Progettazione e realizzazione di un’ architettura per applicazioni Web basata sul modello MMASS: un esempio applicativo per il monitoraggio di siti e visitatori.

Relatore: Prof.ssa Stefania Bandini
Controrelatore: Prof. Roberto Polillo
Correlatori: Dott. Giuseppe Vizzari
Dott.ssa Sara Manzoni

Tesi di Laurea di:
Andrea Bonomi
Matricola: 053958

Anno Accademico 2003-2004
ai miei genitori

a Gemma
Ringraziamenti

Ringrazio la Prof.ssa Bandini che, accettandomi come suo tesista, mi ha dato la possibilità di conoscere e frequentare il LabIC e il Prof. Polillo che ha accettato di essere il controrelatore per questo mio lavoro di tesi.

Ringrazio infine Giuseppe, Sara, Paolo, Fabio, Ettore e tutti i ragazzi del LabIC per i preziosi consigli e i momenti passati insieme.
# Indice

1 **Introduzione** 4
   1.1 Motivazioni 4
   1.2 Obiettivi 4
   1.3 Organizzazione della tesi 5

2 **Agenti e Web** 6
   2.1 Aree applicative 6
      2.1.1 Information Retrieval e Filtering 7
         2.1.1.1 WebCrawler 8
         2.1.1.2 Google 11
      2.1.2 Advising e Focusing 14
         2.1.2.1 Letizia 14
         2.1.2.2 Segue 16
         2.1.2.3 Footprints 17
      2.1.3 Supporto alla interazione Web 19
         2.1.3.1 Assistenti virtuali 19
         2.1.3.2 Ambienti virtuali 21
      2.1.4 Monitoraggio di siti Web 23
         2.1.4.1 Netcraft 23
         2.1.4.2 eTrust Web Access Control 24
   2.2 Temi di ricerca 25
      2.2.1 Web Agent Support Program 27
      2.2.2 Mondi virtuali 3D come società di agenti 29
INDICE

3  Web come sistema ad Agenti 31
   3.1  MMASS ........................................ 31
       3.1.1  Spazio .................................. 32
       3.1.2  Campi .................................. 33
       3.1.3  Agenti .................................. 33
   3.2  Adattamento del modello MMASS .......... 35
   3.3  Modellazione del Web tramite MMASS .. 36

4  Ambiente software per il modello MMASS 39
   4.1  Piattaforma .................................. 39
   4.2  Spazi ........................................ 41
       4.2.1  Posti .................................... 42
       4.2.2  Spazi regolari ......................... 44
           4.2.2.1  VoidSpace ......................... 44
           4.2.2.2  GridSpace ......................... 44
           4.2.2.3  VonNeumannGridSpace .......... 45
           4.2.2.4  VonNeumannTorusSpace ........ 45
           4.2.2.5  MooreGridSpace .................. 46
           4.2.2.6  MooreTorusSpace ................. 46
           4.2.2.7  HexagonalGridSpace .......... 47
           4.2.2.8  Visualizzazione .................. 47
       4.2.3  Spazi generici ......................... 50
           4.2.3.1  GraphSpace ....................... 50
           4.2.3.2  PositionedGraphSpace .......... 50
           4.2.3.3  XmlSpace ......................... 51
           4.2.3.4  Visualizzazione .................. 52
           4.2.3.5  Creazione e modifica degli spazi 54
   4.3  Agenti ..................................... 55
   4.4  Campi ...................................... 58
       4.4.1  Diffusione .............................. 60
           4.4.1.1  Diffusione nulla ............... 61
           4.4.1.2  Flooding corretto ............... 61
INDICE

4.4.1.3 Diffusione per spazi regolari ...................... 62
4.5 Strumenti ........................................... 63
  4.5.1 Object Inspector .................................. 64
  4.5.2 Console ........................................... 65
  4.5.3 Memory Monitor .................................... 66
4.6 Esempio di utilizzo ................................... 66

5 WebSpace ............................................. 69
  5.1 Descrizione dell’ applicazione ......................... 69
  5.2 Creazione della struttura spaziale ....................... 70
    5.2.1 Creazione da struttura nota ......................... 71
    5.2.2 Acquisizione della struttura di un sito Web ........ 73
    5.2.3 Generazione dinamica dello spazio secondo la navigazione degli utenti ................................. 75
  5.3 Raccolta di informazioni: le tracce degli utenti .......... 77
    5.3.1 Proxy ........................................... 79
    5.3.2 Filtro Web Server ................................ 80
    5.3.3 Tracker ........................................... 81
  5.4 Generazione dei contenuti adattivi ....................... 84
    5.4.1 Contenuti adattivi di WebSpace ...................... 86
  5.5 Funzionalità di monitoraggio ........................... 87
  5.6 Visualizzazione della struttura spaziale ................. 89
    5.6.1 Mappe disegnate a mano ........................... 90
    5.6.2 Mappe a grafo .................................... 91
    5.6.3 Mappe fish-eye e iperboliche ....................... 94
    5.6.4 Mappe Land-Use .................................. 96
    5.6.5 Mappe WebSpace .................................. 98
  5.7 Sperimentazione ...................................... 100

6 Sviluppi futuri ........................................ 102
Capitolo 1

Introduzione

1.1 Motivazioni

La tecnologia degli agenti viene largamente utilizzata come strumento per l’ analisi, la progettazione e l’ implementazione di sistemi complessi. In particolare, i sistemi multiagente costituiscono un valido approccio per la modellazione e la simulazione, consentendo di rappresentare direttamente le entità coinvolte, il loro comportamento e interazioni.

Anche per quanto riguarda il mondo del Web, esistono svariate applicazioni ad agenti, che spaziano dalla ricerca alla comunicazione e all’ awarness. A volte si tratta di sistemi a singolo agente, in cui l’ idea di agente è legata al fatto che si tratta di programmi che lavorano con un certo grado di indipendenza mentre altre volte si tratta di altre applicazioni caratterizzate da più agenti che comunicano tra di loro. Mancano applicazioni che modellano il Web come un sistema ad agenti situati, in cui l’ insieme delle pagine costituisce lo spazio in cui gli agenti si muovono.

1.2 Obiettivi

Lo scopo di questa tesi è la progettazione e la realizzazione di una piattaforma che consenta di sviluppare applicazioni basate sul modello MMASS (Multilayered Multi Agent Situated System), la modellazione del Web come un sistema ad agenti e la realizzazione di un’ applicazione basata sul modello definito.
La finalità della piattaforma è rendere più semplice lo sviluppo di applicazioni basate sul modello MMASS, fornendo agli sviluppatori gli elementi di base per realizzarle.

La modellazione del Web come un sistema ad agenti è la parte centrale della tesi, in cui viene descritto come modellare, in termini di MMASS, la struttura delle pagine che compongono un sito Web come uno spazio e gli utenti che navigano il sito come degli agenti situati nello spazio.

La realizzazione di un’ applicazione, basata sul modello descritto e sulla piattaforma, che fornisca sia dei contenuti adattivi agli utenti del sito che delle informazioni per il Web Master permette di dimostrare la validità del modello stesso e la praticità dell’ utilizzo della piattaforma.

### 1.3 Organizzazione della tesi

Nel capitolo 2 viene descritto l’ utilizzo degli agenti nell’ area Web, vengono mostrate sia diverse applicazioni che si basano su questo approccio che aree di ricerca ancora aperte.

La modellazione del Web come un sistema ad agenti basato sul modello MMASS (Multilayered MultiAgent Situated System) viene presentata nel capitolo 3. Nello stesso capitolo è contenuta anche una descrizione del modello MMASS stesso.

Il capitolo 4 contiene la descrizione della piattaforma che permette di realizzare applicazioni basate sul modello MMASS e che è stata realizzata durante questo lavoro di tesi. Viene descritto il lavoro svolto partendo dalla piattaforma realizzata da Andrea Rizzi per arrivare alla realizzazione della versione attuale e viene anche illustrata un’ applicazione di esempio basata sul modello MMASS ed implementata tramite la piattaforma.

Nel capitolo 5 viene mostrata WebSpace, un’ applicazione che sfrutta la modellazione del Web come un sistema ad agenti descritta nel capitolo 4 e la piattaforma MMASS descritta nel capitolo 3, per fornire contenuti adattivi agli utilizzatori di un sito Web e fornire supporto ai Web Master.

Le conclusioni e gli sviluppi futuri sono presentati nel capitolo 6.
Capitolo 2

Agenti e Web

In questo capitolo verrà descritto lo stato dell’arte degli agenti dell’area del Web e verranno analizzate diverse applicazioni ad agenti che operano in questo ambito.

2.1 Aree applicative

Prima capire quali sono le applicazioni dei sistemi ad agenti nell’area Web bisogna fare un passo indietro e cercare di capire cosa si intende per agente in questo settore. Per fare questo si può partire da un articolo di Wooldridge, "Intelligent Agents" [Wooldridge 1995], in cui l’autore cerca di fare luce sui principi fondanti alla base del concetto di agente.

