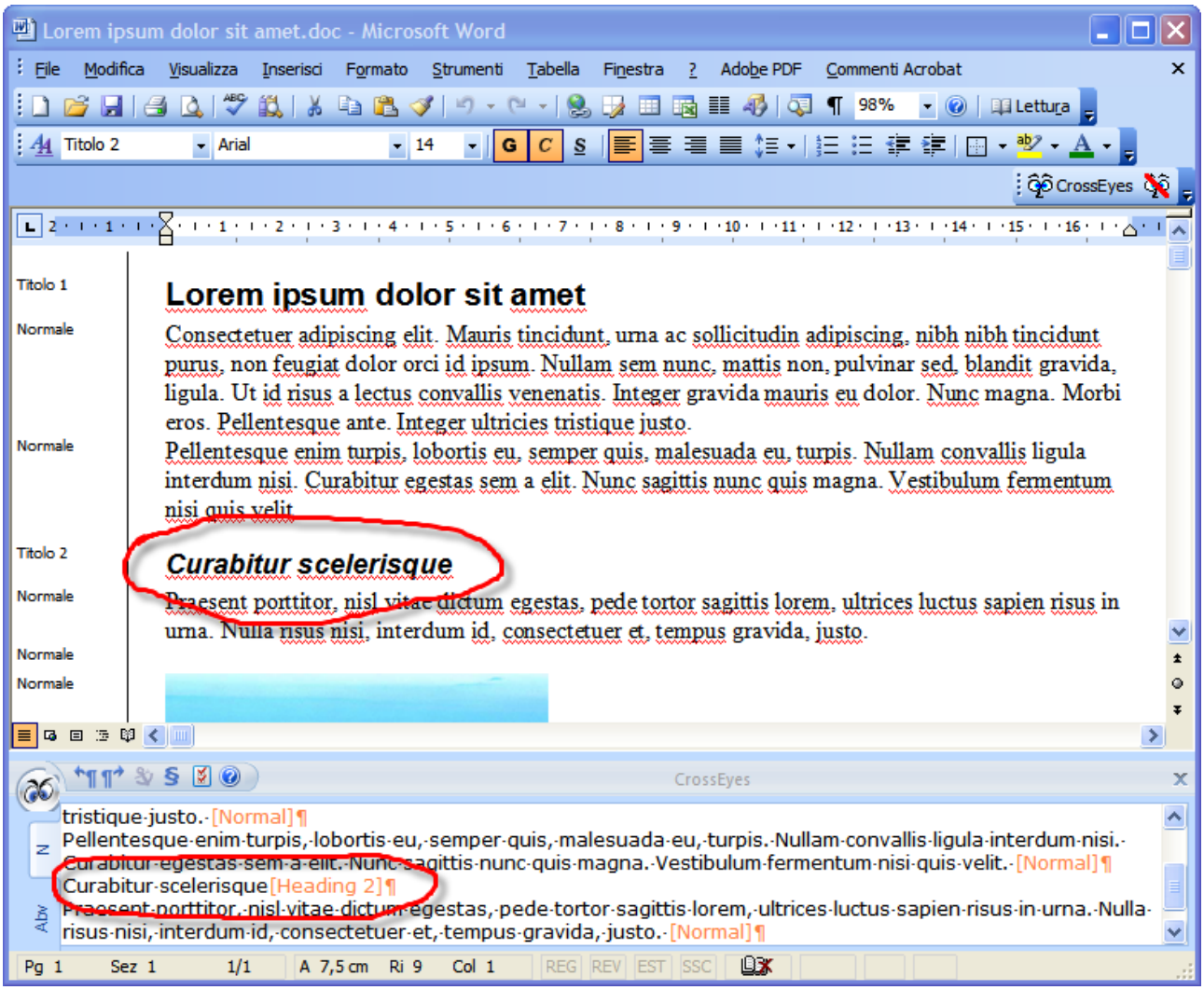

Figura 2. CrossEyes in azione: anche Word possiede un proprio markup

Esercizio 3: per comprendere quanto sia importante la struttura, aprire un documento doc, pdf e html ed usare i tasti scorciatoia di Jaws per navigarli.
Per esempio premere H per leggere i titoli (in doc e PDF attivare se necessario le Navigation Quick Keys), usare Ctrl e Alt per leggere le tabelle, Tab e Maiusc+Tab per scorrere i campi di un modulo. Per una lista completa dei tasti utilizzabili, premere Ins+H per accedere all'help di Jaws o Ins+F1 per un help contestuale.

