Alma Mater Studiorum – Università di Bologna

FACOLTA DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Triennale in Informatica

Metadatazione di Documenti di un CMS : un' Esperienza

Tesi di laurea in Tecnologie Web

Relatore
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Davide Malfatti

I Sessione
ANNO ACCADEMICO 2009/2010

Indice

Capitolo 1

Introduzione

In un articolo pubblicato nel 2001 Cory Doctorow, giornalista specializzato nell' IT e famoso scrittore di libri di fantascienza, parla dell'importanza della creazione di meta-dati riguardanti le pagine web, e dei problemi connessi alla realizzazione di tali meta-informazioni [DOC].
Nella sua introduzione Doctorow evidenzia come la nascita e la diffusione di xml, il linguaggio markup più utilizzato al mondo, abbia notevolmente migliorato la qualità e quantità di dati e regole che è possibile aggiungere ad un determinato testo. Questo sono infatti i meta-dati: dati strutturati che ci parlano di altri dati non strutturati.
Verso la fine del suo articolo l'autore accenna ad un mondo utopico dove ogni risorsa online è referenziata dai suoi meta-dati. Un mondo dove la ricerca delle informazioni è immediata ed esaustiva. Questo mondo utopico deriva dal massiccio e costante utilizzo di meta-dati nella definizione, catalogazione e ricerca dei contenuti online. Tramite questi infatti, la ricerca di informazioni nel Web è può diventare molto più rapida e funzionale. Quasi dieci anni dopo la pubblicazione di quell' articolo, la creazione di meta-dati è ancora un argomento di forte interesse nella comunità scientifica.
Una valutazione pubblicata da Encraft nel maggio 2010, stima a 240.000.000 il numero di siti internet creati nella storia del WWW [LIV]. Questi siti rappresentano un enorme insieme di contenuti testuali e multimediali, tutti collegati tra loro e disponibili agli utenti. Da questo vastissimo insieme di contenuti e informazioni, nasce la necessità e la volontà che tali contenuti possano essere facilmente reperibili ad ogni utente utilizzatore del servizio. Una metodologia proposta per avvicinarsi alla risoluzione di questo problema è proprio rappresentata dalla creazione informazioni supplementari riguardanti il

testo: i meta-dati e, insieme ad essi, la realizzazione di una tecnologia in grado di leggere e sfruttare tali dati aggiuntivi. Questa tecnologia, deve essere in grado di permettere a qualsiasi tipo di utente creatore di contenuto del WWW, di creare ed editare i meta-dati relativi in maniera semplice, veloce ed intuitiva.
Il tema centrale dello scritto pubblicato da Doctorow riguarda quali effettivamente siano i sette grandi ostacoli relativi alla creazione dei meta-dati. Vedremo come ad alcuni di questi problemi siano stati parzialmente risolti dalle applicazioni Web sviluppatesi ad oggi.

- **People lie:** Gli utenti per referenziare i propri contenuti possono utilizzare meta-informazioni non precisamente corrispondenti alle informazioni contenute dal testo. Questo può essere in parte combattuto guidando l'autore di testo e meta-dati nella creazione delle meta-informazioni da definire, e permettendo di impostare per ogni categoria solo una ristretta opzione di meta-dati.
- **People are lazy:** Gli utenti che solo occasionalmente creano del contenuto online non comprendono l'importanza dei meta-dati e non sono propensi a “perdere del tempo” definendoli. Una miglioria in questo può sicuramente essere determinata da una interfaccia relativamente semplice veloce e guidata per la creazione di informazioni, che permetta scelte rapide tra una serie di valori proposti.
- **People are stupid:** Se questo non fosse un dato di fatto, o se questo problema fosse risolvibile, non ci sarebbero più guerre e problemi ben più seri della creazione di testo fornito di meta-informazioni e markup. Impossible to resolve.
- **There's more than one way to describe something: I**n ogni linguaggio e tecnica espressiva esistono più modalità per descrivere gli stessi concetti. Ci sono poi istanze e parole che hanno un significato in un contesto e uno differente in un altro. La classificazione delle parole chiave utilizzate nel testo ridurrebbe il rischio di tali ambiguità.

Oggigiorno nuove tecnologie possono essere d'aiuto nella risoluzione di alcune

di queste problematiche. Il crescente sviluppo di applicazioni appositamente disegnate per il Web ha notevolmente aumentato le potenzialità di creazione e pubblicazione di pagine ed articoli da parte di ogni tipo di utente fruitore del WWW.

La crescita ed espansione di software come Wiki e CMS avvenuta negli ultimi anni ha favorito la collaborazione tra utenti alla creazione di informazioni.
Attraverso queste applicazioni si ottiene la possibilità di mettersi in contatto con persone sconosciute e collaborare assieme ad esse, alla creazione di contenuti sempre più completi ed esaurienti.
I wiki, tra i quali l'esempio più noto è sicuramente Wikipedia, implementano pagine Web relative a temi specifici e permettono a tutti gli utenti registrati al servizio di contribuire alla creazione e crescita delle informazioni contenute[Wik10]. I wiki permettono inoltre attraverso uno specifico linguaggio di markup la determinazione di keywords, termini utili per referenziare il testo, e meta-dati che descrivano e cataloghino il contenuto [IMPV].
La stessa cosa può avvenire nei CMS. Questi sono applicazioni Web ancora più evolute dei wiki: si basano infatti su numerose librerie interne che permettono, anche ad utenti inesperti della programmazione Web, la realizzazione di contenuti e servizi di condivisione di informazioni, opinioni e interazione sofisticati[OCMS]. La prospettiva di un Web dove si veda crescere l' utilizzo di applicazioni come i CMS, può rappresentare la strada tecnologica da intraprendere per la crescita sia della meta-dazione dei documenti, sia per il diffondersi di un web semantico, intelligente e strutturato.

I CMS sono applicazioni dalle possibilità di espansione praticamente infinite, perché seguono una progettazione e risoluzione dei problemi modulare. Questo dà la possibilità di creare e aggiungere continuamente delle funzionalità ad un dato CMS, e quindi, ad un dato sito web sviluppato tramite queste applicazioni.
La modularità dello sviluppo può consentire un continuo innalzamento del livello di astrazione del software, migliorandone l'interazione tra persona e computer. Così se nei primi CMS la scrittura del contenuto doveva avvenire

tramite codice HTML, ora può essere effettuata tramite editor di testo avanzati che permettono una formattazione strutturata dell'output e l' auto-conversione dello stesso in codice HTML.
Tra i tanti ambiti di sviluppo portarti avanti da software come i CMS, è ovviamente presente anche il problema della meta-datazione dei contenuti Web: sono attualmente disponibili alcune implementazioni di CMS che supportano il salvataggio e la creazione di meta-dati riguardanti articoli e testi pubblicati online. Citiamo alcuni esempi che verranno ripresi nei capitoli a seguire.

Milos è un progetto per la condivisione di informazioni riguardanti i contenuti delle librerie digitali. Le librerie digitali sono portali che offrono diversi tipi di contenuto: su di esse possiamo trovare ogni genere di file multimediale e testuale. Il problema di software come le librerie digitali è che queste, per quanto permettano facilmente l'importazione ed esportazione dei loro contenuti, raramente sono fornite della funzionalità di scambiare tra loro informazioni relative ai contenuti. Milos offre quindi tramite la lettura, traduzione e salvataggio di queste informazioni in un formato proprio, la condivisione, importazione ed esportazione di tali meta-informazioni. Milos basa il suo funzionamento sul linguaggio di markup XML.
Il markup, relativamente ad un testo, è una tecnica che permette di strutturare il testo e differenziare le varie parti che lo formano a seconda del loro significato e della funzione che svolgono.
XML è il linguaggio di markup più utilizzato dalle tecnologie Web. Questo non solo per la facilità della sua sintassi, ma anche perchè attraverso XML sono poi facilmente creabili altri linguaggi di markup utilizzabili per diversi ulteriori sviluppi.

OLA e AMN sono applicazioni per la creazione automatica di meta-dati. Se in precedenza abbiamo esaminato il tema dello scambio di informazioni tra diversi software, ora analizziamo il caso dove l'incarico di definire totalmente le meta-informazioni viene lasciato ai programmi. I due applicativi svolgono tale servizio con modalità differenti.

AMN è un applicazione composta da due moduli differenti che cooperano nella ricerca di meta-dati analizzando l'oggetto della ricerca sotto diversi aspetti: il primo si cura di conoscere e creare informazioni analizzando l'oggetto stesso; il secondo invece analizzando il suo contesto.
OLA invece permette la creazione di meta-informazioni a partire da documenti testuali di diversa formattazione. I documenti per ora supportati sono HTML, PDF, PPT e Word. Per ognuno di questi formati esiste un modulo che si preoccupa della creazione di informazioni. Tali informazioni sono poi passate ad un generatore di meta-dati.

Esistono poi diversi CMS che durante la creazione di contenuto online permettono la creazione e il salvataggio di informazioni ad esso relative. Queste informazioni vengono richieste all'utente in fase di creazione tramite un apposito form di inserimento. Ogni CMS organizza, salva,visualizza ed utilizza tali informazioni liberamente. L'amministratore di ogni sito implementato tramite CMS, cioè il responsabile del contenuto e delle sue applicazioni, può liberamente decidere come gestire la creazione di meta-dati relativi ai contenuti del suo sito. Il mio lavoro di tesi ha riguardato proprio questo ambito: scopo dell'elaborato è quello di fornire automaticamente all'interno di un CMS, Drupal, una interfaccia “user friendly” per la creazione di meta-dati relativi ai contenuti pubblicati sul sito. Tale funzionalità svincola l'amministratore del sito dalla responsabilità di dovere realizzare, all'interno del portale, i presupposti tecnici che permettano ad ogni utente creatore di articoli di definire meta-informazioni relative. Il sistema, chiamato Buck, si preoccupa di creare un form per la definizione di meta-informazioni durante la creazione degli articoli. Sono inoltre state adottate diverse strategie per suggerire all'utente i valori plausibili assegnabili a tali informazioni.
La valutazione del sistema prodotto, presente nel quarto capitolo della dissertazione, si basa sul raggiungimento da parte del sistema Buck di specifici criteri. Il sistema dovrà essere in grado di aiutare sensibilmente l'amministratore del sistema nella creazione di scheletri “user-friendly” per la introduzione di meta-dati riguardanti gli articoli. Le migliorie offerte non solo dovranno

riguardare gli aspetti implementativi, ma anche decisionali, nella realizzazione dei meta-dati. Il sistema dovrà infatti stabilire quali utenti fruitori dei servizi offerti, potranno controllarne le funzionalità e scegliere quindi quali siano gli utenti autorizzati alla creazione o definizione di meta-informazioni assegnabili a determinati contenuti.
In altro ambito, il sistema avrà anche la responsabilità di elaborare una strategia competitiva per favorire l'immissione dei meta-dati da parte dell'utente creatore del contenuto. Si valuterà quindi se la metodologia proposta da Buck per la creazione di meta-informazioni sia uno strumento adatto ad un utilizzo su larga scala.

Le possibili evoluzioni delle potenzialità del sistema sono sostanzialmente due. La prima riguarda la prospettiva di estendere le meta-informazioni create da parte degli utenti tramite lo sviluppo di appositi moduli automatizzati intenti alla creazione di ulteriori meta-dati. Questa funzionalità da un lato ridurrebbe il lavoro necessario per la creazione di articoli e informazioni necessario ad ogni utente, e dall'altro farebbe diminuire il rischio di immissione di meta-informazioni scorrette.
La seconda evoluzione del sistema Buck potrebbe avere come argomento il Semantic Web. Il Semantic Web è un progetto W3C per la creazione di contenuti Web dotati di un significato ontologico interpretabile dalle macchine. All'interno di questo vasto processo per la creazione di contenuti dal significato ontologico interpretabile dalle macchine, si passa anche attraverso la necessità di definire statement RDF sugli stessi contenuti. Uno statement è una tripla definita dalla relazione di tre concetti: soggetto – verbo – complemento oggetto.