L’articolo si apre con il tentativo di definire cosa sia un agente; mentre molti aspetti risultano avere una importanza più o meno evidente al variare del dominio in cui un agente agisce (la capacità di apprendere, ad esempio, a volte risulta essenziale, a volte no), ma è il concetto di autonomia che risulta essere centrale nella definizione di agente. Successivamente Wooldridge affronta il problema di definire un agente intelligente, inteso come agente dotato di flessibilità, dove flessibilità indica tre proprietà fondamentali: reattività, capacità di percepire i cambiamenti dell’ambiente e regolarsi di conseguenza per ottenere il proprio goal; presenza attiva nell’ambiente, capacità di mostrare un comportamento guidato dagli obiettivi prendendo iniziative per ottenere il proprio goal; capacità di lavorare in società, possibilità di interagire con altri agenti per ottenere il proprio goal.
CAPITOLO 2. AGENTI E WEB

<table>
<thead>
<tr>
<th>Tipologia agente</th>
<th>Tipologia agente</th>
<th>Singolo/Multi</th>
<th>Locale/Server</th>
<th>Complessità</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebCrawler</td>
<td>informativo</td>
<td>multi</td>
<td>server</td>
<td>bassa</td>
</tr>
<tr>
<td>Google</td>
<td>informativo</td>
<td>multi</td>
<td>server</td>
<td>bassa</td>
</tr>
<tr>
<td>Letizia</td>
<td>di interfaccia</td>
<td>singolo</td>
<td>locale</td>
<td>media</td>
</tr>
<tr>
<td>Segué</td>
<td>di interfaccia</td>
<td>singolo</td>
<td>locale</td>
<td>media</td>
</tr>
<tr>
<td>Footprints System</td>
<td>di interfaccia</td>
<td>singolo</td>
<td>locale</td>
<td>media</td>
</tr>
<tr>
<td>eTrust Web Access Control</td>
<td>monitoraggio</td>
<td>multi</td>
<td>server</td>
<td>bassa</td>
</tr>
</tbody>
</table>

Nelle applicazioni reali dei sistemi ad agenti, si trovano spesso privilegiate alcune di queste caratteristiche, mente altre sono marginali o del tutto assenti. Nelle applicazioni legate al mondo del Web troviamo differenti tipologie di agenti, ad esempio agenti informativi, di interfaccia, autonomi o cooperativi, che lavorano lato server o client, ma la caratteristica che li accomuna è il loro agire in modo autonomo, pur con gradi differenti di autonomia. La seguente tabella illusta le principali caratteristiche degli agenti delle applicazioni prese in esame:

Sempre in un ottica funzionale si è scelto di fare una panoramica di applicazioni ad agenti in ambito Web suddividendole secondo lo scopo che si prefiggono, piuttosto che per la tipologia di agenti impiegati. Le aree applicative prese in esame sono le seguenti:

- Information Retrieval e Filtering
- Advising e Focusing
- Supporto alla interazione Web
- Monitoraggio di siti Web

2.1.1 Information Retrieval e Filtering

C’è una frase di Tim Berners Lee (l’inventore del Web) che aiuta a inquadrare chiaramente il fenomeno dell’aumento rapido e caotico dei contenuti presenti in Internet: “L’ondata di entusiasmo per il sistema di ippertesto globale induce a riversare risorse sempre più ingenti dello sforzo di fornirgli più contenuto e più possibilità di migliorare l’organizzazione e la catalogazione. E qui si è di fronte ad un circolo vizioso, perché quanto più aumentano i contenuti interessanti disponibili nel web tanto più i lettori sono incentivati a collegarsi: analogamente, quanto più aumentano le persone che ’scorrono’ il web, tanto più aumenta l’incentivo a riversarvi contenuti che interessino gente.” [Lee 1997]
I principali strumenti per non perdersi in questo mare di dati sono le Directory, cioè elenchi di siti suddivisi per categorie, e i Motori di ricerca che permettono di ricercare le pagine che contengono un dato termine. Mentre le Directory sono concettualmente semplici (trattandosi di elenchi redatti a mano) e gli utenti le consultano per trovare i contenuti che li interessano, i motori di ricerca costituiscono un’interessante area di ricerca e di sperimentazione per nuove tecnologie legate agli agenti e al Information Retrieval.

I motori di ricerca che verranno qui brevemente descritti sono WebCrawler, il primo motori di ricerca basato su agenti e Google, che attualmente è quello maggiormente utilizzato e che permette di far ricerche sul maggior numero di pagine.

2.1.1.1 WebCrawler


Figura 2.1: Interfaccia di ricerca semplice di WebCrawler
Per effettuare velocemente le ricerche in milioni di documenti presenti sulla rete, WebCrawler deve prima recuperarli, indicizzarli e memorizzarli su un database. Successivamente, quando un utente farà una ricerca, questa verrà fatta sul database stesso.

I principali componenti del sistema sono i seguenti:

**Query Server** È il Web Server e si occupa di gestire l’ interfaccia attraverso cui gli utenti fare le ricerche (query) e genera le pagine contenenti i risultati. L’interfaccia è disponibile in due differenti formati, semplice e avanzata; la versione semplice presenta solo un campo di testo in cui inserire la o le parole da ricercare, mentre nella versione avanzata sono presenti una serie di campi che permettono di specificare altri parametri di ricerca. Per ogni risultato di una ricerca viene calcolato un valore di ‘Rank’ che determina la corrispondenza tra la query ed il risultato. Maggiore è questo valore (che è espresso in percentuale), maggiore dovrebbe essere l’ attinenza tra il documento e la ricerca effettuata. I risultati vengono ordinati secondo il grado di rilevanza e suddivisi in pagine.

**Database** È responsabile della memorizzazione delle pagine e degli indici. Maggiore è il numero di pagine che vengono acquisite, maggiori sono le possibilità di trovare dei contentuti che soddisfano una data ricerca, anche se aumenta anche il numero di documenti che non rilevanti che compariranno tra i risultati delle ricerche. Tenere aggiornato il database delle pagine è di fondamentale importanza, perché il contenuto delle pagine evolve nel tempo, vecchie pagine vengono cancellate e ne vengono create di nuove.

**Search Administrator** I suoi due compiti principali sono l’acquisizione delle pagine e la loro indicizzazione. Per trovare la pagina da recuperare viene utilizzato un elenco iniziale che viene esteso aggiungendovi gli indirizzi di altre pagine che vengono presi dai link presenti nelle pagine già acquisite. Il recupero vero e proprio delle pagine viene eseguito da agenti che sono coordinati dal Search Administrator. Dopo che una pagina è stata recuperata, viene eseguita un analisi sintattica e vengono generati degli indici da utilizzare successivamente in fase di ricerca ricerca. Nella fase di analisi viene deciso quali parole saranno inserite negli indici; in genere verranno escluse le ’stop-words’, cioè quelle parole come gli articoli o le congiunzioni che non sono utili ai fini della ricerca.

**Agents** Gli agenti, chiamati Crawlers, svolgono il compito di recuperare le pagine su cui successivamente verranno effettuate le ricerche. Operano iterativamente, scaricando una pagina Web, processandola e seguendo i link per trovare altre
pagine da scaricare e il risultato di questa operazione è una collezione di documenti (pagine HTML e documenti di testo) da memorizzare sul database. È stato scelto di utilizzare degli agenti perché c’era la necessita di acquisire più pagine in parallelo (per minimizzare i tempi sprecati nelle attese dei dati dalla rete) da macchine differenti e si desiderava un sistema facilmente scalabile. Gli agenti operano indipendentemente e comunica con il Search Administrator solo per l’assegnazione dei compiti e per riceverne i risultati.

Figura 2.2: Architettura di WebCrawler

Per determinare un ordinamento \( (\text{Ranking}) \) tra i risultati di una ricerca, viene calcolata una funzione, chiamata relevance per ogni documento che soddisfa la query. Questa funzione è così definita:

\[
R(doc, query) = \min \left( \sum_{i=1}^{w} tf(i) \cdot \left(1 - \frac{\log(res(query))}{\log(doc)}\right), mr \right)
\]

- \( tf(i) \) è la frequenza di una parola in un dato documento, calcolata come il numero di occorrenze della parola diviso il numero totale di parole presenti nel documento stesso;
- \( res(query) \) è il numero di risultati che soddisfano la query;
- \( doc \) è il numero totale di documenti;
- \( mr \) è un parametro che indica il massimo valore di rank ammesso per un documento.
2.1.1.2 Google

Google è nato come prototipo di motore di ricerca in grande scala che facesse largo uso delle strutture presenti negli ippertesti. Google è stato progettato per acquisire ed indicizzare efficientemente il web e produrre risultati più soddisfacenti di quelli forniti dai motori di ricerca esistenti [Brin Page 1998].

Realizzare un motore di ricerca di grandi dimensioni in grado di indicizzare miliardi di pagine e di supportare milioni di ricerche al giorno non è un compito facile; nonostante il costo del hardware continui a scendere e le sue performance ad aumentare, anche il Web ed il numero di suoi utenti cresce rapidamente.

Nel design di Google si è voluto tener conto di entrambi i fattori, per permettergli di crescere insieme al web. Ad esempio, per avere buone performance, Google è stato implementato in C e C++ ed è stato pensato per funzionare sia su Solaris che su Linux.

L’acquisizione delle pagine è fatta da un insieme di agenti distribuiti chiamati Crawler. L’URL Server invia la lista di URL da recuperare ai Crawler, che poi agiscono autonomamente. Una volta che le pagine sono state acquisite vengono inviate allo Store Server, che si occupa di comprimere le pagine e di memorizzarle in un Repository.

Come algoritmo di compressione è stato scelto zlib\(^1\) che è stato considerato un buon compromesso tra velocità e grado di compressione; la compressione che si ottiene sul Repository è di circa 1 a 3, mentre con bzip\(^2\) era di 1 a 4, ma zlib risulta significativamente più veloce.

![Figura 2.3: Struttura del Repository di Google](image)

L’Indexer si occupa dell’indicizzazione dei documenti acquisiti. Opera decomprimendo i documenti, facendo il parsing e convertendoli in degli insiemi di parole chiamati Hit. Ogni Hit contiene la parola, la sua posizione all’interno del documento, la dimensione (relativa al carattere utilizzato nel documento) e se è maiuscola o minuscola. Le Hit vengono distribuite in un gruppo di Barrel e vengono ordinate da un insieme di Sorter.

---

\(^2\) http://sources.redhat.com/bzip2/
Un'altra funzione degli Indexer è di estrarre i link dai documenti che vengono passati al URL Resolver. L'URL Resolver trasforma gli URL relativi presenti nei documenti in URL assoluti che poi verranno utilizzati per acquisire nuove pagine.

Oltre ad un'architettura adatta ad essere scalata, Google ha introdotto novità anche per quanto riguarda il ranking delle pagine, cioè nel modo in cui viene assegnato un punteggio ad ogni pagina per ordinare i risultati.

Esse sono PageRank e l'Hypertext-Matching Analysis.

PageRank è un algoritmo brevettato che si basa sulla considerazione che la 'qualità' di una pagina dipenda dal numero di link che la raggiungono e dalla 'qualità' delle pagine da cui partono questi link. Prima di Google, nessun altro motore di ricerca partiva da questi presupposti. Contare il numero di link permette di avere una stima approssimativa dell'importanza di una pagina: più link arrivano, maggiormente saranno interessante. PageRank estende questa idea, pesando diversamente i link secondo la pagina da cui partono e normalizzando secondo il numero di link presenti della pagina.