Comprendere cosa si intende per struttura: come è fatto un documento per un computer

Ricreate *nella vostra mente* un foglio stampato, di qualsiasi tipo: un giornale, un libro, un depliant, la prima cosa che vi viene in mente ed osservatelo. Probabilmente il contenuto testuale non è chiaro, si tratta di un'immagine mentale, ma la forma invece lo è. È facile distinguere i titoli, il testo normale, le immagini, gli elementi evidenziati in riquadri colorati e la disposizione di questi elementi secondo uno schema preciso (per esempio, un titolo sarà più grande e verrà posto prima del testo a cui si riferisce). Ora, immaginate di fare la stessa cosa insieme ad altre persone. Difficilmente i contenuti potranno coincidere, ognuno avrà un'immagine mentale di questo documento diversa. Però, alcuni elementi saranno uguali per tutti: i titoli avranno un aspetto diverso dal testo normale, le figure saranno facilmente distinguibili dal resto, gli elenchi puntati e le tabelle ben delineate e distinguibili. Questo è grosso modo quello che accade a un computer: il computer

non è in grado di capire di cosa parla il testo, ma è invece molto efficace quando si tratta di interpretarne la struttura.

Esercizio 1: in Word, aprite un documento, verificate che sia attiva la vista Normale (Visualizza>Normale) e quindi selezionate Strumenti>Opzioni. Digitate 3cm nella casella di testo Larghezza area di stile della scheda Visualizza e fate clic sul pulsante OK. Nell'area destra del documento dovrebbe essere presente una nuova sezione in cui vengono mostrati gli stili applicati al testo. Verificare se gli stili applicati rispettano i significati semantici degli stili stessi (a un titolo principale, dovrà essere assegnato lo stile Titolo 1, al testo normale lo stile Normale, e così via).

Esercizio 2: aprire un documento in Firefox osservare i tag che delineano la struttura del documento e individuate le analogie.

Altre opzioni: il markup

Ora è necessario comprendere che un documento riprodotto su un computer possiede due livelli di struttura: una è quella logica, propria del documento e delineata dall'autore dello stesso. L'altra appartiene a quello che in XML viene chiamato l'abstract document. Nel caso di documenti in formato binario, come il doc, la struttura dell'abstract document viene rappresentata dagli stili di paragrafo, che fanno le funzioni di entrambe le strutture.
È una rappresentazione spuria, che andrà a sparire (le più diffuse suite di strumenti per l'ufficio hanno già adottato XML come formato di file di default, e tutti i più diffusi programmi di impaginazione sono in grado di gestire XML, chi più chi meno), ma con cui dobbiamo fare i conti.
Nell'immagine seguente viene mostrato un documento Word in cui sono visibili entrambe le strutture, sia quella logica sia quella del markup aggiunto al documento.