Sono i meta-dati allora le possibili basi su cui si possono basare le conoscenze delle macchine. Infatti, se adeguatamente catalogati ed utilizzati, potrebbero essere essi stessi i valori degli statement RDF per la creazione delle triple “soggetto – verbo – complemento oggetto”.

Capitolo 2

Creazione di meta-dati su Content Management System

2.1 L'importanza di creare dei meta-dati

La meta-datazione dei contenuti online è una problema che oggigiorno gran parte degli enti che pubblicano dei contenuti sul WWW, dalle università ai venditori online, deve porsi. La mancanza di meta-informazioni riguardanti un articolo pubblicato può infatti precludere la presa visione dello stesso dagli utilizzatori.

La soluzione al problema può portare a ricerche più esaustive e quindi ad aumentare le possibilità che l'articolo creato venga referenziato dai motori di ricerca ed apprezzato dall'utente consultatore; la presenza e la disponibilità di meta-informazioni riguardanti gli articoli facilita anche la catalogazione degli stessi quale che sia l'ambito di interesse, dall' accademico al commerciale.

Ovviamente non tutti gli utenti creatori di contenuti Web sono in grado di organizzare l'implementazione di tali meta-informazioni e keywords. La soluzione proposta in questa tesi è quindi quella di utilizzare un software CMS che guidi l'utente nella creazione di tali informazioni necessarie alla catalogazione del suo articolo.

Un CMS, acronimo di Content Managment System, è un programma per la gestione di contenuto online sofisticato che permetta un allargamento dei servizi che possono offrire i siti internet statici. Si tratta di software ad alto livello scritto per facilitare la creazione di contenuto per tutti gli utenti. In particolare però sono gli utenti inesperti della programmazione web a trarne i maggiori vantaggi: tramite CMS è possibile realizzare pagine Web in pochi passi senza necessità di conoscere alcun linguaggio di programmazione. In un CMS è

possibile mettere a disposizione dell'utente una serie molto vasta di funzioni. Tra queste, la definizione di meta-dati riguardanti il contenuto.
Ci apprestiamo ora a vedere qualche caso di studio del problema: analizzeremo esempi di creazione e successivo di meta-dati creati tramite Content Management System.

2.2 Milos : A Multimedia Content Management System for Digital Library Applications

Milos è un componente software sviluppato per la progettazione, catalogazione e implementazione di funzioni di libreria digitale. Una libreria digitale è un portale dove sia possibile l'accesso, a pagamento o meno, a vario materiale testuale o multimediale. Milos supporta la memorizzazione, il recupero e la descrizione dei contenuti,quali essi siano salvati in tali librerie digitali.
La descrizione di tali dati vien definita da un modello arbitrario definito tramite uno schema xml: questo garantisce al sistema una gestione flessibile dei vari tipi di meta-dato e descrizioni che possono essere salvati dalle diverse applicazioni con le quali Milos di interfaccia.
Le librerie digitali, che chiameremo d'ora in avanti DL, sono solitamente limitate ed in grado di contenere solo un certo tipo di materiale multimediale e i suoi relativi meta-dati. Sebbene il contenuto effettivo di tali DL, materiale multimediale o testuale, sia ben importabile ed esportabile tra un software DL ed un altro, lo stesso non si può dire per le informazioni relative al contenuto stesso. Questo problema nasce dalla generale assenza di standard strutturati per la classificazione dei files ed il salvataggio dei meta-dati relativi. L'obbiettivo di Milos è allora quello di realizzare una piattaforma per la condivisione di tutte le meta-informazioni sui contenuti presenti nelle varie DL.
La soluzione proposta del problema, nasce dall'analisi del funzionamento dei database. I database fungono spesso da componenti base per diverse applicazioni. La maggioranza dei CMS, ad esempio, fa uso di database per l'archiviazione delle informazioni relative al sistema e ai dati su di esso contenuti. I database forniscono quindi la stessa metodologia per il salvataggio

dei dati a tutte le loro applicazioni, utilizzando all'interno uguali modalità di archiviazione : le proprie tabelle e relazioni. Questo è sempre vero quale che sia l'applicazione che li utilizza.

L'intento di Milos è quello di riuscire ad offrire gli stessi servizi dei database:il software è stato disegnato per potere funzionare come Multimedia Content Management System(MCMS) sottostante a diverse applicazioni DL, ed offrire ad esse lo stesso servizio: il salvataggio, l'archiviazione e gestione di tutte le meta-informazioni create sui contenuti, quale sia la metodologia e tecnologia utilizzata per il salvataggio nelle singole applicazioni.

Il sistema dovrebbe essere in grado di manipolare ed estrapolare informazioni da dati formattati, sezioni di testo, presentazioni, dati xml e files multimediali.

2.2.1 Architettura di Milos

MILOS è composto da tre componenti principali: Metadata Storage and Retrieval (MSR), Multi Media Server (MMS) e Repository Metadata Integrator (RMI).

Tutti questi moduli sono implementati come servizi accessibili via Web e la loro interazione è realizzata tramite SOAP, un semplice protocollo che realizza lo scambio di informazione tra componenti software differenti[GSR].

Il MSR gestisce i meta-dati delle DL. E' implementato tramite database realizzato tramite sintassi XML, linguaggio universale per la realizzazione di markup sul testo. Gli obbiettivi di questo modulo sono quelli di supportare ed abilitare la descrizione di contenuti multimediali e dei loro meta-dati, qualsiasi sia il formato in cui vengono proposti. Le varie applicazioni DL infatti, in base alle proprie necessità, effettuano in maniera differente il salvataggio delle meta-informazioni riguardanti i loro contenuti.

MMS gestisce i documenti multimediali utilizzati dalle applicazioni DLs: questo deve avvenire ugualmente per tutti i contenuti qualsiasi sia il supporto fisico su cui sono memorizzati e qualsiasi sia la strategia di salvataggio degli stessi. Il modulo deve essere in grado di confrontarsi con il sistema di archiviazione di ogni specifica applicazione e generare il proprio schema relativo ai meta-dati.

Questo autorizza le applicazioni a non preoccuparsi dei dettagli tecnici riguardanti le tecniche di salvataggio ed encoding dei files.

Il RMI fornisce l'interfaccia di accesso alle applicazioni DL dei servizi precedenti. Inoltre, esso implementa la mappatura di meta-dati a seconda dei diversi schemi in cui essi sono salvati. Ad ogni applicazione viene fornito l'accesso, oltre a tutti i meta-dati salvati utilizzando la sua tecnologia, anche a meta-dati altrimenti inaccessibili all'applicazione stessa perchè con essa incompatibili.

Gli autori di Milos dopo avere analizzato le metodologie di salvataggio di meta-dati utilizzate dalle varie applicazioni DLs, hanno deciso di sperimentare una propria alternativa per la soluzione di questo problema. Le risoluzioni adottate dalle varie Dls sono state ritenute inadatte per problemi di inefficienza nella ricerca, o per il costo troppo alto richiesto dalla conversione degli schemi di salvataggio dei files. Ne sarebbe nato un prodotto incompleto che mediasse troppo nel suo funzionamento tra delle buone performance e delle discrete funzionalità.

La soluzione alternativa, non adottata dalle Dls, nasce dall'utilizzo di Xml per la creazione di schemi atti alla meta-datazione dei dati. L'utilizzo di questo linguaggio permette la realizzazione di schemi arbitrariamente complessi e la facile importazione ed esportazione degli stessi. E' stato quindi realizzato una database XML con speciali features per le applicazioni DL, in grado di memorizzare qualsiasi documento Xml ben formato. Una volta che il documento sia stato memorizzato nel database, possono immediatamente essere eseguite delle interrogazioni su di esso per l'accesso ai suoi dati. Un requisito che gli sviluppatori di Milos hanno voluto implementare è, infatti, la performance del sistema durante le procedure di inserimento e ricerca dati.

L'idea chiave del progetto è che qualsiasi sia il tipo di dato che le Dls trattino, queste possano salvare le informazioni relative utilizzando uno scheletro

uniforme per l'archiviazione di tali dati. Così la MMS sarà in grado di mantenere una mappa di tutti i documenti di meta-dati esistenti e li referenzierà tramite URN. Questi URN saranno poi utilizzati dalla varie applicazioni per prelevare o memorizzare tali documenti contenenti meta-dati dalla MMS, che ricopre quindi la funzione di gateway tra risorse e applicazioni.

2.3 La creazione automatica di meta-dati

E' opinione comune di alcuni ricercatori e scienziati che, a causa di varie e talvolta ragionevoli motivazioni, la creazione di meta-dati non possa essere lasciata unicamente alle capacità umane. La soluzione che questi vorrebbero adottare sembra dirigersi verso una soluzione software che sia in grado di implementare automaticamente la definizione di meta-dati.

La prima ragione che potrebbe spingere verso questa soluzione è la seguente: i creatori di meta-informazioni riguardanti articoli si rivelano spesso troppo onerosi da ingaggiare, soprattutto quando l'ambito di interesse è accademico ed educativo.

Una seconda ragione, ben diversa dalla prima, è causata dalla difficoltà per gli utenti di creare queste meta-informazioni. Per poter realizzare meta-dati riguardanti un testo, gli utenti devono infatti seguire sintassi standard per la definizione di tali dati. Quando l'utente è invece fortunato, dovrà compilare uno o più form, contenenti tanti campi da completare almeno quante sono le meta-informazioni da creare.

La soluzione a queste due problematiche allora, potrebbe essere rappresentata dall'automatizzazione della determinazione di tali meta-dati: si tratterebbe quindi di progettare un sistema che, a partire da un documento testuale e dal contesto in cui esso si trova, sia in grado di trovare e memorizzare le informazioni rilevanti.

La generazione automatica di meta-dati si suddivide in quattro parti differenti appartenenti al processo: analisi del contenuto, analisi del contesto, analisi dell'uso e analisi della struttura. Mentre nell'analisi del contenuto l'informazione viene estrapolata dalla conoscenza dell'oggetto stesso (che può avvenire tramite keyword e lingua), nell'analisi del contesto viene coinvolta una indagine

sull'ambiente contestuale all'oggetto di interesse. Informazioni supplementari sul contesto dell'oggetto possono infatti contribuire alla meta-datazione dell'oggetto stesso. Anche le finalità e modalità d'uso di un oggetto possono infine contribuire alla creazione di nuove meta-informazioni.

2.3.1 Due esempi tramite framework

Ci apprestiamo ora a descrivere come tramite framework sia possibile implementare dei generatori automatici di meta-informazioni.
Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework troviamo una serie di librerie e di codice utilizzabili con uno o più linguaggi di programmazione. Notiamo che la definizione di framework è molto simile a quella che abbiamo precedentemente dato di CMS. I due concetti sono similari e in alcuni casi queste due tipologie di applicazione svolgono la stessa funzione: fornire librerie, metodi e ambienti per gli sviluppi di nuovi servizi software.