La funzione PageRank, che è una distribuzione di probabilità, è così definita:
\[ PR(A) = (1 - d) + d \sum \frac{PR(T_i)}{C(T_i)} \]

- \( A \) è la pagina di cui si vuole calcolare PageRank;
- \( d \) un parametro (Damping Factor) impostato a un valore compreso tra 0 e 1 (generalmente è 0.85);
- \( T_1 \ldots T_n \) sono pagine contenenti dei link verso \( A \);
- \( C(A) \) è uguale al numero di link che partono da \( A \).

Il valore di PageRank per ogni pagina indicizzata viene calcolato attraverso un algoritmo iterativo (che fornisce un valore approssimato).

L’Hypertext-Matching Analysis riguarda l’analisi del contenuto delle pagine, cioè Google, al posto di limitarsi a ricercare le parole trattando le pagine Web come dei documenti di testo, è sensibile anche al contenuto dei tag che specificano la formattazione della pagina. Ad esempio una parola scritta con un carattere più grande (relativamente alla dimensione del resto del testo del documento) sarà considerato più importante di una scritta più in piccolo.

2.1.2 Advising e Focusing

Come per le applicazioni descritte nel paragrafo precedente, anche lo scopo dei sistemi qui presentati è quello di ricercare informazioni, solo che queste applicazioni sono dei Personal Assistent e servono ad aiutare l’utente nel suo lavoro con il Browser.

Questi programmi osservano e analizzano l’attività degli utente per poterli poi aiutare a trovare ciò di cui hanno bisogno.

2.1.2.1 Letizia


Letizia osserva la navigazione di un utente e cerca di aiutarlo, fornendogli suggerimenti sulle pagine a cui potrebbe essere interessato.

Letizia può essere considerata un agente di interfaccia autonomo[Lieberman 1997]; due dei principi degli agenti di interfaccia autonomi sono la manipolazione indiretta e la delegazione di compiti da parte degli utenti.

Nella ‘manipolazione diretta’ l’utente deve fare da solo tutti i passaggi necessari a raggiungere il suo obiettivo. Durante una navigazione in cui cerca dei documenti, un utente deve leggere ogni pagina e decidere se ne è soddisfatto e quali link andare a vedere. Un agente di interfaccia permette invece di passare a una ‘manipolazione indiretta’, cioè può eseguire autonomamente delle operazioni e controllare i risultati. Questo permette all’utente di delegare all’agente un certo numero di compiti.

Figura 2.6: Struttura di un’applicazione con un agente di interfaccia autonomo

L’interfaccia di Letizia consta di diverse finestre; una è la tradizionale interfaccia del browser, attraverso cui l’utente può navigare come è abituato a fare. Letizia osserverà il suo comportamento e utilizzerà i dati raccolti per decidere quali sono i suoi interessi. Mentre l’utente naviga una pagina, Letizia esplora tutti i suoi link e mette in evidenza, in un’altra finestra, quei link che portano a contenuti che potrebbero risultare maggiormente interessanti per l’utente.

Per determinare quali sono gli interessi dell’utente, Letizia si basa sull’analisi del contenuto della pagine viste dall’utente e dei ‘Preferiti’ (Bookmarks). Viene tenuta traccia di ogni visita per determinare quali sono le pagine viste con maggior frequenza e che quindi suscitano maggior interesse.

Anche il numero di link che partono da una pagina viene utilizzato per determinarne l’importanza: maggiore è il numero di link ad argomenti di interesse presenti in una pagina, maggiore è l’interesse dell’utente per la pagina.
2.1.2.2 Segué

Il progetto Segué parte dalla constatazione che generalmente gli utenti utilizzano spesso i Bookmarks e mentre poco la 'Cronologia' (Browser History), che, se organizzata bene, potrebbe risultare utile.

L’idea è che la Cronologia sia poco utilizzabile, perché non organizzata secondo dei criteri che ne facilitano l’utilizzo. I massimi raggruppamenti forniti dai browser sono la suddivisione per giornata o per sito web.

Segue si propone come un ‘Browsing History Agent’, che si occupa di dividere la Browser History in segmenti (gruppi di pagine inerenti che riflettono un dato interesse) e fornisce una interfaccia grafica intuitiva per accedere ai segmenti.

Analizzando le pagine consultate dall’utente, Segué decide autonomamente quando creare nuovi segmenti. La creazione di nuovi segmenti riflette i cambi di interessi dell’utente. I cambi di interesse vengono individuati comparando la pagina attuale con le pagine visitate precedentemente e ricercando delle similitudini [Shearin 2000].

4http://web.media.mit.edu/~sibyl/projects/segue/index2.html
5Ulteriori informazioni sulla analisi della navigazione al fine di determinare gli interessi di un utente si trovano in [Lieberman 1999].
2.1.2.3 Footprints

Footprints crea la nozione di ‘oggetti ricchi di storia’, cioè di oggetti che tengono traccia del loro uso, come l’angolo piegato ad una pagina di un libro indica che la pagina è stata selezionata dal lettore [Henry Lieberman 1998].

L’idea da cui parte il lavoro di ricerca di Alan Wexelblat è che, a differenza del mondo reale che appare ‘vissuto’, i documenti digitali appaiono sempre sterili. Ad esempio, quando aprimo un documento o visitiamo una pagina web, abbiamo la sensazione che siamo le uniche persone che hanno a che fare con quei dati.

Nel mondo digitale, le compagine si sommergono con montagne di dati grezzi, come le statistiche dei web server o gli storici degli acquisti effettuati. Molti di questi dati non verranno mai esaminati. Essi mancano di relazioni per formare una storia utile e al massimo si possono fare correlazioni statistiche tra di loro [Wexelblat 1999]. Dato che nel mondo reale le informazioni relative alla storia delle cose, che spesso risultano utili, sono intrisecamente legate agli oggetti a cui si riferiscono (come dei gradini consumati o dei libri ‘vissuti’) si vuole provare ad arricchire anche il mondo digitale con queste informazioni.
Footprints nasce come prototipo di un sistema per la visualizzazione di spazi di informazioni ricchi di storia. È un agente di interfaccia che tiene traccia dell’uso che viene fatto di un sito, proponendo all’utente delle diverse visualizzazioni, come il percorso che ha compiuto visitando il sito. Questa visualizzazione mostra come un utente percepisce l’organizzazione dello spazio, piuttosto che la struttura imposta da chi ha creato i contenuti, cioè ne mostra l’uso che ne viene fatto, come un percorso tracciato da molta gente che ha camminato in un posto.

Così come una casa cambia aspetto secondo le persone che vi abitano, la visualizzazione del sito non sarà statica, ma evolverà nel tempo, secondo l’uso che ne verrà fatto. L’utente dovrebbe riconoscere così come ‘familiari’ i documenti che visualizza di frequente e poter aver traccia dei percorsi che ha compiuto.

La presentazione di queste informazioni all’utente avviene attraverso la visualizzazione, utilizzando la geometria iperbolica secondo lo stile di [Lamping et al 1995] di un grafo che mostra i percorsi che ha effettuato ed i documenti che ha visualizzato.

L’utente può ineragire con la mappa, trascinandola in ogni direzione per visualizzare i percorsi che lo interessano e può cliccare su ogni nodo (che rappresenta un documento) per mostrare il suo titolo o doppio-cliccare per visualizzare il suo contenuto in una finestra del browser.

L’uso del colore permette di evidenziare la popolarità di un documento: i più popolari sono rappresentati da un’icona rossa mentre gli altri assumono varie gradazioni di rosa fino a quelli meno visitati che appaiono bianchi.

L’ultimo documento aperto è evidenziato grazie alla sua icona colorata di nero. Oltre alla visualizzazione iperbolica, è presente anche una rappresentazione ad albero del sito e anche in questa visualizzazione sono usati gli stessi codici cromatici per identificare i gradi di utilizzo.
2.1.3 Supporto alla interazione Web

Uno degli ambiti per cui vengono utilizzati gli agenti sul web è quello di essere utilizzati per interagire con gli utenti. L’idea è quella che un utente potrebbe sentirsi solo nel mare del web, per cui si cerca di riproporre una delle situazioni tipiche del mondo reale, cioè l’interazione con gli altri. L’interazione può avvenire tra utenti (magari mediata da agenti) o tra utenti ed agenti. Gli esempi delle funzioni svolte dagli agenti in questo campo sono numerose: dal fornire informazioni all’essere avversari nei giochi multigiocatore. Più in generale, spesso gli agenti hanno la funzione di sostituire delle persone per determinati ruoli senza che questo sia evidente.

In rappresentanza di questa categoria di agenti verranno presi in considerazione gli assistenti virtuali, che forniscono supporto in linguaggio naturale agli utenti ed gli ambienti virtuali, che sono popolati, oltre che da utenti, da agenti.

2.1.3.1 Assistenti virtuali

Gli assistenti virtuali sono una tecnologia che permette all’utente di ottenere informazioni nel modo più naturale possibile, come se dialogasse con una persona. L’assistente è generalmente rappresentato da un avatar a cui possono venire poste delle domande.
CAPITOLO 2. AGENTI E WEB

in linguaggio naturale e che fornisce delle risposte come se stesse parlando, magari muovendo la bocca o cambiando espressione. L’impressione per l’utente è quella di trovarsi di fronte a qualcuno disposto ad aiutarlo nella ricerca di ciò di cui ha bisogno.

Nella loro attuale implementazione, questi sistemi hanno anche dei problemi, ad esempio, essi si focalizzano sull’efficienza della generazione dei dialoghi piuttosto che sulla possibilità di integrazione con i gli altri sistemi aziendali. Si tratta di soluzioni standalone che non possono essere integrate o che richiedono che il sistema aziendale sia adattato per poterli integrare [Gurki 2001].

Il progetto ADVICE (Virtual Sales Assistant for the Complete Customer Service Process) si prefigge di sviluppare un’architettura per per assistenti virtuali (ed in particolare per commessi virtuali) che superi le limitazioni dei sistemi esistenti. I commessi virtuali sono una tipologia di assistenti virtuali che richiede una stretta integrazione con il sistema informativo aziendale, in quanto devo accedere a molte informazioni riguardanti i prodotti. Nelle implementazioni attuali, in genere, i sistemi che implementano commessi virtuali accedono a basi di dati create appositamente, con tutti i problemi che esso comporta: necessità di aggiornamenti su più sistemi differenti, informazioni non aggiornate e non complete.