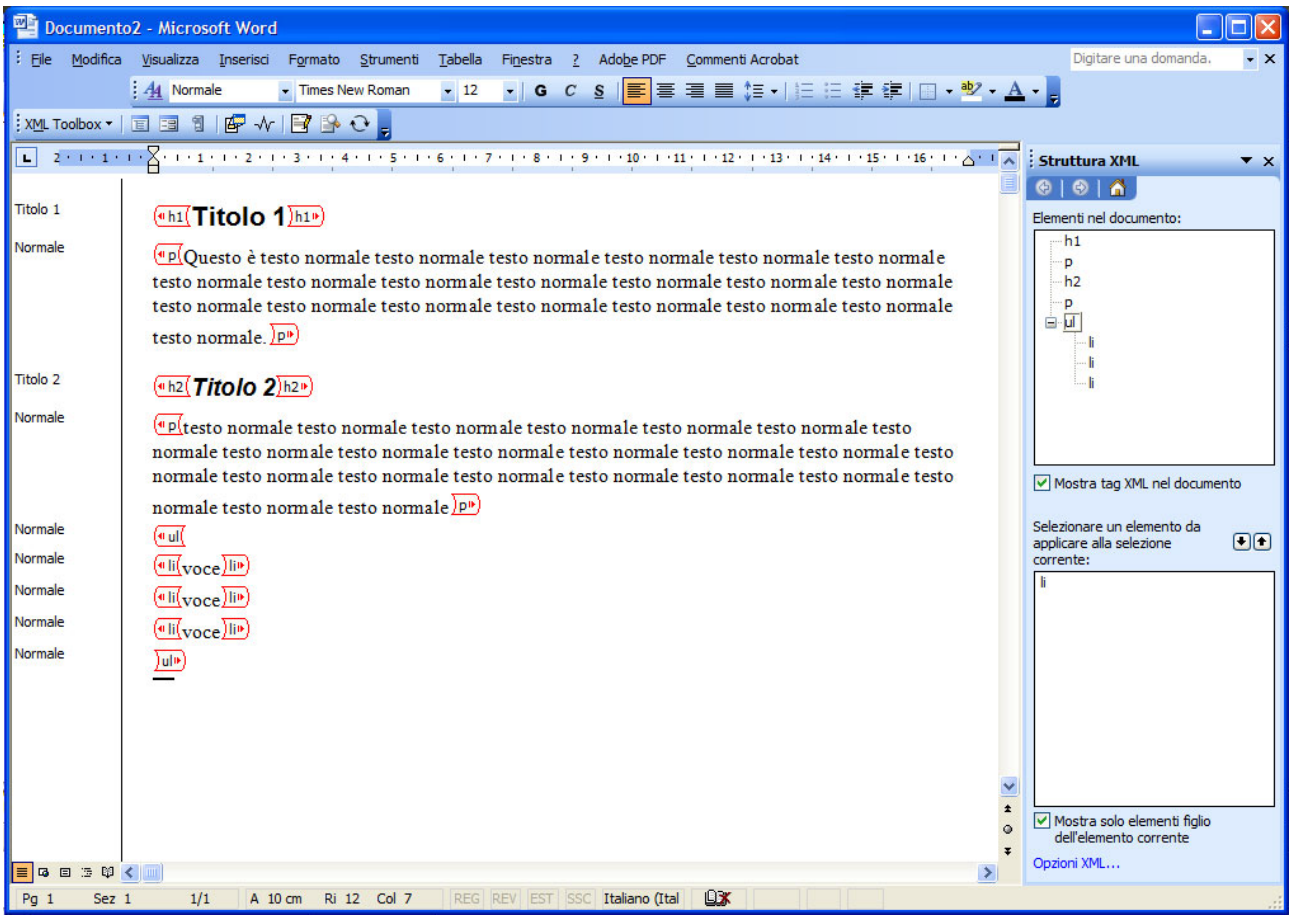

Figura 3. Nella barra degli stili sono visibili i nomi degli stili di paragrafo applicati al testo. Nel testo stesso sono visibili i marcatori che delineano la struttura a tag del documento. Le due strutture sono entrambe attive, contemporaneamente. In assenza dei tag, a fare da marcatori sono gli stili di paragrafo.

È importante comprendere questa dualità, soprattutto in vista di quando nella Quarta Parte impareremo a verificare e validare i documenti elettronici accessibili.
Si tratta di un passaggio complesso, poiché è naturale osservare il documento soltanto da un punto di vista logico, come ce lo mostra la mente attraverso gli occhi.
Un computer non ha la minima idea del significato logico delle parole “Titolo 1” o del tag <h1>. Però, è molto efficace nell’individuarle e, se un programma gli indica cosa farne, nell’elaborarle.
È soltanto per nostra comprensione che utilizziamo nomenclature dotate di semantica, in grado di descrivere il significato logico di quei marcatori. Dal punto di vista del computer, che elabora quelle informazioni, non hanno significato intrinseco, sono astratte e rappresentano soltanto dei puntatori.
I problemi di interoperabilità che abbiamo visto in precedenza (XPress non interpreta tutte le mie informazioni quando importo il file, Indesign mi ha perso i testi alternativi, e così via) dipendono dal fatto che i formati normalmente utilizzati (nello specifico, i formati doc o rtf) non sono trasparenti, sono formati proprietari di cui non si conoscono tutte le specifiche. Un documento Word potenzialmente potrebbe essere aperto e fruito al 100% soltanto utilizzando per la sua fruizione Word. Gli altri programmi devono basarsi su filtri di importazione, la cui efficacia dipende da tanti fattori.

Esercizio: aprite un documento .doc con il Blocco Note di Windows. Osservate come il documento sia illeggibile (il doc è un formato proprietario).