2.3.2 Primo esempio: AMG, Automatic Metadata Generator

AMG è un servizio internet basato su framework settato per la generazione automatica di meta-dati. Il suo funzionamento si basa su due classi di moduli, denominati Object-based indexers e Context-based indexers . Object-based indexers genera meta-dati basati solamente sull'apprendimento derivante dall' oggetto stesso, decontestualizzato da qualsiasi altro dato o informazione.
Context-based indexers utilizza invece informazioni contestuali per la realizzazione.
Il framework è inoltre provvisto di plugin per l'estrazione di meta-dati dalle presentazioni eseguite in Power-Point chiamato Extractor, e di un altro modulo, denominato MetadataMerger, funzionale all'unione dei vari meta-dati creati dai vari moduli. Il modulo in questione ha il compito di evitare che si vengano a creare conflitti tra le informazioni create.

2.3.3 Secondo esempio: OLA, Extractor e Generator

A differenza dell'applicazione AMG, che si preoccupa di creare diverse metodologie per l'inserimento di meta-dati, l'obbiettivo primario del programma OLA è quello di estendere le funzionalità del framework in maniera flessibile ed estendibile, così da facilitarne l'inserimento all'interno del sistema.

Il sistema OLA implementa diversi moduli atti all'importazione di meta-informazioni carpite dai formati di diffusione maggiormente utilizzati per la diffusione di articoli(e.g. HTML, PDF, PPT, Word) . Il risultato di tali esportazioni di meta-dati può essere utilizzato tramite un servizio web che il sistema produce. E' difatti possibile, grazie alla flessibilità del software realizzato, collegarsi ad esso tramite altre applicazioni ed ottenere meta-dati relativi ai documenti desiderati.

OLA basa quindi il suo funzionamento su due classi di componenti principali: Extractor e Generator. L'extractor è responsabile dell'estrazione di informazioni testuali su un oggetto. All'interno del sistema può esistere un solo extractor per ogni tipologia di oggetto che possa essere analizzata: uno quindi per ogni diversa estensione di file o documento che possa essere preso in analisi.

Il generator invece utilizza l'output dell'extractor per produrre i meta-dati. Possono contemporaneamente esistere diverse versioni del modulo generator, perché ognuna di esse potrà focalizzare la propria attenzione, e quindi creazione, su informazioni diverse. L'utente utilizzatore del servizio è quindi chiamato alla scelta di quale, tra tutti quelli presenti, modulo generator utilizzare. Inoltre, la già citata flessibilità del software, permette a chiunque di creare plugin per estendere le funzionalità di generators e extractors.

2.4 Un CMS per la creazione di meta-dati: WordPress

Uno tra i Content Management System più diffusi del momento è sicuramente WordPress[WordP]. Il software merita una citazione nel contenuto di questa tesi per la grande rilevanza che i suoi sviluppatori hanno dato al problema di realizzare meta-datazione dei contenuti.

Ogni articolo creato con questa applicazione può essere classificato in base ad

una categoria di appartenenza (i.e. articolo sportivo). A sua volta ogni categoria può essere messa in relazione con le altre, e diventa quindi possibile creare una gerarchia di tutte le categorie che realizzate. Di conseguenza, ogni articolo associato ad una determinata categoria viene automaticamente messo in relazione con tutti gli altri articoli a loro volta associati ad una qualche categoria. Potere creare una gerarchia tra le categorie e quindi tra gli articoli appartenenti, permette un' ordinata suddivisione dei contenuti presenti nel sito, dando un significato non solo logico, ma anche ontologico alla suddivisione.
Inoltre Wordpress permette agli utenti creatori degli articoli di realizzare specifici meta-dati riguardanti il contenuto stesso: la definizione di quali siano questi meta-dati è totalmente a discrezione dell'autore dell'articolo. Questo lascia estrema libertà e responsabilità all'utente creatore del contenuto. Non sono previsti infatti form o template di inserimento standard che guidino la creazione di codeste meta-informazioni. Il creatore può liberamente scegliere quali debbano essere le informazioni importanti e imporre vincoli di tipo su quali debbano e possano essere i valori assegnabili ad ogni meta-dato.
Questa funzionalità rende Wordpress molto appetibile nel contesto di un progetto che si occupa del salvataggio di meta-informazioni su testo.

2.5 BT IPTV

BT è il più grande operatore telefonico del Regno Unito. Tra i diversi settori operativi di questa azienda troviamo anche la trasmissione di programmi televisivi via rete(IPTV). La grande quantità di materiale multimediale da controllare e distribuire ha portato l'azienda ad adottare un sistema di Content Management System per la gestione dei contenuti offerti. L'innovazione tecnologica necessaria all'adempimento dei servizi offerti agli utenti consumatori, ha ovviamente portato l'azienda a dovere migliorare la suddivisione ed organizzazione del materiale a disposizione. Tale evoluzione delle informazioni riguardanti materiale multimediale è stata realizzata, in gran parte, grazie alla adozione di un complesso sistema per la creazione di meta-dati, divenuti indispensabili al funzionamento del sistema [CFS].
L'azienda, durante lo sviluppo del software per la gestione dei contenuti, ha

realizzato come, fornire all'utente una facile interfaccia di meta-datazione degli articoli, fosse un requisito fondamentale del nuovo applicativo.
E' stato quindi sviluppato un tool chiamato MEX2 (Metadata EXtraction Version 2.0) che provvede alla creazione di meta-informazioni. In una prima parte del suo funzionamento il plugin stabilisce caratteristiche tecniche relative all' oggetto di interesse. Una volta stabilite queste informazioni e presentatele all'utente, si lascia a quest'ultimo la possibilità di inserire ulteriori informazioni di impossibile determinazione per il programma.
Se inizialmente l'utilizzo del tool MEX2 era quello di creare meta-informazioni interne al sistema, successivamente i suoi utilizzi sono cambiati. L'azienda ha infatti stabilito come un primo salvataggio di meta-dati non dovesse essere definitivo ma piuttosto, essere integrato poco a poco grazie alla interazione degli utenti.
L'architettura modulare del tool permette inoltre facili estensioni del programma stesso: nuovi formati di trasmissione, fonti di provenienza del materiale e algoritmi di interpretazione possono facilmente essere aggiunti al programma. MEX2 permette agli utenti il salvataggio di meta-dati attraverso vari formati, inclusi CMSL e XML.
BT nella distribuzione dei suoi contenuti prevede anche un sistema per la validazione del contenuto multimediale da pubblicare: questa validazione si basa sull'analisi di alcune proprietà del prodotto. Se il risultato di questa analisi è positivo, allora il materiale viene autorizzato alla pubblicazione. I meta-dati vengono utilizzati anche in questo frangente per la descrizione del contenuto e del sistema di valutazione dello stesso.

Capitolo 3

Il sistema Buck

Buck, acronimo di “Basic user creation kit” e allo stesso tempo di “Basic users cooperation kit”, è un sistema per la meta-datazione di contenuti su CMS.
Il sistema Buck permette l'importazione automatica, all'interno del particolare CMS Drupal, di template atti a diversificare e classificare tutti gli articoli che verranno creati su tale CMS. Ogni template rappresenterà un modello di articolo che sarà possibile creare sul portale di interesse.
Per presentare come funzioni il sistema Buck è necessario capire il funzionamento del CMS Drupal, definendo alcuni utenti fittizi che verranno utilizzati durante la spiegazione.

3.1 Definizione degli utenti

admin: E' l'utente amministratore del sistema. Colui che ha permessi illimitati e irrevocabili e stabilisce quali permessi concedere a tutti gli altri utenti. Per ogni sito, l'amministratore è l'utente che installa l'applicazione. Lo chiameremo Alberto.

Developer user: utente che utilizza o meno il servizio offerto dal CMS. Conosce il sistema e ne sviluppa eventuali modifiche ordinate dall'amministratore ed eventualmente suggerite da authenticated user. Questo utente ha la possibilità di cambiare anche il funzionamento di alcune parti del sistema ed è quindi provvisto di un accesso a dove il sistema fisicamente risiede. Quali parti del sistema possano cambiare viene deciso dall'amministratore. Lo chiameremo Barbara.

Authenticated user: utente registrato al sistema e loggato in un determinato

istante. Può condividere dati personali, creare, visionare ed editare del contenuto, cambiare l'interfaccia grafica personale del sito (quindi cambiare come il sito si presenta a lui stesso, ma non agli altri utenti). Lo chiameremo Carlo.

Anonymous user: Utente che semplicemente consulta il sito per ottenere informazioni. Non ha alcun interesse nel correggere o modificare il contenuto e le funzionalià del CMS. Lo identificheremo come Daniela.

3.2 Drupal

Drupal è un CMS modulare distribuito sotto licenza GNU GPL. Diventato un progetto libero dal 2001, negli ultimi anni l'evoluzione e la distribuzione di Drupal sono state fortemente determinate dalla crescita delle community nate attorno al CMS. In tali community si svolge un duplice processo: il più importante è lo sviluppo del software che avviene grazie ad utenti appassionati ed interessati, il secondo è la diffusione del software ad utenti "alle prime armi", ansiosi di mettersi alla prova.
Drupal è un software di ultima generazione e prettamente di alto livello: fa utilizzo di tecnologie avanzate per la differenziazione dei suoi contenuti come RSS, e sta attualmente sviluppando estensioni per l'integrazione con una sintassi RDF, presupposto indispensabile per l'integrazione del Semantic-Web.

Un qualsiasi utente del web, chiamato in questa dissertazione Daniela, può liberamente scaricare l'ultima versione del programma Drupal e poi installarla, iniziare a studiarla e creare nuovi sviluppi o contenuti su di essa. Da questa possibilità fornita a tutti gli utenti Daniela, sono nati centinaia di moduli, o per spiegarci meglio plugin, che possono essere aggiunti al programma Drupal per poterne ampliare funzionalità e potenzialità.
Tutto lo sviluppo che ha accompagnato la crescita di questo software ha portato al fatto che Drupal dal novembre 2009 sia stato adottato per gestire i contenuti del sito della casa bianca: http://www.whitehouse.gov/.

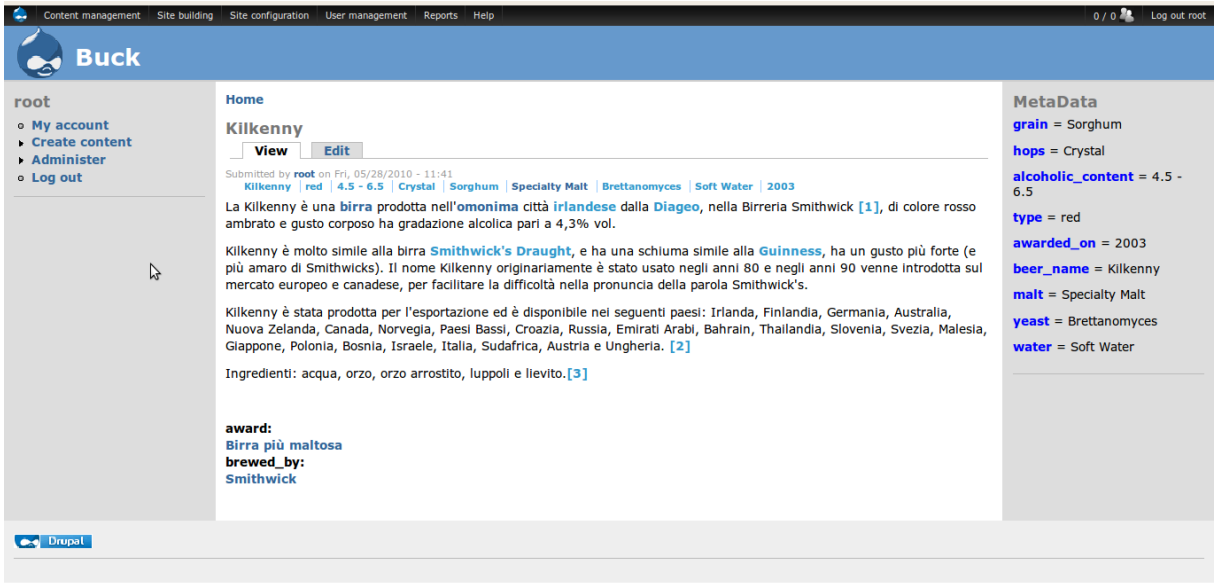
Gran parte del codice Drupal è scritto nel linguaggio di scripting php, attualmente uno dei più usati per la realizzazione di applicazioni web lato server [PHPDOM] .
Gli altri linguaggi presenti nel sistema sono javascript per la realizzazione di script, HTML e CSS per la stampa del contenuto e impostazione dell'output. Il linguaggio SQL viene invece utilizzato per la formulazione di interrogazioni al database su cui il sistema poggia.
Le grandi quantità di dati che il sistema richiede per il suo funzionamento rendono infatti necessario l'utilizzo di una tecnologia efficiente per l'accesso e scrittura dei dati. Drupal inoltre nasce come un sistema per la gestione dei contenuti condivisa. Diversi autori nello stesso istante potrebbero decidere di operare una modifica concorrente agli stessi dati. L'utilizzo di un database per l'archiviazione degli stessi, risolve automaticamente tutti i problemi di concorrenza nell'accesso e modifica delle informazioni e dell'integrità del sistema in caso di crash.
Inoltre, un semplice backup del database rende possibile esportare il proprio sito e cambiarne il server di residenza.