ADVICE è un sistema multi-agente che permette agli utenti di dialogare in linguaggio naturale con degli assistenti virtuali. Il funzionamento del sistema può essere così schematizzato:

1. L’utente formula una domanda in linguaggio naturale.
2. Il testo è inviato al Interface Agent per l’analisi.
3. L’Interface Agent un modulo di analisi del linguaggio naturale per generare gli speech acts.
5. L’Interface Agent genera la risposta in linguaggio naturale a partire dal speech acts.

http://www.advice.iao.fhg.de
[Smith Barry 1990]
6. La risposta viene presentata all’utente da un avatar tridimensionale.

![Diagramma dell’architettura di un agente del sistema ADVICE](image)

**Figura 2.11: Architettura di un agente del sistema ADVICE**

### 2.1.3.2 Ambienti virtuali

Gli ambienti virtuali sono luoghi di ritrovo su Internet che modellati come fossero degli ambienti. Possono rappresentare ambienti reali (esistenti o esistiti), realistici o di fantasia. Esempi di ambientazioni reali possono essere una città del passato (esistita) o un museo (esistente), mentre una sala riunioni virtuale può essere un luogo realistico ma non reale, cioè non rappresentare nulla di realmente esistente ma rappresentare qualcosa che potrebbe esistere e che si ispira a qualcosa di esistente. Esistono anche ambientazioni immaginari, come possono essere quelle fantasy o fantascientifiche.

Il termine ‘virtuale’ può essere forvante, perché, qualunque tipologia di ambiente rappresentino, si tratta comunque di luoghi di ritrovo ‘reali’ in cui cioè delle persone si ritrovano. Non si tratta quindi di cose meno reali di quanto non lo sia una telefonata.

I primi ambienti virtuali erano unicamente testuali e si differenziavano in due grandi categoria: MUD (Multi User Dungeon) dei veri e propri giochi di ruolo e MOO (MUD Object Oriented) più orientati alla comunicazione e con oggetti dotati di comportamenti.

In molti di questi sistemi è stata utilizzata la tecnologia ad agenti: ad esempio, in molti MUD i personaggi non giocanti (ad esempio i mostri) sono degli agenti con una loro conoscenza del mondo, dei desideri (ad esempio: difendere il tesoro e uccidere i giocatori), dei sensori attraverso cui percepiscono l’ambiente e gli altri e degli attuatori che gli permettono di modificare il mondo (e di uccidere i giocatori).
CAPITOLO 2. AGENTI E WEB

Figura 2.12: Interfaccia di un MUD testuale

Una delle evoluzioni di questi sistemi è stata l’ integrazione con in web, che permette di rappresentare informazioni non solo testuali e semplifica l’ interazione degli utenti tra di loro e con l’ambiente.

Differenti tipologie di interfacce possono anche coesistere, ad esempio Lingua MOO⁹, un MOO a scopo educativo creato nel 1995 presso l’ University Of Texas ha sia un’interfaccia ‘classica’ testuale che una via web. In questo MOO è disponibile una serie di componenti che permettono di fare e-learning in un ambiente multiutente, permettendo agli insegnanti di creare stanze virtuali in cui tenere le lezioni.

⁹http://lingua.utdallas.edu
CAPITOLO 2. AGENTI E WEB

2.1.4 Monitoraggio di siti Web

Il monitoraggio delle reti e, nel caso preso in esame, dei siti Web, è un’ attività spesso demandata a degli agenti. In questo caso, il termine ‘agente’ si riferisce a dei compontenti che abbiano un dato grado di autonomia, cioè che operino senza la necessità di supervisione. Un’ altra caratteristica di questi agenti è la loro locazione fisica, visto che spesso gli agenti di controllo sono agenti sparsi su varie macchine di una rete e che comunicano tra di loro o con un server centralizzato per comunicare problemi o effettuare delle rilevazioni. Possono essere utilizzati per scopi differenti, come verificare a dati intervalli se un servizio Web è accessibile o misurare il variare dei tempi di risposta nel tempo.

2.1.4.1 Netcraft

Netcraft[10] è un servizio che controlla periodicamente molti dei siti di Internet per verificare se sono raggiungibili e calcolare il numero di giorni trascorsi dall’ ultimo riavvio. Oltre ad informazioni dettagliate sui singoli siti Web, Netcraft permette anche di visualizzare delle interessanti statistiche sulla rete in generale. Ad esempio, si può visualizzare dei grafici che mostrano quale sia l’ andamento dei server Web più utilizzati o quali sono i sistemi operativi più diffusi sui server in Internet.

Nonostante non sia disponibile molta documentazione sulla tecnologia utilizzata da Netcraft, la struttura di base dovrebbe essere simile a quella utilizzata in alcuni motori di ricerca: degli agenti si occupano di interrogare i vari siti web e di memorizzare le informazioni che ricavano (in questo caso, non il contenuto delle pagine, ma i tempi di accesso, gli identificativi dei server, ecc...) su di un database centralizzato.

Figura 2.14: Una schermata di Netcraft che mostra diverse informazioni riguardanti sul sito www.superfighetto.it, tra cui l’uptime, il sistema operativo, e il Web server utilizzato

2.1.4.2 eTrust Web Access Control

Il funzionamento di eTrust Web Access Control della Computer Associates[11] è basato sul monitoraggio e la selezione del traffico web relativo ad un server. Una volta ricevuta la richiesta, questo si occupa di verificare le credenziali dell’utente (autenticazione) e le risorse accessibili (autorizzazione); se la richiesta è accettata, il risultato è inviato al browser web della macchina remota.

Più che di uno strumento specifico per il monitoraggio, si stratta di un sistema che permette di incrementare e rendere omogenea la sicurezza su un insieme di server web (ad esempio appartenenti alla stessa rete aziendale).

Il sistema è costituito da tre componenti:

CAPITOLO 2. AGENTI E WEB

- Policy Server
- Policy Manager
- Agenti per il Web.

Il Policy Server fornisce autorizzazioni e autenticazioni agli utenti e agli agenti installati, il Policy Manager fornisce un’interfaccia che permette di definire gli utenti, le loro regole di accesso, i servizi e gli agenti installati.

Gli agenti, specifici per ogni differente tipo di Web Server o Application Server, sono i broker tra gli utenti Web e le risorse; vengono installati sul server e comunicano con il Policy Server per ottenere autenticazioni e autorizzazioni. Sono loro che si occupano di intercettare ogni richiesta fatta dagli utenti e verificare che possa essere inoltrata.

In questo sistema, ogni agente svolge il suo compito in modo completamente indipendente dagli altri, comunicando solo con il Policy Server.

![Figura 2.15: Una schermata del Policy Manager che permette di definire le autorizzazioni per gli utenti](image)

2.2 Temi di ricerca

Per quanto riguarda quello che sarà in futuro il rapporto tra web ed agenti, non si può non parlare dell’idea di immaginare il web com un mondo virtuale. Questa idea non è
certo nuova, possiamo trovarne delle descrizioni che risalgono a prima della diffusione di Internet come fenomeno di massa negli scritti di William Gibson, in cui immagina il ‘cyberspazio’ come una realtà virtuale immersiva basata su una rete mondiale di computer.

Figura 2.16: La rete del futuro, come è immaginata nel film Johnny Mnemonic, basato su un racconto di William Gibson

Sono già stati fatti numerosi tentativi e definiti degli standard per il web 3D (ad esempio VRML[12]), che, a dire il vero, non hanno mai preso molto piede.

Perché il web 3D possa essere utilizzato da un maggior numero di persone, è necessario che vengano soddisfatte alcune condizioni, tra cui:

- avere client con una buona potenza di calcolo e schede grafiche 3D;
- bassa latenza sulla rete e buona ampiezza di banda;
- interfacce utente usabili;
- contenuti che diano valore aggiunto a questa differente modalità di navigazione.

Gli attuali limiti che impediscono al web 3D di avere successo, non sono tanto negli aspetti tecnologici (basti pensare ai giochi multiplayer 3D), ma nella mancanza di un sistema aperto, flessibile e potente, che invece è proprio la caratteristica vincente del web ‘tradizionale’.

2.2.1 Web Agent Support Program

WASP è un progetto che si prefigge di sviluppare un framework che permetta di sviluppare agenti che operino sul web, occupandosi sia la parte di modellazione che degli aspetti implementativi. Le tipologie di agenti di cui questo progetto si occupa sono le più variegate: agenti informativi di ricerca, avatar 3D, sistemi locali e server side, agenti singoli o multipli. Molti lavori che sono stati fatti e che sono visibili sul sito del progetto riguardano le comunità virtuali 3D e il VRML, che viene visto come una possibile evoluzione dell’attuale web “a 2 dimensioni”.

Figura 2.17: L’interfaccia utente di prototipo realizzato nell’ambito del progetto WASP

Il framework che dovrà essere sviluppato si prefigge di supportare i seguenti aspetti:

- navigazione intelligente e recupero di informazioni;
- aggiornamento di documenti e informazioni;
- interfacce utente per applicazioni web;
- documenti dinamici con applets;

CAPITOLO 2. AGENTI E WEB

- descrizioni dichiarative delle credenze degli agenti basate sulle preferenze degli utenti;
- modellazione dichiarativa del comportamento coordinato e cooperativo di agenti software;
- programmazione di sistemi a singolo agente o multi agente\(^{\text{WASP}}\).

![Diagram] Figura 2.18: Le tipologie di agenti web identificate nell’ambito del progetto WASP

Uno dei risultati raggiunti da questo progetto è la definizione di un’architettura per agenti web. Questa architettura, che viene descritta dettagliatamente in [Huang et al 2001], è una semplificazione ed estensione di [Bell 1995]. È stata scelta l’architettura proposta da Bell come base, perché essa presenta importanti vantaggi, tra cui l’essere generica, flessibile e, essendo una estensione di BDI, avvalersi di numerose teorie formali.
CAPITOLO 2. AGENTI E WEB

2.2.2 Mondi virtuali 3D come società di agenti

Un interessante esempio di come l’idea di web 3D è stata coniugata con la tecnologia ad agenti è [Smith et al. 2003] viene descritto un framework per la realizzazione di mondi virtuali 3D, cioè di ambienti multi utente accessibili via rete.

Questo framework non si limita a considerare aspetti come la multi-utenza o un buon...
motore 3d, definisce anche un sistema che permette di associare comportamenti agli elementi del mondo. L’idea è quella di passare dal concetto di mondo virtuale object-oriented ad agent-oriented, cioè un mondo dove ogni oggetto può essere un agente capace di azioni autonome, di ricevere input dall’ambiente tramite sensori sensori e di modificarlo con i suoi attuatori. Il termine ‘Autonomo’ si riferisce al fatto che ogni agente ha un suo thread di controllo che decide da solo quali azioni compiere\cite{Jennings2000}. Gli agenti, che sono situati in un punto preciso dello spazio, sono in grado di costruirsi la loro rappresentazione della situazione e di utilizzarla per fare dei ragionamenti.