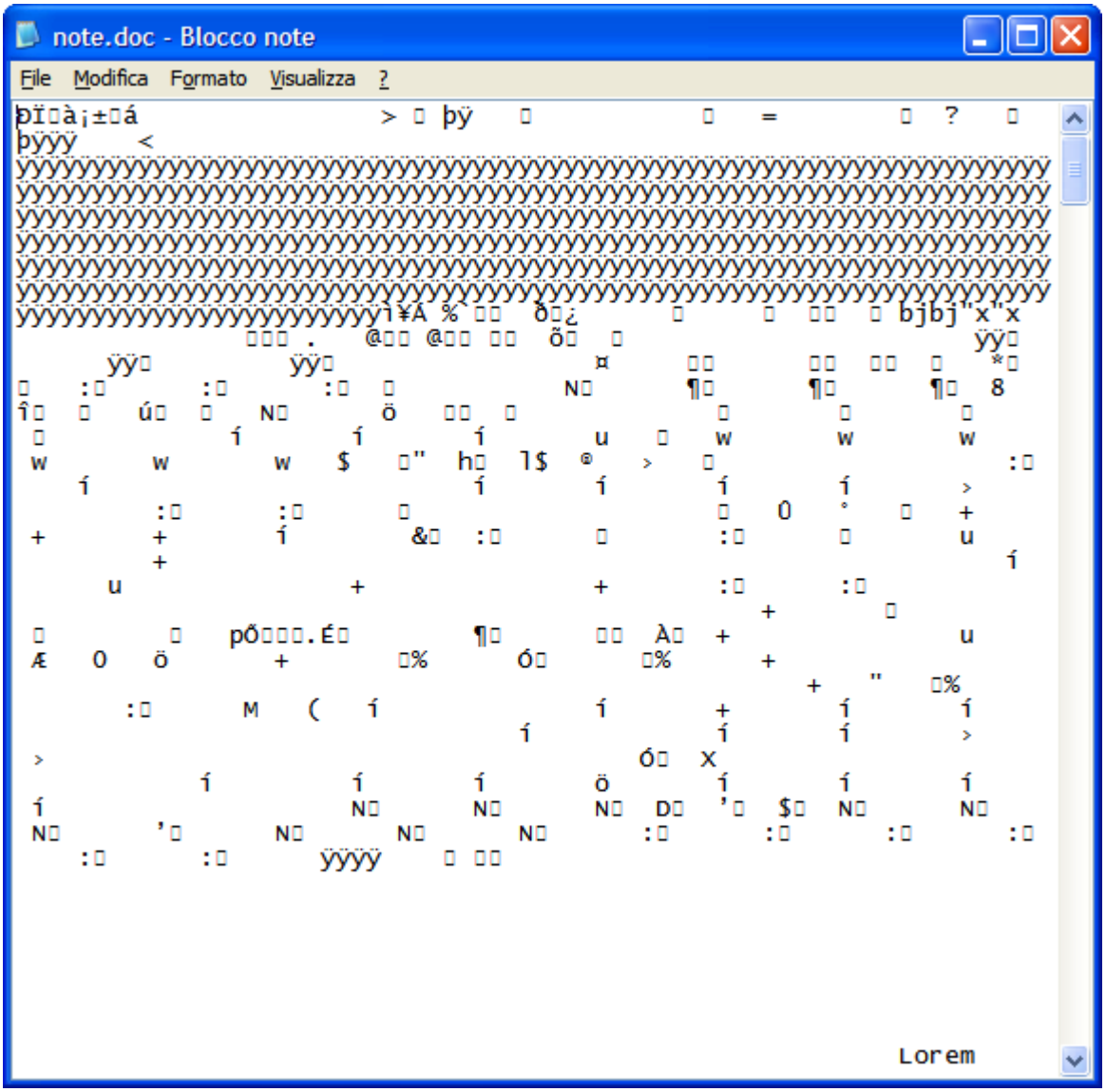

Figura 4. Un documento doc aperto in Blocco Note di Windows è illeggibile, poiché il formato binario del file può essere interpretato e riprodotto soltanto da Word o da un programma dotato di un filtro di importazione di questo formato.