Nel sistema Drupal si ottiene la creazione di un contenuto, cioè la pubblicazione di pagine web, selezionando un determinato content-type da cui crearlo. Un content-type è un template, precedentemente settato rispetto alla creazione del contenuto in questione, che fornisce all' Authenticated user, cioè Carlo, il forum di creazione della tipologia di articolo che gli interessa. I content-types sono quindi gli scheletri degli articoli che gli utenti andranno a creare. Ne sono la base, poiché stabiliscono quali saranno le informazioni e i contenuti che essi potranno presentare.
L'utente Carlo quindi, ogni volta che vorrà creare del contenuto, dovrà scegliere quale tra i content-type a disposizione sia quello più propenso a soddisfare le sue necessità: quello che fornisce la struttura dati che più assomigli al suo contenuto.
Di default in Drupal esistono content-types per la creazione di pagine statiche, blog, history, forum, ecc..
La ragione per cui esistano diversi template per la creazione di contenuto è

molto semplice: tutti i content-types hanno funzionalità diverse, e le meta-informazioni e fields che i template relativi alla creazione di contenuto richiedono all'utente Carlo, cambiano proprio in base al tipo dell'articolo che l'utente creerà. Le informazioni supplementari richieste al creatore dell' articolo quindi, specifiche per ogni content-type, garantiscono una funzionalità chiave: la creazione di meta-dati, ovvero dati relativi ad altri dati, riguardanti l'articolo stesso. In ogni content-type è quindi possibile specificare una particolare lista di quali meta-dati il template possa dare la possibilità di definire. Questi meta-dati potranno essere di utilizzo nella ricerca, categorizzazione e catalogazione dello stesso articolo sia all'interno del CMS, sia all'interno dell'intero WWW.

Il sistema Drupal, non solo permette la scelta tra vari template, ma lascia all'amministratore del sistema, l'utente Alberto, la libertà di creare nuovi template contenenti relative informazioni del tutto personalizzate. In ognuno di questi possono essere inseriti diversi fields, e cioè campi dove poter specificare dei dati, implementati per via dei widgets più vari.
La promozione e pubblicazione di pagine internet realizzate tramite Drupal deve forzatamente passare per questa modalità: non esistono altre metodologie se non la creazione di nuovo contento selezionato dalla lista dei content-types.

3.3 Buck



3.3.1 Il problema

Buck è il sistema che ho sviluppato utilizzando il sofware Drupal: potremmo quindi, per fare un riferimento a quanto appena detto sul CMS, dire che Buck è un nuovo plugin realizzato per Drupal.
Si tratta di un modulo di supporto per l'utente amministratore del sistema, chiamato Alberto, che lo aiuta nella configurazione ed espansione dello stesso.
E' l'amministratore colui che decide i funzionamenti del sistema, cosa questo possa o non possa fare. Ed è l'amministratore che decide quali siano le funzionalità del sistema: sta quindi a lui decidere quali siano i plugin da abilitare e quali invece disabilitare.
Abbiamo spiegato in precedenza come la creazione di un articolo su Drupal possa avvenire soltanto tramite la selezione tra uno dei content-type disponibili.
Può però succedere che un Authenticated user Carlo, voglia creare una tipologia di articolo (i.e. Vino) relativamente al quale non sia presente il template di crezione relativo.
E' quindi compito all'amministratore Alberto creare un nuovo template e personalizzare i fields e meta-dati relativi ad esso a suo piacere. Creare un nuovo template su Drupal è una procedura né troppo lunga né troppo difficoltosa, a

condizione ovviamente che Alberto sia abbastanza esperto nella gestione del CMS. L'amministratore del sistema, una volta informato della volontà di un utente di creare un tipo di articolo, potrebbe ragionevolmente decidere di realizzare un content-type specifico per questo. Tuttavia, lasciando al solo Alberto la personalizzazione di tutti i template del sistema, si incorre nel rischio di una mancanza di campi relativi ad alcune templates. Questi infatti potrebbe, per varie ragioni la più semplice delle quali è la dimenticanza, non inserire nel template di creazione di un articolo informazioni necessarie per la corretta meta-datazione dello stesso.
Non solo quindi Alberto deve provvedere alla creazione di tutti campi del content-type, ma anche alla loro scelta e alla possibile decisione su quali siano i valori assegnabili a tali campi. Oltre a rappresentare una grande quantità di lavoro complessivo, pensiamo ad un template di creazione riguardante le birre e al numero di malti esistenti al mondo da inserire tra i valori possibili: questi sono tantissimi. Il ruolo di amministratore implica quindi una altra grande responsabilità: la creazione di fields e quindi la possibilità di creare meta-dati relativi ad un articolo non può e non deve essere incompleta. Se viene tralasciata la definizione di importanti meta-dati infatti, si incorre nel rischio che gli articoli creati siano incompleti di informazioni e keywords e non vengano poi referenziati correttamente dai motori di ricerca.

3.3.2 La soluzione

Buck offre all'amministratore una pratica soluzione a questo doppio problema: la creazione di templates, o nel caso risulti più chiaro content-types, e di informazioni aggiuntive relative ad essi tramite l'importazione degli stessi da un file esterno. Questo file esterno, definitito tramite una sintassi XML, (acronimo di “Extensible Markup Language”), è definitito nel sistema Buck in una classe di una ontologia. Tale classe si preoccupa di implementare una categoria, una istanza della realtà su cui possano essere creati articoli e recensioni: un nuovo content-type. Nella definizione di tale classe, sono presenti le definizioni di tutti i meta-dati assegnabili alla sua relativa istanza dagli utenti e le restrizioni sui loro valori ammissibili. Buck in sostanza, permette ad Alberto di aggiornare

velocemente e esaustivamente il suo sistema, inserendo in esso un nuovo content-type: un content-type definito tramite una classe di ontologia creata da persone specificatamente incaricate di farlo perchè competenti su di essa. Utilizzando il nuovo content-type, appena creato tramite Buck, sarà allora possibile assegnare tutte le meta-informazioni e i collegamenti che la classe dell'ontologia definisce. Questo è stato reso in parte possibile grazie a particolari plugin di Drupal abilitati: la realizzazione di Buck si basa su molti moduli aggiuntivi a disposizione del CMS Drupal. Questi moduli hanno reso possibile realizzare widgets appropriati, così da mettere in condizione l'utente Carlo di svolgere un facile inserimento di valori inerenti durante l'esecuzione del forum per la creazione di nuovi articoli.

3.3.3 I vantaggi di Buck

I campi-dati relativi ad ogni content-type, assegnabili come meta-informazioni, vengono salvati da Buck come vocabolari Drupal appartenenti al content-types creato. Un vocabolario è composto da un insieme di termini (semplici stringhe) tutti relativi ad un unico concetto(i.e. per il vocabolario beer_type avremo termini come “lager”,”ale”,”stout”, ecc...)[BIR]. I valori possibili, o le restrizioni che codeste meta-informazioni devono rispettare, vengono appunto realizzate nel sistema come termini e regole del vocabolario stesso. La scelta di adoperare questo tipo di salvataggio porta un ulteriore sviluppo : vengono create automaticamente metodologie di accesso tra articoli differenti aventi campi di meta-datazione con ugual valore. Questo può avvenire solamente tra articoli appartenenti allo stesso content-type, in quanto un vocabolario, secondo le regole di Buck, appartiene ad uno solo di essi.

Quando navighiamo su un articolo di Buck e clicchiamo sul valore di un suo meta-dato, abbiamo accesso ad una pagina di definizione di tale meta-dato, e quindi termine di un vocabolario. In tale pagina, oltre ad essere possibile effettuare una descrizione del meta-dato stesso, vengono visionate e referenziati tutti gli articoli che lo condividono.

Il sistema cambia inoltre, rispetto a quanto avverrebbe utilizzando la modalità proposta da Drupal, il layout delle meta-informazioni relative ad ogni articolo.

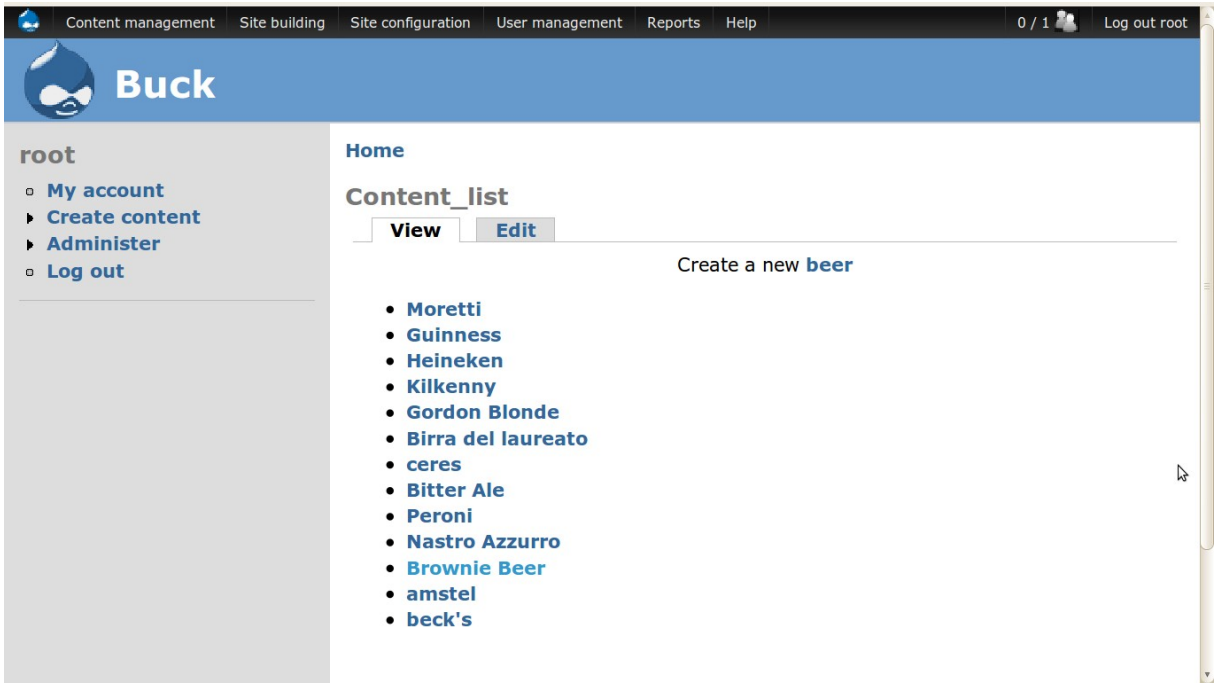
Buck definisce una propria metodologia per la visualizzazione delle meta-informazioni definite. Queste vengono infatti stampate in una “scatola” situata sul lato destro della pagina visualizzante l'articolo, a fianco del corpo dello stesso. Questa diversa metodologia di visualizzazione viene attuata per potere risaltare l'importanza e il significato delle meta-informazioni definite. E' importante infatti che qualsiasi utente che stia consultando l'articolo possa immediatamente ottenere informazioni riguardanti il contenuto, senza la necessità di dovere leggerlo per intero.