È stato definito anche un linguaggio di comunicazione intra-agente, basato su s\cite{SmithBarry1990}, che permette agli agenti di cooperare per raggiungere obiettivi comuni. Ad esempio, in una stanza per conferenze virtuali, l’agente ‘Stanza’ potrebbe avere un sensore per determinare il numero di avatar presenti. Se ritenesse che la stanza fosse troppo piccola potrebbe comunicare con gli agenti ‘Pareti’ per chiederli di spostarsi per far diventare più grande la stanza.
Capitolo 3

Web come sistema ad Agenti

In questo capitolo viene presentato il modello ad agenti per il Web che è oggetto di questo lavoro di tesi. Scopo del modello è definire una base comune da cui possano essere sviluppate differenti applicazioni ad agenti che operano sul Web.

Il lavoro che è stato svolto ha comportato l’ individuazione di un modello ad agenti adatto allo scopo, il suo adattamento e la definizione degli agenti secondo il modello scelto. Il modello utilizzato, MMASS (Multilayered MultiAgent Situated System), verrà descritto in dettaglio in questo capitolo.

3.1 MMASS

MMASS (Multilayered MultiAgent Situated System) [Bandini et al 2002] è un modello di sistema multiagente (MAS) nel quale gli agenti sono posizionati in un ambiente.

In questo modello viene definita esplicitamente la struttura spaziale dell’ambiente dove gli agenti si muovono ed interagiscono. In ogni istante di tempo, ogni posto dello spazio può contenere al massimo un agente (principio di non penetrabilità) ed ogni agente può essere situato su un singolo posto (principio di non-ubiquità).

Gli agenti ereditano le relazioni spaziali definite dal posto che occupano, cioè un agente posizionato sul posto \( p \) è considerato adiacente agli agenti che sono sui posti adiacenti a \( p \) [Bandini et al 2003].

La maggior parte dei sistemi ad agenti prevedono un comunicazione diretta tra gli
agenti, mentre il modello MMASS prevede che la comunicazione sia mediata dallo spazio.

Nel modello sono previste due differenti tipi di interazioni tra agenti, entrambe mediate dalla struttura spaziale:

- interazioni sincrone tra agenti situati su posti attigui;
- interazioni asincrone basate sull’emissione/ricezione di campi.

### 3.1.1 Spazio

Lo spazio costituisce l’ambiente in cui gli agenti di un singolo MASS si muovono ed interagiscono ed è definito come un grafo non orientato di posti. Ogni posto $p$ può essere definito tramite una tripla $\langle a_p, F_p, P_p \rangle$, dove:

- $a_p \in A \cup \{\bot\}$ è l’agente situato sul posto $p$ (se $a_p = \bot$ non c’è nessun agente presente sul posto);
- $F_p \subseteq F$ è l’insieme di campi presenti in $p$, è l’insieme vuoto se non sono presenti campi;
- $P_p \subseteq P$ è l’insieme di posti adiacenti a $p$ [Bandini et al. 2003].

Dato che il grafo non è orientato, la relazione di adiacenza è riflessiva, per cui se il posto $p_a \in P$ è adiacente al posto $p_b \in P$, anche il posto $p_b$ sarà adiacente al posto $p_a$. La relazione di adiacente non è transitiva, infatti se $p_a \in P$ è adiacente a $p_b \in P$ e $p_b$ è adiacente a $p_c \in P$, non è detto che $p_a$ sia adiacente anche a $p_c$.

Il modello MMASS prevede la presenza di più spazi, messi in relazione tra di loro grazie a delle interfacce. Ogni interfaccia è costituita dalla tripla $\langle p_i, p_j, F_\tau \rangle$, dove:

- $p_i, p_j \in P$ sono posti appartenenti a spazi differenti ($i \neq j$);
- $F_\tau$ è il campo interessato dalla interfaccia.

La funzione dell’interfaccia è di mettere in relazione di adiacenza (solo per quanto riguarda la diffusione del campo $F_\tau$) i posti $P_i$ e $P_j$. In questo modo, un campo emesso su di uno spazio, ha la possibilità di diffondersi anche su di un altro spazio.
3.1.2 Campi

Nel modello MMASS, l’ interazione a distanza tra agenti avviene tramite i campi. Si tratta di segnali emessi da agenti e diffusi attraverso l’ambiente che posso essere percepiti da altri agenti.

Questo meccanismo si ispira alla comunicazione tra agenti basata su feromone, ma i campi possono contenere maggiori informazioni, che non siano solo l’intensità. Ad esempio è possibile definire dei campi che contengano dei messaggi in Agent Communication Language\[1\].

I campi si diffondono nello spazio secondo una funzione di diffusione e possono evaporare, cioè diminuire di valore nel tempo fino a non essere più percepibili dagli agenti.

Formalmente un campo è definito come $\langle W_f, Diffusion_f, Compare_f, Compose_f \rangle$, dove:

- $W_f$ è l’insieme di valori che il campo può assumere;
- $Diffusion_f : P \times W_f \times P \to (W_f)^+$ è la funzione di diffusione per il campo che tiene conto di dove è stato generato e con quale intensità;
- $Compare_f : W_f \times W_f \to \{True, False\}$ è la funzione di comparazione e viene usata per verificare se un agente è in grado di percepire un dato campo. Per vedere se un agente è sensibile ad un determinato campo, viene usata questa funzione per comparare il valore del campo con la soglia di sensibilità dell’agente;
- $Compose_f : (W_f)^+ \to W_f$ è la funzione di composizione, che specifica come i campi possono essere combinati (viene utilizzata, ad esempio, quando in un sito sono presenti più campi dello stesso tipo e si vuole considerarli come un unico campo).

3.1.3 Agenti

Gli agenti del modello MMASS sono caratterizzati da un tipo, uno stato e una posizione.

Il tipo dell’agente viene descritto specificando l’insieme degli stati che l’agente può assumere, la funzione di percezione e l’insieme di azioni che può eseguire.

\[1\]Maggiori informazioni su ACL possono essere trovate su [Labrou et al 1999] e [FIPA 2001].
La funzione di percezione, che esprime le sensibilità dell’agente ai campi, associa ad ogni stato dell’agente un vettore di coppie che indicano un coefficiente e una soglia.

Il coefficiente viene applicato ai valori del campo percepito mentre la soglia esprime il livello di sensibilità dell’agente allo specifico campo.

L’agente percepisce il campo solo se il valore ricevuto, moltiplicato per il coefficiente, è superiore alla soglia [Rizzi 2002].

Formalmente un agente è definito come $(s, p, \tau)$, dove:

- $s \in \sum_{\tau}$ è lo stato dell’agente;
- $p \in P$ è il posto dove è posizionato l’agente;
- $\tau$ è il tipo di agente.

Il tipo di agente $\tau$ è definito come $(\sum_{\tau}, Perception_{\tau}, Action_{\tau})$, dove:

- $\sum_{\tau}$ è l’insieme di stati che un agente di tipo $\tau$ può assumere;
- $Perception_{\tau} : \sum_{\tau} \rightarrow [N \times W_{1}] \ldots [N \times W_{|F|}]$ è una funzione che associa ad ogni stato che l’agente può assumere una coppia che rappresenta il coefficiente di recettività e la soglia sensibilità per il tipo di campo;
- $Action_{\tau}$ rappresenta i comportamenti specifici per gli agenti di tipo $\tau$.

I comportamenti specificano quello che gli agenti possono fare: quando e come modificano il loro stato o la loro posizione, come interagiscono con gli altri agenti, come sono influenzati dai campi.

I comportamenti possono essere specificati attraverso un linguaggio che definisce le seguenti primitive:

- $reaction (s, a_{p1}, a_{p2}, \ldots, a_{pn}, s')$ permette all’agente $a$ situato sul posto $p$ di interagire in modo sincrono con gli agenti $a_{p1}, a_{p2}, \ldots, a_{pn}$ situati sui posti $p1, p2, \ldots, pn$ adiacenti al posto $p$. L’interazione avviene con agenti che hanno accettato di far parte dell’interazione, secondo il loro stato interno. L’interazione sincrona si realizza tramite un processo a due passi. Innanzitutto viene definito un protocollo per sincronizzare l’insieme degli agenti autonomi e quando un agente vuole reagire con gli agenti adiacenti, attiva il processo di accordo che stabilisce quali
sono gli agenti adiacenti che hanno scelto di interagire. Dopo il processo di accordo, viene eseguita la reazione sincrona tra l’insieme degli agenti che si sono accordati. L’effetto di questa interazione il cambio dello stato del sito da \( s \) a \( s' \);

- \( \text{emit} (s, f, p) \) permette di emettere un nuovo campo \( f \) sul posto \( p \), cioè sul posto su cui è posizionato l’agente;

- \( \text{trigger} (s, f_i, s') \) specifica che l’agente deve cambiare il suo stato da \( s \) a \( s' \) quando percepisce il campo \( f_i \);

- \( \text{transport} (p, f_i, q) \) permette all’agente di cambiare posizione, dal posto \( p \) al posto \( q \) quando percepisce il campo \( f_i \).

Le azioni \( \text{trigger} \) e \( \text{transport} \) sono dette \textit{intra-agent} perché coinvolgono solo l’agente che le esegue mentre le azioni \( \text{emit} \) e \( \text{reaction} \) sono \textit{inter-agent} in quanto coinvolgono più agenti.

### 3.2 Adattamento del modello MMASS

Per rappresentare il Web come un sistema ad agenti, si è scelto di utilizzare il modello MMASS, rilassandone però alcuni vincoli.

Ciò si è reso necessario perché in modello MMASS si ispira al mondo fisico e ne eredita alcune proprietà, mentre qui si vuole modellare uno spazio ‘virtuale’ dove si vuole prendere utilizzare la realtà solo come metafora a cui ispirarsi.

Rispetto al modello MMASS, viene rilassato il vincolo di non-penetrabilità, cioè possono essere presenti più agenti per ogni posto. Questo vincolo presente in MMASS è dovuto al fatto che nel mondo reale due entità non possono trovarsi contemporaneamente nella stessa posizione, mentre, se immaginiamo il Web come uno spazio e le pagine come dei posti, possiamo benissimo pensare che più agenti possano essere presenti contemporaneamente nello stesso posto.