Il markup: lezioni dal Web

Il Web è un ambiente variegato, consultabile con strumenti diversi e in condizioni decisamente diverse. Per esempio, un utente potrebbe usare un normale browser come Firefox o Internet Explorer, un altro un palmare, un altro ancora un browser testuale. Oppure, una persona non vedente potrebbe utilizzare uno screen reader per farsi leggere i contenuti della vostra pagina. Senza arrivare a questi estremi, una persona miope potrebbe aver bisogno di rendere più grandi i caratteri a schermo per leggere con facilità quanto esposto dall'autore, oppure desiderare di cambiarne i colori per ottenere una migliore consultazione. Se tutto questo viene fissato e bloccato inserendo le informazioni relative alla presentazione dei contenuti nella struttura stessa, tutto questo non accadrà. Non sarebbe male se il file potesse essere salvato in un formato universale, senza perdere però l'impaginazione faticosamente ottenuta e necessaria alla corretta interpretazione sensoriale, vero?. Inoltre, poter essere certi che il proprio lavoro potrà essere consultato da praticamente chiunque e ovunque, su un qualsiasi computer è una sensazione appagante e rassicurante. Esattamente quello che si ottiene con XHTML e CSS: contenuti del documento e relativa struttura in un file, impaginazione in un file diverso, strettamente (strict) separati e in formato solo testo. Chi non è interessato alla presentazione potrà comunque fruire dei contenuti, strutturati secondo le regole

previste dall'autore, mentre gli elementi di formattazione verranno applicati solamente se necessario o richiesto. Si chiama accesso universale, ed è una bella aspirazione.
La cosa buffa è che questa modalità operativa viene dal mondo della stampa, quando è stato inventato SGML.
A questo punto, disponendo del file XML prodotto il risultato potrebbe già essere visualizzato in qualsiasi browser, sia grafico sia testuale. Nel browser grafico il file avrà uno stile di presentazione come deciso dal foglio di stile di default che ogni browser utilizza se l'autore non ne predispone uno apposito (analogamente a quanto accade in Word). Per poter stabilire le proprie regole di presentazione si dovrà creare il foglio di stile CSS, ovvero un ulteriore file di solo testo contenente esclusivamente le informazioni da comunicare al browser su come presentare il documento: <h1> è di colore rosso, nel carattere Arial, di dimensioni 1.5 em - ovvero, grande una volta e mezza più delle dimensioni del testo normale . Se avete capito come funziona la struttura, sarà facile scrivere le regole relative a ciascun elemento riferendosi ad esso col nome del tag che lo definisce (per esempio, <h1> per i titoli di livello 1, <table> per una tabella, <a> per un collegamento). Per distinguere i contenuti, il computer (che ricordiamo, non comprende in alcun modo cosa avete scritto) si riferirà invece a quanto presente fra le parentesi angolari, < e >, disponendo i contenuti secondo le regole da voi preparate e comprendendo che quell'elemento si conclude dove rileverà il nome dello stesso tag preceduto da una barra, per esempio </h1>. L'insieme completo di tag di apertura, contenuto, tag di chiusura si chiama elemento. Piuttosto semplice, vero?
Inizialmente può essere difficile riuscire a distinguere quello che viene definito struttura (in XML, l'abstract document, ovvero quello che si visualizza facendo View>Page source in un browser) dalla presentazione. La traduzione letterale presentazione in realtà è un po' approssimativa, poiché nella terminologia del W3C con presentation si intende il rendering del documento ottenuto con il relativo CSS, o i relativi CSS se sono stati predisposti diversi fogli di stile per le periferiche che possono eseguire il rendering del documento (tipicamente il monitor, ma anche stampanti, computer palmeri, audio). Un problema è costituito anche dalla nostra abitudine nell'osservare i documenti già renderizzati, per esempio un documento Word ci mostrerà a video, anche senza nostro intervento, determinati aspetti grafici associati a particolari elementi (un esempio evidente, i titoli formattati con lo stile titolo). Sembrerebbe quasi che la "presentazione" sia inevitabilmente collegata al documento, ma non è così nemmeno per Word. È solo nascosta (vedi esercizio precedente con CrossEyes).

Esercizio 1: convertire un documento Word a HTML
Scaricare la DTD XHTML 1.0 Strict da http://www.w3.org/TR/XHTML1/XHTML1.zip
Scaricare qad_doc2xml da a http://www.frank-it-beratung.de/doc2xml/english.html

1. Creare un documento con semplice struttura in Word, salvarlo. Aprire qad_doc2xml, caricare la DTD XHTML 1.0 Strict (Get Taglist from DTD), caricare il documento doc (Source) e mappare gli stili di Word ai tag XHTML.
2. Indicate il nome del file di destinazione (Target XML) e fate clic sul pulsante Convert.
3. Osservare il risultato della conversione in un browser
4. Creare un file PDF da Word, aprire Acrobat e osservare la struttura del PDF nel pannello Tag.
5. Identificare le analogie strutturali.