Buck realizza inoltre una interazione bilaterale tra se stesso e le classi di ontologia che importa: tramite il sistema infatti, è possibile modificare dati presenti nel file di origine della classe dell'ontologia. Nel CMS Drupal ogni content-type è modificabile da Alberto o al limite da Barbara, l'utente developer. Questi utenti possono quindi modificare fields, valori ammissibili, restrizioni sulla quantità di valori assegnabili e qualsiasi altro dato relativo ad ogni content-types.
Utilizzando Buck però, Alberto e Alice, non devono porsi la problematica di dovere riportare, documentare e successivamente trascrivere tutti i nuovi valori-modifiche che vengono portati nella definizione di content-type creati tramite Buck: i nuovi valori e le modifiche proposte(secondo i criteri dell'ontologia) vengono automaticamente modificate nel file xml che definisce l'ontologia.

3.3.4 Interfaccia Utente

L' attuale installazione di Buck è accessibile all'indirizzo internet www.malfattitesi.web.cs.unibo.it.
Per potere invece contribuire alla crescita del sito aggiungendo o modificando del contenuto esistente, è necessaria la registrazione al sistema stesso: registrazione che deve essere autorizzata dall'amministratore del sistema: ovvero me medesimo.
Collegandovi al sito per la prima volta potrete notare la richiesta di richiesta di

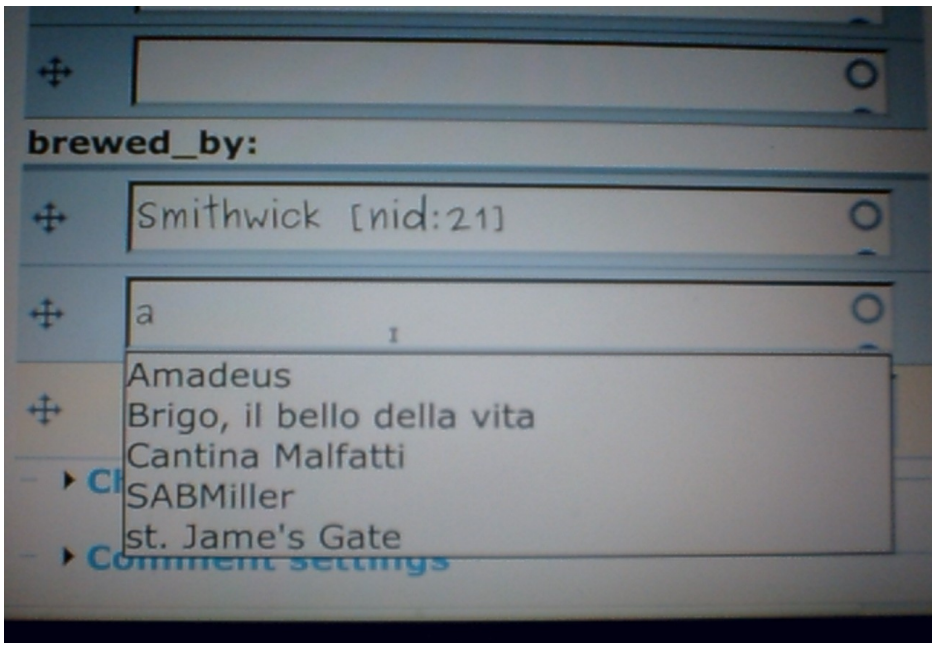
login che appare sulla sinistra dello schermo. L'autenticazione non è necessaria per la navigazione del sito stesso, ma lo è per lo sfruttamento del funzionamento del sistema.
La pagina iniziale del sito offre l'accesso a tutti gli articoli pubblicati, catalogati tramite content-types. Per ogni content-types, viene anche indicato quanti siano gli articoli attualmente presenti nel sistema.



Selezionando un content-type e successivamente una sua istanza
(i.e. beer->Ceres), è possibile accedere alla visualizzazione di un articolo. Nella pagina l'utente Daniela troverà visualizzati in altro a destra, se presenti, i meta-dati relativi. Alla fine della descrizione relativa all'istanza visualizzata, se definiti saranno visibili link ad altri articoli di diversi relativi a diversi content-types, con i quali l'articolo presente è in relazione.

Nel caso l'utente Daniela effettui il login e si trasformi in utente Carlo, sarà per lui possibile vedere il tasto edit, e passare quindi alla fase di modifica delle informazioni relative alla pagina in visualizzazione. In tale pagina l'utente Carlo ha la possibilità di modificare qualsiasi informazione che ritenga errata o aggiungerne di mancanti.
L'utente autenticato Carlo ha inoltre accesso, da ogni pagina del sistema, alla sua

area personale, situata sulla sinistra dello schermo. In questa area egli può cambiare proprie impostazioni personali, il tema utilizzato per la visualizzazione del sistema e creare un nuovo contenuto ed effettuare l'uscita dal sistema.



3.3.5 Interfaccia Admin

Nella corrente installazione di Buck sono presenti due menù amministratore. Il primo, espandibile, è situato nell'area personale di ogni utente, il secondo, funzionante tramite javascript, in alto a sinistra.

Per accedere all'importazione di un nuovo content-type tramite Buck, l'amministratore Carlo dovrà navigare il menù seguendo i seguenti passi: Content Managment – Content-types – Class Import.

Qui una schermata offrirà a Carlo una lista di file.xml dal quale selezionare quello relativo al content-type da creare. Avanzando nel menù, Carlo si ritroverà in una seconda schermata. Da qui, a meno che non voglia cambiare impostazioni al nuovo content, dovrà solamente schiacciare il tasto Submit.

Per potere poi importare vocabolari e termini relativi al content-type appena creato, Carlo dovrà seguire il seguente path: Site Building – Import.

Da qui iniziare l'importazione dei meta-dati utili al sistema. Durante la prima schermata è necessario solezionare i valori “terms” e ”vocabulary” tra le Entities to Import e il valore xml nel formato del file. Nella pagina successiva invece,

Alberto dovrà scegliere, tra una lista, il file corretto da cui importare tali dati: il file corretto è ovviamente quello che detiene lo stesso nome del content creato.
Per abilitare la visione del box contente i meta-dati invece, l'amministratore dove recarsi nella gestione dei blocchi del sistema, tramite il path Site Building – Block – List.
Un blocco è una funzione che produce layout attivabile dall'amministratore. Ne esistono parecchie in Drupal. La mia decisione di implementare la visualizzazione dei meta-dati in questo modo è molto semplice: in questo modo il blocco sarà attivabile qualsiasi sia il tema di visualizzazione prescelto. Se avessi deciso di implementare il box all'interno della visione di un tema, questo non sarebbe stato visibile utilizzando un qualsiasi altro theme.
Dalla lista di blocchi è sufficiente ricercare il blocco relativo(i.e. Metadata), caricarlo nella zona desiderata e attivarlo.

Capitolo 4

Architettura del sistema Buck

4.1 Architettura di Drupal

In questa sezione del mio elaborato vengono presentate l'architettura di Drupal e Buck: alcuni meccanismi e funzionamenti base del CMS, le mie modifiche ed aggiunte che hanno portato alla creazione di Buck.
Innanzitutto rendiamo noti quali siano i requisiti minimi per la programmazione del sistema: Drupal è scritto quasi totalmente in php, e quindi la sua conoscenza è assolutamente necessaria. Php ha una sintassi molto simile a quella del linguaggio C, e quindi dovrebbe essere facilmente comprensibile a coloro che abbiano avuto esperienze di programmazione.
Altri linguaggi che sono stati utilizzati sono : javascript, HTML e CSS per la realizzazione dell'output, SQL per le interrogazioni al database.
Oltre ad una minima conoscenza di questi linguaggi, requisito tecnico fondamentale per "mettere le mani" all'applicazione Drupal, è una discreta abilità nella lettura e comprensione della lingua inglese. La documentazione reperibile in italiano è talmente scarsa che non talvolta, bensì spesso, l'unica fonte di approfondimento consultabile è il sito ufficiale del portale, ovviamente scritto in lingua inglese[Drup10].

4.2 Caratteristiche di Drupal

La prima caratteristica utile da sapere riguardo a Drupal è la seguente: tutto è un "nodo". Un nodo in una struttura a grafo o ad albero è un elemento del sistema raggiungibile attraverso un determinato cammino. In Drupal, ogni contenuto, dall'articolo al termine della tassonomia, è definito come nodo e come tale sarà identificato da un suo codice univoco che ne permetta il raggiungimento.

4.3 Il funzionamento di un modulo: gli hook

Drupal ha rappresentato nella mia breve esperienza di programmatore, il primo e probabilmente unico confronto con un applicazione framework, dotata quindi di una grande quantità di metodi specifici propri, insiemi di regole e prassi da rispettare, ed una forte modularizzazione del codice. L'utilizzo di questo tipo di applicazioni riduce notevolmente il lavoro del programmatore, perchè gli risparmia la riscrittura di centinaia di metodi che sono già stati implementati da altri.
Il sistema è organizzato in moduli ed ognuno di questi ha una sua precisa funzione. Pertanto ogni modulo detiene sia proprie caratteristiche sia permessi propri e per quanto i vari moduli cooperino tra loro, ognuno di questi rappresenta una parte a se stante del programma. Per essere attivato, ogni modulo deve per essere riconosciuto dal sistema: i suoi files devono essere situati in una unica cartella nominata con il nome del modulo stesso e localizzata nella directory “modules”. Ogni cartella deve contenere almeno due files. Il primo moduleName.info, informa il sistema dell'esistenza del modulo stesso e gli fornisce alcune informazioni base. Il secondo moduleName.module contiene il codice effettivo che viene chiamato quando viene richiesta una funzionalità implementata dal modulo. Ogni altro file presente sarà richiamato dal file moduleName.module.

Un hook, termine inglese traducibile in italiano con “uncino”, è una funzione di php utilizzata per estendere la funzionalità del modulo dove l'hook stesso è stato implementato. Tutte le funzioni all'interno di Drupal vengono definite con questo metodo. Solitamente un hook definito può essere richiamato in ogni parte del codice. Una volta invocato, l'hook verrà ricercato tra tutti quelli esistenti nei moduli attivi del sistema. Esistono allo stesso tempo, hook dichiarati e scritti in un modulo che non possono essere riutilizzati per la implementazione di altri, e quindi chiamati fuori dal modulo di appartenenza. Affinché Drupal sia a conoscenza di ogni funzione presente nel sistema, è necessario creare un hook che permetta di definire un’associazione tra un’URL(cioè di una univoca risorsa

di internet) e la funzione incaricata di creare il contenuto per l'URL stesso: tale hook è definito in “moduleName_menu()”. L'hook_menu() è presente in ogni modulo e svolge proprio la funzione di attivare tutte le funzionalità che il modulo stesso implementa. Tramite questo hook il modulo, che in sua assenza non funzionerebbe, si collega a tutti gli altri moduli del sistema dando quindi vita al core di Drupal.