La definizione del posto \( p \) può essere quindi data dalla tripla \( \langle A_p, F_p, P_p \rangle \), dove:

- \( A_p \subseteq A \) è l’insieme di agenti situati sul posto \( p \);

- \( F_p \subseteq F \) è l’insieme di campi presenti in \( p \);
• $P_p \subset P$ è l’insieme di siti adiacenti a $p$.

Questa modifica implica anche che l’insieme degli agenti situati nei posti vicini diventa $A_p \cup A_{p1} \cup A_{p2} \cup \ldots \cup A_{pn}$, dove $p_1 \ldots p_n$ sono i posti adiacenti a $p$. Tra gli agenti situati nei posti attigui, vengono considerati anche tutti gli agenti situati sul posto $p$.

Anche la definizione di agente, a quella propria del modello MMASS, è stata modificata, in quanto un agente può essere presente su più posti contemporaneamente, purché facciano parte di spazi differenti. Questa modifica è stata introdotta perché risulta utile al fine della modellazione definire differenti spazi e poter posizionare un agente contemporaneamente su tutti questi spazi.

Un agente viene quindi formalmente definito come $(s, P_a, \tau)$, dove:

• $s \in \sum$ è lo stato dell’agente;
• $P_a = \{p_{a1}, p_{a2}, \ldots, p_{an}\}$ è l’insieme dei posti dove l’agente è posizionato, con $p_{a1} \in P_1, p_{a2} \in P_2, \ldots, p_{an} \in P_n$ e $P_1 \neq P_2 \neq \ldots \neq P_n$;
• $\tau$ è il tipo di agente.

3.3 Modellazione del Web tramite MMASS

Le due principali componenti di MMASS che sono state utilizzate per definire un modello per gli agenti Web sono lo spazio e gli agenti.

Lo spazio, che è una struttura dinamica in cui gli agenti si muovono, è costituito da tutte le pagine e dai loro collegamenti. Ogni pagina è mappata come un posto dello spazio, mentre ogni hyperlink tra due pagine viene modellato come un collegamento orientato tra i due posti delle spazio corrispondenti alle due pagine. Il collegamento parte dal posto corrispondente alla pagina il cui è presente il link ed è diretto verso il posto corrispondente alla pagina verso cui punta il link.

La necessità che lo spazio sia una struttura dinamica deriva da ciò che si vuole modellare: un sito Web. Esso non è un’ entità statica, ma dinamica, infatti, a causa dell’ aggiunta di nuove pagine, della loro modifica e cancellazione, la sua struttura varia nel tempo.
CAPITOLO 3. WEB COME SISTEMA AD AGENTI

37

Figura 3.1: Rappresentazione di pagine e link tratta da una presentazione del World Wide Web fatta dal Cern del 1992

Come nel modello MMASS, possono essere definite differenti strutture spaziali. Queste possono essere utili per rappresentare altri tipi di collegamenti tra i posti, come, ad esempio uno spazio in cui tutti i documenti creati da uno stesso autore sono collegati tra di loro, oppure spazi in cui i siti sono collegati tra di loro secondo l’argomento che trattano o ancora spazi che rappresentano il percorso effettuato da un utente durante la navigazione del Web e che vengono creati dinamicamente secondo i suoi spostamenti.

Figura 3.2: Web e rappresentazione ad agenti

Gli agenti sono utilizzati per modellare gli utenti che navigano un sito Web. Ad ogni utente è associato un agente che viene posizionato su di un posto che rappresenta il documento che l’utente stà consultando.

Ogni nuova richiesta di una pagina Web da parte di un utente, corrisponde ad uno spostamento dell’agente.
Così come definito nel modello MMASS, ogni agente ha la percezione dello spazio in cui si trova ed è sensibile alla presenza di altri agenti (rivelata attraverso la presenza di campi). Ciò costituisce una innovazione rispetto al modello tradizionale del Web, in cui ogni utente non ha la percezione di chi stà navigando insieme a lui e permette di realizzare applicazioni che sfruttino queste potenzialità, come, ad esempio, una chat tra tutti gli utenti che stanno visitando una data pagina di un sito.

La vita degli agenti dipende dalla sessione di navigazione degli utenti. Appena un nuovo utente visita il sito viene creato l’agente corrispondente che lo seguirà per tutta la durata della sua navigazione. Dopo un determinato lasso di tempo in cui l’utente non fa più richieste al sito, il suo agente verrà rimosso.

Un ulteriore possibilità è la generazione di uno spazio secondo la navigazione di un utente, come viene fatto il Footprints 2.1.2.3. Questo spazio può fornire interessanti informazioni su come ogni utente percepisca la struttura ed acceda ai contenuti presenti in un sito Web e può anche servire a fornire contenuti personalizzati secondo la storia dell’utente.

Per evitare che ogni volta che un utente visiti il sito si ricominci da capo a raccogliere informazioni sui suoi percorsi, si può prevedere che che prima di rimuovere l’agente venga salvato il suo stato e che, se l’utente ritornerà sul sito, gli verrà associato il medesimo agente.
Capitolo 4

Ambiente software per il modello MMASS

Uno degli obiettivi di questa tesi è stato la realizzazione di una piattaforma che permetta lo sviluppo di applicazioni basate sul modello MMASS.

Come punto di partenza per la sua realizzazione è stata usata la piattaforma sviluppata da Andrea Rizzi[1].

In questo capitolo verranno presentati in dettaglio la piattaforma, i principali componenti che la compongono e un esempio del suo utilizzo per la realizzazione di una semplice applicazione ad agenti.

4.1 Piattaforma

La definizione delle funzionalità che la piattaforma deve offrire, la sua modellazione UML e la successiva realizzazione sono state semplificate dal fatto che esisteva già una piattaforma a cui fare riferimento, cioè quella realizzata da Andrea Rizzi durante la sua tesi di laurea.

Una prima fase di analisi del suo lavoro ha permesso di individuare quali potevano essere le funzionalità da offrire agli sviluppatori e cosa mancava per poter sviluppare applicazioni che si basassero sul modello MMASS.

[1] Vedi [Rizzi 2002]
Successivamente si è analizzato il modello UML, che è stato modificato, nell’ottica di realizzare una piattaforma che rappresentasse il più fedelmente possibile il modello MMASS, che fosse più facilmente estendibile e che lasciasse gli sviluppatori meno vincolati dalle scelte implementative fatte da chi ha realizzato la piattaforma.

Per quanto riguarda l’implementazione, si è posta grande attenzione alla realizzazione di classi che fossero efficienti dal punto di vista computazionale, perché ciò è fondamentale nella realizzazione di una piattaforma che sia effettivamente utilizzabile. Questo ha comportato di dover riscrivere molte delle classi che costituivano la precedente piattaforma.

Nella definizione delle API esposte agli sviluppatori, si è cercato di semplificarle il più possibile, per facilitare il suo utilizzo.

A livello implementativo, le classi della piattaforma sono organizzate all’interno di package, di cui, alcuni si rifanno agli elementi definiti nel modello concettuale mentre altri (come ad esempio la visualizzazione grafica o la politica di diffusione dei campi) non trovano riscontro nel modello.

I package che compongono la piattaforma sono i seguenti:

**mmass** package che contiene la piattaforma;

**platform** contiene tutte le classi ed i package che formano il cuore della piattaforma;

**field** campi e gestione dei campi;

**diffusion** politiche di diffusione dei campi;

**network** rete di posti ed archi;

**site** siti;

**space** gestione degli spazi;

**regularspace** spazi regolari;

**graphspace** spazi non regolari;

**util** classi generali di supporto usate da altri package;

**ide** contiene tutto ciò che riguarda la visualizzazione grafica;

**main** inizializzazione dell’interfaccia grafica e sua gestione;

**ui** visualizzazione grafica degli spazi e degli agenti;

**inspector** visualizzazione delle proprietà di un oggetto.
La piattaforma è stata sviluppata seguendo uno dei principi della progettazione orientata agli oggetti, secondo cui si devono tenere separate le cose che cambiano da quelle che restano uguali [Eckel 2000].

L’approccio seguito ha comportato la definizione degli elementi del modello MMASS in termini di interfacce oppure di classi astratte. Queste classi rappresentano le funzionalità che vengono offerte agli sviluppatori. Tali sviluppatori, possono riutilizzare le funzionalità della libreria tramite l’interfaccia a loro nota, senza preoccuparsi di come il sistema è stato implementato. In questo modo, lo sviluppatore trae vantaggio dalla conoscenza di quali servizi può usare rispetto a quelli che può ignorare [Rizzi 2002].

4.2 Spazi

Lo spazio descrive l’ambiente dove gli agenti sono posizionati ed i campi si diffondono. È costituito da un insieme di posti. Gli spazi si dividono in due principali categorie:

- spazi regolari;
- spazi generici.

Gli spazi regolari sono caratterizzati da una struttura regolare nota a priori ed immutabile, cioè una volta che lo spazio è stato creato (cioè una volta che l’oggetto corrispondente
è stato istanziato) non potrà più essere modificato. Non potranno cioè essere aggiunti o rimosi posti e collegamenti.

Gli spazi generici, invece, sono dinamici, cioè possono essere aggiunti e rimosi posti e collegamenti a runtime (nulla però vieta di avere spazi generici creati a priori che non verranno più modificati.)

Ad ogni spazio è associato un identificativo (il suo nome) ed una politica di diffusione dei campi, rappresentata da una classe che implementa l’interfaccia DiffusionSpace.

Tutte le tipologie di spazio vengono implementate estendendo la classe AbstractSpace.

**Figura 4.2: Classi che compongono il package space**

### 4.2.1 Posti

I posti sono i siti su cui possono essere situati agenti e campi. Rispetto alla definizione del modello MMASS, i posti sono stati implementati in modo da poter ospitare anche più di un agente. Nulla però vieta di utilizzarli come quelli definiti nel modello MMASS, aggiungendo il vincolo di un solo agente per sito.

Ad ogni posto sono associate le seguenti informazioni:

- un identificativo numerico (id);
- lo spazio di cui fa parte;
La classe attraverso cui sono implementati i posti è \textit{Site}, mentre la sua estensione \textit{PositionedSite}, definisce un posto che ha come informazione aggiuntiva la sua posizione nello spazio.