Questa trasparenza e interoperabilità si ottiene soltanto standardizzando i processi e utilizzando formati di distribuzione che prevedano l'accessibilità dei file, come PDF, XHTML o ODF.

Tipologie e formati

Non esiste un vero e proprio formato standard da utilizzare per la creazione di documenti elettronici accessibili, ed è per questo che sono state esposte delle caratteristiche slegate da formati precisi.
Per i testi, il formato doc viene molto utilizzato soltanto perché è il formato nativo del word processor più diffuso, Word. Però, altrettanto utilizzato è il formato PDF di Acrobat, così come in

alcuni casi viene utilizzato rtf, che però presenta alcune controindicazioni. e forse in futuro ODF. In realtà il formato più adatto potrebbe essere proprio quest'ultimo, creato proprio per essere un formato standard. Il problema è che a oggi non esistono editor in grado di gestirlo, né in ODF 1.0 né in ODF 1.1, la versione da utilizzare in quanto accessibile. Di conseguenza, la nostra attenzione per ora sarà sui formati doc e pdf.
Un ulteriore formato utilizzato in passato è il txt. Questo formato è costituito da solo testo, di conseguenza non contiene alcuna informazione sulla struttura del testo stesso, né ulteriori specifiche di accessibilità. Proprio per la sua natura è il formato che può essere letto con maggiore facilità, ma che contiene il minor numero di informazioni accessibili.

Formati standard

Per esempio, le nuove versioni di MS Office utilizzano come formato standard `Office OpenXML` (OOXML), Open Office il proprio odf (`Open Document Format`).
Questi formati hanno alcune caratteristiche importanti in comune:

- sono standard ISO
- supportano l'accessibilità

Esistono enti, come `OASIS` (Organization for the Advancement of Structured Information Standards), che si occupano della creazione di `DTD` (document type definition) o `XSD` (xml schema definition) standardizzati, in cui il lavoro di analisi logica è già stato eseguito e testato. Esempi classici sono `Docbook`, `DITA` (Darwin Information Typing Architecture), e `migliaia di altre DTD` usate in ogni ambito.

Perché uno standard?

Uno **standard**, in ambito informatico, ma anche nelle telecomunicazioni, nei sistemi di misura, nell'ingegneria in genere, rappresenta *una base di riferimento, un paradigma codificato per la produzione di tecnologie fra loro compatibili e interoperabili*, che siano componenti hardware, software o infrastrutture di rete.
Diversi enti a livello internazionale come l'ISO (International Organization for Standardization) e l'IEEE (Institute of Electrical and Electronics Engineers) propongono, concordano e ratificano gli standard nei diversi ambiti. Prima di essere considerato tale dalla comunità internazionale, ed essere preso a buon diritto come modello di riferimento, uno standard passa attraverso una serie di fasi di analisi e accreditamento:

- L'analisi delle esigenze dell'utenza da parte delle università e dei settori che si occupano di ricerca e sviluppo per le varie aziende produttrici, dà luogo alla ricerca di soluzioni per i problemi e le necessità eventualmente riscontrate.

- Quando possibile, delle specifiche tecniche vengono emesse sottoforma di descrizioni documentate estremamente dettagliate.

- Il testing e l'utilizzo di tali specifiche da parte della comunità internazionale dei produttori e dei laboratori di ricerca evidenzia le soluzioni migliori. A questo gli enti internazionali possono cominciare a scegliere cosa scartare e cosa mantenere dei vari contributi, producendo l'insieme delle specifiche finali.

- Le specifiche finali vengono accreditate come standard internazionale da un ente scientifico. Il risultato è un documento che descrive il modello cui le ditte di settore dovranno attenersi, pena l'incompatibilità dei loro prodotti tecnologici.

Uno standard, per esempio, nel nostro caso garantisce che i file siano interoperabili, che i programmi li gestiscano tutti allo stesso modo (chiunque abbia provato a usare qualche programma di impaginazione conosce le incongruenze di parsing dei file Word – formato proprietario, basta provare ad aprire un file con Indesign, Xpress o Framemaker. Il risultato non è mai lo stesso).
Nel nostro caso l'interoperabilità è un requisito fondamentale, poiché come vedremo praticamente nella prossima sezione, e come abbiamo già visto in precedenza, le tecnologie assistive utilizzano come base fondamentale la *struttura del documento*, che deve essere conosciuta e riconoscibile.