L'hook moduleName_block(), anch'esso allo stesso modo di moduleName_menu(), è un hook dichiarato di default: si preoccupa della creazione del blocco relativo al modulo che stiamo creando. Un blocco è una scatola(o box) contenente dell'output che viene stampata dal sistema nelle pagine di navigazione del sito. Non è necessario creare un blocco per ogni modulo che stiamo scrivendo, ma il codice utilizzato sarà probabilmente molto simile se non quasi totalmente lo stesso(se ben strutturato) al codice relativo alla implementazione del modulo. Creare un blocco ogni volta che se ne presenti l'occasione, accresce sicuramente le funzionalità del nostro sistema: la funzionalità del blocco è quella di offrire determinate informazioni in maniera rapida e precisa oppure, di fornire scorciatoie per l'esecuzione di operazioni ricorrenti. A volte può rivelarsi non necessario, altre sicuramente si. Nei paragrafi a seguire, verrà offerto l'esempio di un modulo creato esclusivamente per ottenere la visione di un blocco.

4.4 I moduli aggiuntivi

Quando un utente scarica una qualsiasi versione di Drupal la prima operazione dà fare è accedere al menù di amministrazione e iniziare la configurazione della parte relativa alla gestione dei moduli. In questa pagina sono elencati tutti i moduli al momento disponibili nel sistema. Dopo l'installazione solo alcuni di questi sono abilitati: la gran parte di essi è disponibile ma non ancora funzionante. Un modulo per poter essere utilizzato deve precedentemente essere stato inserito e abilitato. L'abilitazione avviene proprio in questa pagina di gestione dei moduli da parte dell'amministratore.

I moduli sviluppati per Drupal sono centinaia, probabilmente migliaia. Gran parte del lavoro dello sviluppatore di servizi tramite Drupal consiste proprio nel testare decine di moduli sino a trovarne uno che implementi nella maniera più completa possibile i servizi richiesti. Da qui il programmatore può partire e personalizzare il modulo per arrivare al risultato finale richiesto.
In rete sono disponibili diverse recensioni dei moduli: varie graduatorie che suggeriscono quali moduli siano migliori di altri. Nel mio lavoro di tesi ne ho sperimentati parecchi e darò descrizione ora di quei moduli aggiuntivi, segnati dal segno **(A),** o base, segnati dal segno **(B),** che ho trovato utili per la realizzazione di Buck.

Content Creation Kit(A)

Questo modulo permette la creazione di nuovi content-type: per ognuno di questi che viene creato il sistema genererà un diverso template per la sottomissione degli articoli. E' inoltre possibile personalizzare i content-types tramite la creazione di fields personalizzati, atti ad aumentare le informazioni assegnabili agli stessi.

Tassonomia(B)

Permette la creazione di vocabolari e termini della tassonomia.
Un vocabolario è un insieme di termini caratterizzato da certe regole. I termini sono invece, per il sistema, stringhe di lunghezza variabile. Su ognuno di questi termini è possibile creare una descrizione ed una discussione da parte degli utenti. I termini possono essere messi in relazione gerarchica tra loro tramite l'utilizzo di definizioni di tassonomie e tesauri.
Questo modulo, sicuramente molto intuitivo, è uno dei più utilizzati nei sistemi Drupal. Grazie a questo possiamo creare qualsiasi tipo di meta-informazioni ed ordinarle tra loro in maniera gerarchica.

Administration Menù(A)

Crea automaticamente un menù di rapido accesso delle impostazioni per l'amministratore del sistema. E' implementato tramite javascript.

User Reference, Option widgets, Node Reference, Fieldgroup, NodeRefCreate, Select(or other), Taxonomy Other.(A)
Tutti questi moduli integrano le funzionalità dei moduli CCK e Tassonomia. Rendono possibile una ampia selezione di valori e tipi da attribuire ai vari widget e migliorano l'interazione tra utente e sistema aumentando notevolmente le funzionalità di quest'ultimo.

Pathauto(A)
Modulo che gestisce automaticamente il path dei contenuti creati. Non utilizzando questa feature, ogni articolo da noi creato sarebbe automaticamente referenziato tramite un link numerico. Cosi invece il sistema assegna alla pagina un link il più possibile somigliante al titolo della pagina stessa. Permette in alternativa di assegnare ad ogni articolo un path relativo qualsiasi.

ImportExportAPI(A)
Modulo che permette di importare ed esportare qualsiasi tipo di contenuto e settaggio dal e nel sistema tramite l'utilizzo di file con sintassi xml. E' molto importante utilizzare ed ampliare questo modulo se si vuole costruire un sistema integrato con altri. Alcune sue parti non sono ancora funzionanti.

4.5 Uno sguardo all' output

La visualizzazione di tutti i contenuti di Drupal passa per il funzionamento dei theme del sistema. Un theme è insieme di diversi files che contribuiscono alla organizzazione e presentazione della stampa dell'output del sistema. I vari temi disponibili e selezionabili dagli utenti sono organizzati come sottocartelle della directory “theme”. In ognuna delle cartelle che implementa i vari temi sono presenti i diversi files per la configurazione dell' outuput finale. E' molto raro che due theme differenti condividano qualche file di configurazione: solitamente, ognuno implementati propri Ecco di seguito elencati i più files importanti

- **page.tpl.php**: il template principale che definisce il contenuto di ogni pagina: tutto ciò che in ogni pagina deve essere stampato
- **node.tpl.php**: questo file definisce il contenuto di ogni singolo nodo. Divide la visualizzazione della pagina in regioni. Ad ogni regione sarà poi possibile assegnare blocchi diversi.
- **block.tpl.php**: definisce il contenuto e la visualizzazione dei blocchi.
- **comment.tpl.php**: definisce il contenuto dei commenti e setta i parametri con i quali debbano essere visualizzati.
- **style.css**: il foglio di stile principale del tema. In questo file possono essere cambiate le posizioni dei all'interno delle regioni dove sono stati assegnati.
- **template.php**: questo file permette di intervenire più a fondo nel tema, ad esempio aggiungendo ulteriori regioni, riscrivendo le variabili di default e altro.

4.6 L'architettura di Buck :Homepage e Content_list

Quando diamo origine ad articoli su Drupal è possibile utilizzare codice HTML e php per la realizzazione del loro contenuto. Ovviamente l'utilizzo di script deve essere riservato solo a sviluppatori ed amministratori del sistema.
Nel sistema Buck ci sono due pagine contenenti codice scritto in linguaggio php: HomePage e Content_List. Su di esse infatti girano due script di interrogazione al database per la creazione dinamica di link agli articoli presenti nel sistema. HomePage esegue uno script che ricerchi tutti i content-types presenti su cui possa essere creato del contenuto. In questo modo l'utente si trova dinnanzi alla lista dei content-types gestiti dal sistema che possano essere visionati, modificati o creati. Tutti i link di HomePage conducono sempre alla pagina Content_List, passandole un argomento tramite metodo get: il content selezionato dall'utente. Content_List legge il content passatole e tramite interrogazioni al database stampa una lista delle istanze presenti sul sistema del determinato content ricevuto. Da questa pagina è anche possibile iniziare la creazione di una nuova istanza, tramite un link al form di creazione della stessa.

4.7 Il file read.php come estensione del modulo cck

La funzionalità importClass descritta nel secondo capitolo, non è presente nel core di Drupal: è una opzione che ho creato estendendo il codice del modulo CCK. Per realizzarla ho dovuto modificare un file esistente (/cck/modules/content_copy.module) e realizzarne un altro di libreria(/files/read.php).

Modifiche al modulo

Di seguito riporto le principali modifiche apportate a content_copy.module.

F = function

F content_copy_menu() : viene dichiarato ImportClass come funzione del menù e associa il suo contenuto all'argomento delle drupal_get_form()

F content_copy_import_form2(): prima pagina del form a due livelli per la sottomissione di un nuovo content-type. Con l'ausilio di due funzioni ausiliarie elenca e stampa i files xml dai quali è possibile importare i content.

F content_copy_import_next(): gestisce la seconda parte del form precedente. Stampa l'array php prodotto in una text-area e ne propone l'inserimento in diverse modalità. Chiama il modulo read.php tramite la funzione content_copy_cClass()

F content_copy_import_form2_submit(): è la funzione che viene chiamata al termine dell' eseguzione di ognuna delle due precedenti. Cambia certi parametri di $form_state per potere gestire come unico il form creato a due livelli. Gestisce quindi il ritorno del form stesso. Funziona in maniera totalmente diversa a seconda di quale delle due funzioni ne abbia scatenato la chiamata.

4.8 Read.php

Qui risiede il vero cuore della funzionalità ImportClass. Le modifiche apportate al file content_copy.module servono solamente per la sottomissione di array di array php che implementino il nuovo content. All' interno di questo file avviene la creazione di tali array e del file xml che conterrà vocabolari e termini dell'articolo.
La prima funzione che viene chiamata all'interno del file è content_copy_cClass(). Questa, funzione madre da cui partono le chiamate a tutte le altre, si preoccupa di caricare il file xml di input contenente i dati del nuovo content-type; istanziare un nuovo file xml dove verranno scritti, secondo la sintassi di Drupal, vocabolari e termini da importare successivamente; dichiarare tutte le variabili globali utilizzate nel modulo e mettere insieme i vari array di stringhe che comporranno alla fine gli array php contenenti il nuovo content-type.
Fase cruciale del funzionamento di content_copy.module è la lettura del file xml sorgente. In questo file, oltre a parametri vari, la funzione deve ricercare tutti i fields che faranno parte del content-type. Una volta trovatili passa la gestione a due altre funzioni: get_content e connectfield.

F connectfield() gestisce tutti quei field che rappresentano dei collegamenti ad altri content-types. La funzione ha il compito di restituire parametri necessari alla definizione di quali nodi potranno essere connessi al nodo corrente. Restituisce un pezzo della stringa di definizione del content.

F get_Content()
Questa funzione invece analizza tutti i fields che non rappresenteranno collegamenti con altri nodi. Sono fields da cui verranno creati i vocabolari e i relativi termini. Il file xml sorgente specifica caratteristiche di questi campi: liste di valori possibili, range da rispettare, numero di valori selezionabili minimo e massimo. Tutte queste impostazioni vengono riscritte in un nuovo file xml che, rispettando una sintassi interpretata da Drupal, permetta a questi fields di essere

importati come vocabolari e termini di tassonomia. La funzione quindi non contribuisce direttamente alla creazione del content-type, ma crea i suoi meta-dati relativi.

F valueModify($arg), F parse($arg)
Servono per la rimozione di caratteri potenzialmente non interpretabili dal database.