In generale non è necessario istanziare manualmente le classi corrispondenti ai posti, dato che gli spazi regolari si occupano loro della creazione di tutti i posti, mentre gli spazi generici hanno già definiti dei metodi che permettono di creare nuovi posti.

Ad ogni posto è possibile associare un insieme di proprietà (\textit{property}); ogni proprietà è caratterizzata da un nome che la identifica (\textit{key}) e da un valore corrispondente (\textit{value}). L’ identificativo deve essere una stringa, mentre il valore della proprietà può essere un qualunque oggetto. A runtime è possibile eseguire qualunque operazione con le proprietà ed i loro valori (cioè aggiungerne di nuove, modificarne i valori, rimuoverle).

Per impostare una proprietà si utilizza il metodo \textit{setProperty(key,value)} mentre per ottenere il valore bisogna usare \textit{getProperty(key)}. La rimozione di una proprietà avviene impostando il suo valore a \textit{null}.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{site.png}
\caption{Figura 4.3: Classi che compongono il package \textit{site}}
\end{figure}
4.2.2 Spazi regolari

Gli spazi regolari sono costituiti da posti e collegamenti disposti secondo uno schema ordinato.

Per creare uno spazio regolare basta specificare alcuni suoi parametri, come, ad esempio, le dimensioni larghezza ed altezza di una griglia, e verranno creati autonomamente tutti i posti e i collegamenti tra di loro. Un esempio di spazio regolare è una griglia bidimensionale i cui posti sono collegati tra loro tramite l’intorno di Von Neumann.

Attualmente sono stati implementati i seguenti tipi di spazi regolari:

- Spazio vuoto
- Griglia
- Griglia con l’intorno di VonNeumann
- Griglia toroidale con l’intorno di VonNeumann
- Griglia con l’intorno di Moore
- Griglia toroidale con l’intorno di Moore
- Griglia esagonale

4.2.2.1 VoidSpace

La classe VoidSpace rappresenta lo spazio vuoto, cioè che non contiene nessun posto. Come per tutti gli altri spazi regolari, non è possibile aggiungere nuovi posti.

Essendo uno spazio con una struttura regolare nota a priori, lo spazio vuoto è stato considerato uno spazio regolare.

4.2.2.2 GridSpace

È una griglia bidimensionale i cui posti sono tutti non collegati tra di loro. Tutti gli altri spazi regolari, ad esclusione di VoidSpace, sono delle sue generalizzazioni.

Il costruttore richiede come parametri (oltre al nome identificativo dello spazio ed eventualmente alla classe di diffusione scelta) la larghezza e la lunghezza della griglia.
Ogni posto della griglia è caratterizzato da un \textit{id} numerico progressivo. La numerazione parte dal primo posto in alto a sinistra che ha come \textit{id} il numero 0. Il posto alla sua destra è il numero 1, e così via. Il primo posto di ogni altra riga ha come \textit{id} quello successivo a quello dell’ultimo posto della riga precedente.

\textbf{4.2.2.3 VonNeumannGridSpace}

La classe \textit{VonNeumannGridSpace}, che estende \textit{GridSpace}, è una griglia bidimensionale i cui posti sono collegati tra di loro con l’intorno di Von Neumann, cioè ogni posto ha (fino a) 4 vicini: il posto alla sua destra, quello alla sua sinistra, quello sopra e quello sotto.

Essendo una griglia e non uno spazio toroidale (cioè non chiuso su se stesso) tutti i posti che si trovano sui bordi hanno meno di 4 vicini.

\begin{figure}
\centering
\includegraphics[width=0.2\textwidth]{intorno_von_neumann.png}
\caption{Intorno di Von Neumann}
\end{figure}

\textbf{4.2.2.4 VonNeumannTorusSpace}

La classe \textit{VonNeumannTorusSpace}, che estende \textit{VonNeumannGridSpace} e implementa l’interfaccia \textit{TorusSpace}, rappresenta uno spazio toroidale, cioè chiuso su se stesso, in cui i posti della prima riga sono attigui con quelli dell’ultima e quelli della prima colonna sono attigui con quelli dell’ultima. L’intorno dei posti è quello di VonNeumann.

A differenza di quanto accade per la classe \textit{VonNeumannGridSpace}, in questo caso tutti i posti hanno esattamente 4 vicini.

Questa classe implementa l’interfaccia \textit{TorusSpace}, che non definisce alcun metodo, ma serve solamente per contrassegnare tutti gli spazi toroidali.
4.2.2.5 MooreGridSpace

Attraverso la classe MooreGridSpace, che estende GridSpace, è implementata una griglia i cui posti sono collegati con l’ intorno di Moore.

I posti interni hanno 8 vicini, mentre i posti collocati sui bordi hanno un numero inferiore di vicini.

![Intorno di Moore](image)

Figura 4.6: Intorno di Moore

4.2.2.6 MooreTorusSpace

La classe MooreTorusSpace estende MooreGridSpace e implementa l’ interfaccia TorusSpace.

Si tratta di uno spazio toroidale i cui posti sono collegati tra di loro con l’ intorno di Moore. Trattandosi di uno spazio toroidale, ogni posto ha esattamente 8 altri posti adiacenti a lui.

![Esempio di spazi toroidali 4 x 4 con l’intorno di Moore](image)

Figura 4.7: Esempio di spazi toroidali 4 x 4 con l’ intorno di Moore in cui stato è evidenziato in giallo un posto ed in arancione i posti a lui adiacenti
4.2.2.7 HexagonalGridSpace

La classe HexagonalGridSpace, che estende GridSpace, è una griglia i cui posti sono collegati tra di loro esagonalmente, cioè ogni posto ha (fino a 6) vicini.

La disposizione dei posti è sempre quella della griglia regolare, solamente che rendere più chiara la struttura di questo spazio è stata definita un’ apposita classe per la visualizzazione che sposta a destra tutti i nodi delle righe pari, in modo che l’ intorno di ogni posto formi un esagono.

Figura 4.8: Esempio di griglia esagonale 6 x 7

4.2.2.8 Visualizzazione

La classe GridView permette la visualizzazione di spazi regolari a griglia, cioè che estendono GridSpace.

Per ottenere delle visualizzazioni personalizzate, è possibile impostare i seguenti parametri:

- colore di sfondo (default: bianco);
- immagine di background (default: nessuna);
- colore di base dei posti (default: nero);
- colore degli archi (default: nero);
- classe per la visualizzazione degli agenti (default: DefaultDrawAgent);
- classe per la visualizzazione dei posti (default: DefaultDrawNode);
- visualizzazione dei posti (default: true);
• visualizzazione *id* posti (*default: false*);

• visualizzazione degli archi (*default: true*);

• visualizzazione freccia orientamento archi (*default: false*);

• intervallo di aggiornamento della visualizzazione (*default: 500ms*);

• distanza tra i posti (*default: 25px*).

Figura 4.9: Esempio di utilizzo di GridView per la visualizzazione di una griglia bidimensionale con l’intorno di Von Neumann

Un’ ulteriore classe che permette di visualizzare degli spazi a griglia, è *IsoGridView*, che fornisce una visualizzazione isometrica.

Una sua interessante estensione è *FieldHeightIsoGridView*, in cui l’ altezza di ogni posto è determinata dall’ intensità di un dato campo in quel posto moltiplicata per un moltiplicatore. La presenza del moltiplicatore permette di decidere se creare delle ‘montagne’ o delle ‘depressioni’ dove il campo è maggiormente intenso.
Figura 4.10: Esempio di utilizzo di GridViewFieldHeightIsoGridView per la visualizzazione di una griglia bidimensionale con l’intorno di Von Neumann

Figura 4.11: Esempio di utilizzo di GridViewFieldHeightIsoGridView e di GridView per la visualizzazione del medesimo spazio
4.2.3 Spazi generici

Gli spazi generici possono essere descritti come dei grafi orientati, in cui ogni nodo è un posto ed ogni arco indica un’adiacenza tra due posti.

Mentre nel modello MMASS la relazione di adiacenza tra due posti è riflessiva (cioè se il posto $a$ è adiacente al posto $b$ allora anche il posto $b$ sarà adiacente al posto $a$), nello sviluppo della piattaforma si è scelto di implementare dei collegamenti orientati tra i posti, cioè unidirezionali. Questo non è in alcun modo un limite, visto che per creare uno spazio che rispetti il modello MMASS basta creare due collegamenti tra due posti, uno che parte dal primo verso il secondo ed uno che parte dal secondo verso il primo. È stata fatta questa scelta per essere, dove possibile, il maggior generici possibili rispetto al modello.

L’approccio che è stato utilizzato per creare degli spazi generici è di utilizzare dei file XML che ne descrivono la struttura, specificando i posti (con la loro eventuale posizione) e gli archi che li collegano.

4.2.3.1 GraphSpace

La classe GraphSpace implementa uno spazio a grafo orientato in cui i nodi sono oggetti di tipo Site.

Questo tipo di posti non sono caratterizzati da una posizione spaziale e la mancanza di questa informazione rende più complessa la visualizzazione dello spazio.

Questa classe può essere utile per creare spazi non regolari generati dinamicamente, in quanto prevede metodi per l’aggiunta e la rimozione di posti e di collegamenti a runtime.

4.2.3.2 PositionedGraphSpace

La classe PositionedGraphSpace estende GraphSpace ed ogni nodo è un oggetto della classe PositionedSite, cioè un posto caratterizzato da una posizione nello spazio.

La posizione di un sito è data da un’oggetto che implementa l’interfaccia SitePosition. Attualmente sono state realizzate due implementazioni: Site2DPosition e Site3DPosition, che rappresentano rispettivamente una posizione in uno spazio bidimensionale ed una in uno spazio tridimensionale.
Anche questa classe, come *GraphSpace*, prevede l’aggiunta e le rimozione di posti e di collegamenti a *runtime*.

### 4.2.3.3 XmlSpace

Tramite la classe *XMLSpace*, che estende *PositionedGraphSpace*, è possibile leggere e memorizzare uno spazio tramite un file XML in cui vengono specificati tutti i posti ed i loro collegamenti.