4.9 Metadata module

Metadata è il nome del modulo che ho creato per facilitare la visualizzazione delle meta-informazioni riguardanti gli articoli.
Il sistema Drupal offre già diverse metodologie per la visualizzazione di tali meta-dati (che ricordiamo essere termini di un vocabolario). La mia volontà però era quella di creare una area allineata al testo ma allo stesso tempo staccata ed indipendente da questo. Per ottenere questa soluzione avrei potuto modificare i files CSS e page.tlp.php del theme per la visualizzazione dei termini della tassonomia. Il problema è che tali files cambiano per ogni theme e quindi non avrei avuto una soluzione portabile del problema.
Viste queste difficoltà nel trovare una soluzione definitiva al problema ho optato per una risoluzione alternativa, decidendo di implementare un nuovo modulo con l'obbiettivo di creare un blocco per la visualizzazione dei meta-dati:
"_metadata_block_content()" è la funzione principale del modulo, che gestisce la stampa delle meta-informazioni.
In questa funzione troviamo subito un controllo che stabilisce in che tipo di pagina l'utente stia navigando: se si trova in una pagina node(e quindi in un articolo di qualsiasi genere) allora il blocco deve essere visualizzato e l'esecuzione del modulo può andare avanti, se si trova in una pagina di edit delle impostazioni, no. Successivamente tramite la chiamata arg() il modulo ottiene informazioni su quale sia il nodo di visualizzazione attuale. Nel seguente ciclo while il programma, attraverso vari accessi alle tabelle del database, ottiene informazioni su quali siano gli eventuali valori assegnati ai meta-dati

dell'articolo corrente. Il risultato di queste informazioni viene scritto tramite codice HTML nella variabile di ritorno $block_content, ritornata e successivamente stampata.

4.10 Sincronizzazione del content-type con xml relativo

Abbiamo già spiegato come i vari content-types siano stati importati da file xml relativi a classi di istanze di una ontologia. Questi files possono essere editati in minime parti dal sistema implementato.
Tali modifiche avvengono quando vengono editate alcune impostazioni(i.e. il nome) dei fields o dei vocabolari relativi ai vari content utilizzando l'applicazione, così da ottenere una certa sincronizzazione tra i files sorgenti e le modifiche che apportiamo nel sistema stesso.
I cambiamenti riportati nel codice per permettere queste operazioni sono i seguenti e riguardano i seguenti files:
/taxonomy/taxonomy.admin.inc
/cck/includes/content.admin.inc
In entrambi viene chiamato l'hook _field_change_XML($arg1,$arg2,$arg3) che permette l'aggiornamento del file sorgente xml. La funzione rimuove il nodo del file contenente l' informazione originaria per sostituirla con la nuova appena impostata dall'utente nel sistema. Nel caso il file originario sia stato rimosso la sua mancata modifica viene notificata all'utente tramite un messaggio di errore.
Attualmente questo sistema permette solamente l'aggiornamento dei fields: uno sviluppo sicuramente interessante sarebbe quello di permettere a ogni impostazione modificata all'interno del sistema, di essere resa tale anche nel file sorgente.

4.11 Files supplementari

Ecco una breve lista di files e supplementari utilizzati.
La cartella /files contiene tutto il materiale aggiuntivo al funzionamento di del sistema. In essa troviamo:

data.xml, file di supporto per read.php

/types, cartella contente i content type aggiunti o che è possibile aggiungere.

Capitolo 5

Valutazione

5.1 I problemi affrontati da Buck

Come già discusso nei precedenti capitoli, non sono poche le problematiche che si incontrano nella definizione dei meta-dati.
Inanzi tutto è necessario scegliere quali debbano essere i meta-dati relativi ad un tipo di contenuto Web che debbano e possano essere definiti. Si pone quindi il problema di come implementare la realizzazione di tali meta-dati: nella creazione di un sistema che ne permetta la definizione, bisogna stabilire a chi spetti la decisione di scegliere quali siano i meta-dati possibili da assegnare: utenti o amministratore.

Per potere realisticamente dare un valore al contributo del sistema Buck per la comunità scientifica ci apprestiamo a discutere i maggiori problemi evidenziati nella realizzazione di meta-dati inerenti al testo. I parametri utilizzati per poter esprimere un giudizio finale saranno i seguenti: semplificazioni e migliorie che il software offre nell'amministrazione di un sistema, agevolazioni percepite dall'utente e l'integrazione con le moderne tecnologie del Web.

5.2 Da una a N scelte in meno per l'amministratore

Inizialmente l'amministratore del sistema contenente gli articoli di interesse deve scegliere un metro per la meta-datazione. E' infatti auspicabile che ogni articolo abbia potenzialmente, riferite a se stesso, le stesse quantità di informazioni di ogni altro. Immaginiamo di consultare un portale dove una certa quantità di articoli sia classificata da una vasta lista di meta-dati ed un altra da una lista molto inferiore. L'utente che naviga il portale avrà la percezione che alcuni articoli in consultazione siano stati curati diversamente e che quindi il loro

contenuto possa essere, in qualche modo, poco qualificato.
L'amministratore inoltre deve scegliere come i meta-dati debbano essere creati: le principali soluzioni che abbiamo incontrato nel corso del secondo capitolo sono sostanzialmente due: o l'amministratore stesso sceglie quali siano i meta-dati creabili attorno ad un contenuto, o lascia l'arbitraria scelta di questi agli utenti. Entrambe queste soluzioni hanno però delle controindicazioni.
Nel caso in cui l'amministratore scelga di essere lui stesso a volere fornire agli utenti i campi da riempire per potere realizzare meta-informazioni, sarà suo compito provvedere alla implementazione di tutti i meta-dati relativi. Ogni CMS permette, ed in ogni caso consiglia, la creazione di contenuti Web attraverso la selezione di un tipo di contenuto. I meta-dati quindi, per potere descrivere esaustivamente e precisamente gli articoli, devono essere creati relativamente ad ogni tipo di contenuto. In un sistema di piccole dimensioni questo non rappresenta un grosso problema. Se però prendiamo in considerazione portali di larga scala, con al loro interno definizioni per decine di tipi di contenuto, il lavoro dell'amministratore aumenta esponenzialmente. Non solo cresce il lavoro manuale di definizione di meta-dati, ma tale lavoro implica da parte dell'amministratore per potere arrivare ad una definizione completa degli articoli, anche una vasta conoscenza di ogni tipo di contenuto di interesse. Per ogni meta-dato inserito, egli dovrà anche selezionare quali tipologie di valori potranno essergli assegnati, quanti valori potranno essere inseriti, ecc .
Scegliendo la seconda opzione invece l'amministratore lascia al creatore di ogni articolo la decisione di quali meta-informazioni creare al riguardo. Questa soluzione comporta rischi ancora peggiori: utenti diversi potrebbero definire lo stesso tipo di meta-dato(i.e. una data) utilizzando però label diverse per la sua identificazione: nel linguaggio naturale le stringhe “data di nascita” e “giorno di fondazione” hanno pressapoco lo stesso significato, ma appaiono inequivocabilmente come due dati diversi. La standardizzazione della visualizzazione e presentazione di dati all'interno di un sito internet, ne migliora senza alcun dubbio la consultazione da parte degli utenti e la valutazione che questi avranno su di esso.
L'utilizzo dell'installazione Buck invece, risolve brillantemente questo problema.

Come abbiamo visto in precedenza, Buck offre all'amministratore di un portale la possibilità di importare come Content-Types, classi di una ontologia definita.
Una ontologia è nell'ambito informatico un'esplicita e formale specificazione di una concettualizzazione . Questa descrive il dominio del discorso mediante l'utilizzo di classi di oggetti (che per fare un riferimento noto potremmo chiamare content-types), relazioni tra classi, restrizioni, proprietà e affermazioni sulle istanze delle classi[VIT].
Questo significa che un' ontologia crea di per sé una sua visione della realtà. Per ogni ambito od oggetto definito dall'ontologia, questa provvederà a creare una un insieme di informazioni univoco dedito alla descrizione di tale oggetto. Buck sfrutta tutte le proprietà che un' ontologia definisce per determinare relazioni e meta-informazioni che un nuovo Content-Type dovrà implementare.
L'utilizzo di una ontologia per la definizione dei tipi di contenuto, e quindi l'utilizzo di Buck, permette all'amministratore con una sola scelta di liberarsi di tutte le problematiche relative alla creazione dei meta-dati.
Sarà infatti il sistema Buck a determinare e implementare quali siano i meta-dati specificabili su ogni content-type, a scegliere quali e quanti valori questi possano assumere e le relazioni con altri content-type.
Per inserire un nuovo tipo di contenuto, all'amministratore del sistema sarà sufficiente importare la classe dell'ontologia relativa. Questi conserva comunque, in qualità di amministratore del proprio sistema, il diritto di non condividere le scelte fatte dai creatori dell'ontologia e di cambiare quindi settaggi a piacere del nuovo content-type.

5.3 Una guida ed uno stimolo per l'utente

Anche per un utente navigatore od utilizzatore di portali, l'utilizzo di Buck può risultare vantaggioso.
Quando un utente crea del contenuto online, non si cura solitamente della creazione di meta-dati riguardanti il suo contenuto. Come scritto nel già citato articolo di Cory Doctorow sulla creazione dei meta-dati le persone sono pigre: sono pochissimi gli utenti che una volta creata una pagina Web hanno "la voglia" di andare a realizzare meta-informazioni su di essa. Questo avviene

probabilmente anche come conseguenza del fatto che la maggior parte degli utenti inesperti del Web non comprende l'importanza della meta-datazione dei contenuti, o più probabilmente, non la conosce affatto.
La creazione di meta-dati è oltretutto spesso complicata. In molte applicazioni e situazioni, come ad esempio i Wiki, i meta-dati devono essere salvati tramite metodologie e sintassi totalmente sconosciute all'utente medio fruitore del Web. E' quindi necessario guidare l'utente durante la formazione di un nuovo articolo, e ribadisco durante e non dopo, in una semplice e veloce ma allo stesso tempo completa creazione dei meta-dati relativi. Se infatti in un sistema progettato per la creazione di articoli e delle relative meta-informazioni queste due fasi fossero divise, buona parte degli utenti rischierebbe di saltare la fase di creazione di informazioni supplementari. E' quindi di estrema importanza che la creazione di contenuto e meta-dati relativi avvenga allo stesso momento e nella maniera più guidata possibile per l'utente. Il sistema Buck compie esaustivamente questo compito.
Quando infatti un utente vuole creare un nuovo contenuto su Buck, il sistema lo "guida" per il completamente di un form "user friendly". Nel form viene ovviamente permessa la definizione del contenuto dell'articolo, il corpo dell'informazione. Al tempo stesso però il creatore troverà a sua disposizione molti altri campi da completare, a seconda di quanti ne siano stati importati dall'ontologia. Questi campi aggiuntivi riguardano per la stragrande maggioranza le meta-informazioni assegnabili all'articolo: l'utente quindi attribuendo valori a questi campi automaticamente crea delle meta-informazioni.
Il sistema si preoccupa di rendere estremamente semplice e veloce questa selezione: infatti per tutti i meta-dati i cui valori sono selezionabili da una lista l'interfaccia permette direttamente la selezione dalla lista stessa. Per tutti gli altri campi per cui non è invece presenta una lista, è comunque in funzione un dispositivo di auto-completamento. Questo significa che appena l'utente comincerà a scrivere un valore, il sistema offrirà una lista di valori plausibili all'utente tra cui poterne selezionare uno. Ovviamente mano a mano che l'utente proseguirà nella scrittura del valore scelto, il campo di scelta offerto dal programma si restringerà.

Un altro vantaggio di cui l'utente può usufruire dall'utilizzo del sistema è rappresentato dall'utilizzo del dispositivo di visualizzazione delle meta-informazioni create a specifica di ogni articolo. Alcuni portali infatti, soprattutto tra i CMS, non rendono disponibile una chiara visione di quali siano i meta-dati definiti sui loro contenuti. Buck al contrario implementa una visualizzazione standard per tutti i meta-dati e localizza questa proiezione di valori sempre nella stessa zona della pagina. Questo avviene per favorire la standardizzazione nella visualizzazione di contenuti ed informazioni relative e per portare quindi l'utente ad una sempre maggiore conoscenza ed utilizzo del sito.

Integrazione con moderne tecnologie e sviluppi futuri.
Come abbiamo già spiegato nel capitolo relativo alla gestione del sistema, l'ontologia utilizzata da Buck per l'importazione di content-types è definita da un documento facente uso di sintassi XML. Il sistema Buck è stato tarato per funzionare esattamente con nodi e attributi di una determinata ontologia. Non sarebbe però difficile estendere le sue funzionalità e rendere il sistema interattivo anche con altre ontologie facenti uso di strutture differenti. Tutto il lavoro necessario a questa evoluzione è l'implementazione di un modulo "ponte" per ogni nuova ontologia utilizzata, che effettui una traduzione tra gli schemi della ontologia base e la nuova. L'ontologia di attuale utilizzo, come già detto, è stata definita tramite una sintassi XML. Questo agevola sicuramente l'eventuale traduzione e lettura delle sue strutture dati verso altri formati di salvataggio. Ovviamente la situazione ideale in cui potremmo trovarci, sarebbe quella di dovere mettere in comunicazione tra loro due documenti XML.

5.4 Giudizio finale

Esprimere un giudizio conclusivo su un lavoro realizzato non è mai semplice, soprattutto se allo stesso tempo si ricopre il ruolo di scrittore e progettista del lavoro stesso.
Per ricapitolare quanto detto sino ad ora: Buck facilita il lavoro dell'amministratore di sistema deresponsabilizzandolo dalla creazione di scheletri per la realizzazione di nuovi tipi di contenuto. Questa funzionalità

nasconde anche il vantaggio di risparmiare una grande mole di lavoro allo stesso amministratore di sistema. Inoltre, a questi è lasciata la possibilità di cambiare tutte le impostazioni settate da Buck sui content-types.
Migliorie sensibili vengono percepite anche dall'utente facente uso del sistema, che durante la creazione del nuovo articolo, viene guidato nella specificazione di meta-dati relativi ad esso. Le meta-informazioni da questo specificate saranno poi chiaramente visualizzate nella pagina relativa all'articolo.
Tutte queste motivazioni spingono a dare una valutazione estremamente positiva al software per due semplici ragioni. La prima è il buon livello di potenzialità che esso offre: tramite questo infatti, l'amministratore può realizzare nell'arco di breve tempo una quantità discreta di nuove tipologie di contenuti. La seconda ragione è la buona interazione che esso offre all'utente: Buck migliora per certi aspetti, le potenzialità offerte dal software WordPress presentato nel secondo capitolo. A differenza di quest'ultimo però, Buck guida l'utente nella creazione dei meta-dati e offre allo stesso una giusta quantità di meta-informazioni definibili.

Capitolo 6

Conclusioni

A conclusione dell'elaborato svolto credo sia utile analizzare in cosa sia consistito il lavoro per la buona realizzazione dello stesso.
Buona parte del tempo necessario alla realizzazione del sistema Buck è stato utilizzato per lo studio del software Drupal. E' stato infatti necessario studiarne il funzionamento di alto livello, la struttura interna e le metodologie di espansione e modifica della stessa. Il mio lavoro è frequentemente equivalso al testing di diversi moduli aggiuntivi attualmente disponibili per l'applicazione, moduli che mi permettessero di sviluppare nuove funzionalità e di capire quali fossero invece i difetti presenti nel sistema. La possibilità di potere svolgere questa attività di ricerca mi ha permesso di venire a conoscenza di alcuni delle funzionalità del sistema Drupal. Credo inoltre, che oltre a mio personale motivo di orgoglio, questo possa esserlo per tutta la comunità scientifica. Questo elaborato rappresenta infatti il risultato del lavoro non di una sola persona, ma la congiunzione degli sforzi di una comunità scientifica che coopera alla crescita ed espansione dei sempre maggiori servizi offerti. La collaborazione e interazione con le varie comunità italiane di sviluppo del software è stato per me un prezioso strumento per la progettazione dei moduli grazie soprattutto al confronto avuto con utilizzatori dello stesso.
Ulteriore motivo di orgoglio per il sottoscritto è stato rappresentato dalla possibilità di effettuare ricerca riguardante un progetto molto attuale nell'informatica odierna. Man mano che progredivo nell'analizzare i problemi derivanti dalla mancanza di meta-dati sui contenuti Web e nella conoscenza del software, ho cominciato a intravedere la possibilità dello stesso per la modifica di progetti personali già realizzarti che mi piacerebbe, grazie alle nuove conoscenze sviluppate, potere modificare(per ulteriori informazioni a riguardo consultare il sito internet www.lepietrevive.altervista.org) .

Il nome del sistema che ho realizzato, come ampiamente chiarito durante la dissertazione, è Buck. Le estensioni possibili per questo acronimo sono due: “Basic User Content Kit” e “Basic Users Cooperation Kit”. I due nomi ci indicano quali vogliano essere le prerogative offerte dal sistema. Inanzi tutto ogni ad utente viene fornito un tool automatico per la creazione di contenuti online. Su tali articoli il Kit Buck permette la specificazione di varie meta-informazioni. L'utente viene quindi creato e agevolato nella creazione di tali contenuti. Il secondo nome attribuibile al sistema, “Basic Users Cooperation Kit”, indica invece la possibilità per gli utenti di condividere le informazioni create tra loro. In questo caso estendiamo il significato della parola condivisione a modifica: ogni utente ha può partecipare alla miglioria del contenuto creati da altri utenti. L'utilizzo iniziale del sistema è quindi quello di dare la possibilità ad ogni utente di creare degli articoli, e termina fornendo a tutti la possibilità di cooperare alla miglioria e all'esattezza dei contenuti stessi.

Questo servizio di cooperazione nella realizzazione di contenuto offerto da Buck è già stato realizzato anche attraverso altre applicazioni Web. Wiki e CMS implementano infatti servizi simili a quelli offerti dal la mia applicazione.

I wiki però, a mio modesto parere, sono software limitati. Le loro applicazioni non sono varie e complete come quelle che troviamo in un CMS. All'intero di wiki è sicuramente possibile creare contenuti e meta-dati riguardanti, ma poco altro. Se l'obbiettivo della comunità scientifica vuole essere quello di creare meta-datazione riguardante ogni articolo pubblicato sul Web, i wiki non sono forse lo strumento migliore. E' infatti plausibile che l'utente che pubblica tali articoli sia interessato ad allegare agli stessi materiale multimediale, sondaggi, opinioni. Il processo per la meta-datazione del Web non può e non deve passare attraverso una sua limitazione di creazione di contenuti.

I CMS invece, già presentati come applicazioni modulari basate su vaste librerie interne, proprio per come sono strutturati permettono una continua espansione delle loro funzionalità. Se contemporaneamente tali applicazioni renderanno sempre più semplice agli utenti la creazione di contenuto online sempre più sofisticato e svilupperanno sistemi per la creazione di meta-informazioni migliori perché “user friendly” e talvolta automatici, allora la loro crescita di

utilizzo da parte degli utenti potrà continuare a crescere, insieme alla creazione di meta-dati riguardanti un articolo.
Tra tutti i CMS che potevano essere scelti per la realizzazione di questo progetto, la decisione di adottare Drupal si è sicuramente dimostrata la scelta giusta. Il sistema in questione è caratterizzato infatti da due componenti molto importanti, quantomeno in parte mancanti negli altri CMS Open Source [Joo], [DRU10].
Il primo di questi è un modulo che permetta la gestione della tassonomia. Questo modulo, utilizzato all' interno di Buck, ha permesso di dare un significato ontologico ai valori utilizzati per la creazione di meta-dati. Ogni termine utilizzato è infatti in relazione con tutti i termini che avrebbero determinato una definizione diversa del meta-dato.
La seconda motivazione deriva dal lavoro in atto sul software Drupal per potere integrare i contenuti pubblicati su di esso con il progetto del Semantic-Web. Il Semantic-Web è un progetto che vuole dare accessibilità alle macchine di tutto il contenuto web che esse stesse memorizzano. Accessibilità intesa come la possibilità di comprendere il contenuto del testo, poterlo ri-elaborare, potere creare nuove informazioni e collegamenti a partire da questo testo.
Questo porta ovviamente ad interessanti sviluppi futuri che l'applicazione Buck potrebbe subire in futuro. Sarebbe sicuramente interessante potere inserire nel funzionamento del sistema la tecnologia del Semantic-Web per la conoscenza e diffusione di tutti i meta-dati creati dall'applicazione Buck. Questo potrebbe portare un notevole incremento delle informazioni che il portale offre al mondo del WWW.

Bibliografia

[CFS] Katherine Curtis, Paul W. Foster, Fred Stentiford ,
“Metadata - The Key to Content Management Services"
http://www.ee.ucl.ac.uk/~fstentif/curtis.htm#fig2
visitato nel giugno 2010

[CMD] Kris Cardinaels, Michael Meire, Erik Duval ,
”Automating Metadata Generation: the Simple Indexing Interface ”,
http://citeseerx.ist.psu.edu

[Wik10] Wikipedia. “Wikipedia”
http://en.wikipedia.org/wiki/Wikipedia,
visitato giugno 2010.

[Drup10] Drupal ,”Drupal”,
http://drupal.org
visitato giugno 2010

[COP] Flavio Copes, “Guida Drupal”,HTML.IT
http://cms.html.it/guide/leggi/146/guida-drupal/,
visitato maggio 2010.

[DOC] Cory Doctorow, “Metacrap: Putting the torch to seven straw-men of the meta-utopia”, 2001
http://www.well.com/~doctorow/metacrap.htm

[WordP] WordPress, “WordPress”,
http://www.wordpress-it.it/wiki/Main/WordPress
visitato giugno 2010.

[Joo] Joomla, “Joomla”,

http://www.joomla.org/
visitato maggio 2010.

[NIC] Dennis Nicholson ,
”Ensuring Metadata Interoperability Across Scottish Content Management Systems and Digital Repositories: Guidelines for Best Practice ”,
http://cms.cdlr.strath.ac.uk,
2005-06

[IMPV]Angelo Di Iorio, Alberto Musetti, Silvio Peroni, Fabio Vitali,
“Crowdsourcing semantic content: a model and two applications ”,
http://palindrom.es/phd/wp-content/uploads/2009/06/oWiki_ASAAI20_CR.pdf,
2009

[PHPDOM] PHP,
http://php.net/manual/en/book.dom.php,
visitato aprile 2010.

[LIV] Giancarlo Livraghi, 2009, http://www.gandalf.it/data/data1.htm

[VIT] Fabio Vitali, Corso di Tecnologie e Web ,Università di Bologna,
“Semantic Web: teoria”, 28/4/2009

[OCMS] OpensourceCMS, http://php.opensourcecms.com/, visitato maggio 2010

[GSR] Claudio Gennaro , Pasquale Savino , Fausto Rabitti, LNCS, “Milos: A Multimedia Content Management System for Digital Library Applications”, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.63.3010, 2004

[BIR] Francesco Re, Silvana Giornano, Mariotti Publishing, “birra, oltre la schiuma c'è di più”, isbn:978-88-8226-318-8, 2008

[WLO] Paul Wlodarczyk, “Automating CMS metadata – could that work? How? “,
http://paulwlodarczyk.wordpress.com/2008/08/19/automating-cms-metadata-could-that-work-how/, 19/08/2008