Un esempio di un file XML che viene generato quando viene salvato uno spazio è il seguente:

```xml
<?xml version="1.0"?>
<!DOCTYPE graph[
<!ELEMENT graph ( node*, arc* )>
<!ELEMENT node ( att* )>
<!ATTLIST node id CDATA #REQUIRED>
<!ATTLIST node label CDATA #REQUIRED>
<!ELEMENT att EMPTY>
<!ATTLIST att name CDATA #REQUIRED>
<!ATTLIST att value CDATA #REQUIRED>
<!ELEMENT arc EMPTY>
<!ATTLIST arc source CDATA #REQUIRED>
<!ATTLIST arc target CDATA #REQUIRED>
]

<graph>
  <node id="1" label="">
    <att name="x" value="47.0"/>
    <att name="y" value="48.0"/>
  </node>
  <node id="2" label="">
    <att name="x" value="87.0"/>
    <att name="y" value="48.0"/>
  </node>
  <node id="3" label="">
    <att name="x" value="48.0"/>
    <att name="y" value="79.0"/>
  </node>
  <arc source="1" target="2"/>
  <arc source="1" target="3"/>
  <arc source="2" target="1"/>
  <arc source="3" target="1"/>
</graph>
```
I file XML che vengono generati contengono il DTD (Document Type Definition) che permette di verificarne la validità.

Per la rappresentazione degli spazi in XML sono stati definiti i seguenti tag:

- **<node>** specifica un posto nello spazio. È caratterizzato da un identificativo univoco (*id*), da un’ etichetta e da un’insieme di sotto-nodi di tipo **<att>**;

- **<att>** specificano i parametri della posizione. Nel caso di uno spazio bidimensionale avremo due nodi **<att>** per ogni **<node>** e conterranno i valori *x* e *y* della posizione. L’utilizzo di questo tag permette di specificare informazioni aggiuntive riguardanti un posto, come, ad esempio, il colore;

- **<arc>** rappresenta un arco (orientato) tra due posti. È caratterizzato dai parametri *source* e *target*, che corrispondono agli identificativi del posto da cui parte l’arco e di quello in cui arriva.

### 4.2.3.4 Visualizzazione

La classe *GraphView* permette di visualizzare spazi di tipo *PositionedGraphSpace* in cui la posizione di ogni posto è un oggetto di tipo *Site2DPosition*, cioè spazi bidimensionale, mentre non è stata fatta nessuna implementazione per quanto riguarda la visualizzazione di spazi tridimensionali.

Dato che ogni posto è caratterizzato da una posizione bidimensionale, il processo di visualizzazione è abbastanza semplice: prima vengono disegnati i posti nella loro posizione, poi vengono disegnati gli archi tra il centro dei posti, infine, ogni 500ms vengono disegnati, per ogni posto, i campi e gli agenti presenti.

La visualizzazione può essere personalizzata secondo le proprie esigenze impostando i seguenti parametri:

- colore di sfondo (*default*: bianco);
- immagine di background (*default*: nessuna);
- colore di base dei posti (*default*: nero);
- colore degli archi (*default*: nero);
- classe per la visualizzazione degli agenti (*default*: *DefaultDrawAgent*);
• classe per la visualizzazione dei posti (default: DefaultDrawNode);
• visualizzazione dei posti (default: true);
• visualizzazione id posti (default: false);
• visualizzazione degli archi (default: true);
• visualizzazione freccia orientamento archi (default: false);
• intervallo di aggiornamento della visualizzazione (default: 500ms).

Per quanto riguarda la visualizzazione degli agenti è possibile specificare la classe (che deve implementare l’interfaccia DrawAgent) che si vuole usare per visualizzarli. In questo modo è possibile creare differenti classi per visualizzare gli agenti con forme differenti, ad esempio quadrati, oppure attraverso un’immagine caricata da un file. La classe DefaultDrawAgent disegna gli agenti come dei cerchi colorati.

Anche la visualizzazione dei posti e dei campi può essere personalizzata attraverso la definizione di una classe. In questo caso si tratta di DrawNode che si occupa sia di visualizzare i posti che i campi.

L’implementazione di default (DefaultDrawNode) disegna i posti ed i campi come dei cerchi colorati.

Figura 4.12: Esempio di utilizzo di GraphView per la visualizzazione di uno spazio caricato da un file XML.
4.2.3.5 Creazione e modifica degli spazi

Nella piattaforma è implementato, nella classe GrapEditor (che estende GraphView), un editor di grafi bidimensionali.

L’editor permette all’utente di effettuare, tramite un’interfaccia grafica, le seguenti operazioni:

- creare un nuovo spazio;
- caricare uno spazio esistente;
- memorizzare su di un file la struttura di uno spazio;
- aggiungere dei nuovi posti;
- spostare dei posti;
- cancellare dei posti esistenti;
- creare nuovi collegamenti tra i posti;
- cancellare collegamenti tra i posti.

Questo editor permette di effettuare modifiche sullo spazio anche a runtime, ad esempio, permette di modificare la struttura di uno spazio mentre su di esso sono presenti degli agenti. Si possono gestire tutti gli spazi non regolari che sono delle istanze della classe PositionedGraphSpace, mentre gli spazi di tipo XMLSpace, possono essere anche caricati e memorizzati, oltre che modificati.

Come la classe da cui eredita, anche questa permette di caricare un’immagine di sfondo, e ciò può essere utile per creare uno spazio ricalcando un’immagine. Ad esempio, possiamo caricare come immagine la pianta di un edificio e possiamo creare uno spazio da utilizzare in una simulazione ricalcando la planimetria sottostante.
4.3 Agenti

Gli agenti della piattaforma sono realizzati in modo da poter implementare agenti del tipo MMASS e sono quindi caratterizzati da una posizione su di un posto in uno spazio, da delle percezioni dell’ ambiente, da una capacità di muoversi, di generare campi e da dei comportamenti.

La piattaforma sviluppatata prende in considerazione gli agenti descrivendo e supportando le operazioni principali che possono compiere.

Per implementare un agente è sufficiente estendere la classe astratta `mmass.agent.AbstractAgent`. Sono già disponibili due differenti estensioni di questa classe, `SingleLayerAgent` e `MultiLayerAgent`, che rappresentano rispettivamente l’ agente che è situato su di un solo spazio e l’ agente che si trova contemporaneamente su più spazi differenti.

Per definire un agente è sufficiente estendere una di queste due classi e ridefinire i seguenti metodi:

- **init** questo metodo è richiamato all’ avvio dell’ agente e si occupa dell’ inizializzazione dell’ agente stesso;

- **action** è richiamato a intervalli regolari di tempo; permette di caratterizzare il comportamento dell’ agente;

- **cleanup** viene chiamato quando il funzionamento dell’ agente viene interrotto.

Figura 4.13: Esempio di utilizzo dell’ editor per creare uno spazio a partire dalla planimetria di un ambiente e successivo utilizzo dello spazio in una simulazione.
CAPITOLO 4. AMBIENTE SOFTWARE PER IL MODELLO MMASS

Non è necessario ridefinire tutti questi metodi, ad esempio, un agente immobile che si limita a generare un campo quando viene creato si può limitare a ridefinire `init`, mentre un agente che si muove alla ricerca di un determinato valore di campo, deve ridefinire `action`.

Le classi `SingleLayerAgent` e `MultiLayerAgent` mettono a disposizione dello sviluppatore degli agenti gli stessi metodi, solo con un numero di parametri differenti, in quanto i metodi di `MultiLayerAgent` permettono di specificare a quale degli spazi su cui si trova l’ agente si riferisci l’ operazione voluta. Ad esempio, il metodo `getMaxAdj`, che permette di vedere in quali dei posti vicini è maggiore il valore di un dato tipo di campo, prevede come parametro in `MultiLayerAgent`, oltre al nome del tipo di capo, lo spazio a cui si riferisce la nozione di vicinanza dei posti.

I principali metodi messi a disposizione sono i seguenti:

- `emit(field)` permette all’ agente di emettere un campo sul posto in cui si trova (o su uno dei posti su cui si trova nel caso di agente multi layer);
- `getFields()` fornisce l’ elenco dei campi presenti;
- `getField(type)` restituisce, se è presente, un campo di un dato tipo;
- `getMaxAdj(type)` restituisce una lista dei posti adiacenti in cui il valore del campo del tipo dato è massimo;
- `getRandomMaxAdj(type)` restituisce un elemento a caso tra i risultati di `getMaxAdj`;
- `getMinAdj(type)` restituisce una lista dei posti adiacenti in cui il valore del campo del tipo dato è minimo o nullo;
- `getRandomMinAdj(type)` restituisce un elemento a caso tra i risultati di `getMinAdj`;
- `getRandomAdj()` restituisce un posto adiacente a caso;
- `tryToMove(site)` muove l’ agente in un nuovo posto, a patto che questo non contenga altri agenti;
- `moveTo(site)` muove l’agente su di un nuovo posto, anche se questo contiene già altri agenti;

Di seguito verranno forniti alcuni esempi di differenti agenti implementati.
Agente immobile:

```java
public class AgentA extends SingleLayerAgent {
    public AgentA(String name, Site site) {
        super(name, site);
    }
}
```

Agente immobile che genera un campo di tipo $B$ al momento della sua inizializzazione:

```java
public class AgentB extends SingleLayerAgent {
    public AgentB(String name, Site site) {
        super(name, site);
    }
    public void init() {
        emit(new Field("B", 10, 1, 0, 10, Color.red));
    }
}
```

Agente che si muove a caso e genera un campo di tipo $C$ che evaporà:

```java
public class AgentC extends SingleLayerAgent {
    public AgentC(String name, Site site) {
        super(name, site);
    }
    public void action() {
        tryToMove(getRandomAdj());
        emit(new Field("C", 3, 0.5, 0.2, 3, Color.blue));
    }
}
```

Agente che si muove verso i posti in cui l’intensità del campo $D$ è maggiore:

```java
public class AgentD extends SingleLayerAgent {
    public AgentD(String name, Site site) {
    }
```
super(name, site);
}
public void action()
{
    tryToMove(getRandomMaxAdj("D"));
}

4.4 Campi

I campi costituiscono il meccanismo di interazione indiretta a distanza tra gli agenti. La comunicazione viene mediata da una infrastruttura di diffusione che si basa sullo spazio.

I campi vengono emessi dagli agenti sul posto in cui si trovano e sono caratterizzati da un valore di intensità che si attenua man mano che il campo viene diffuso ai posti vicini. La diffusione continua fino a quando il valore dell’intensità del campo è maggiore di zero.

Oltre ad attenuarsi durante la diffusione, i campi possono evaporare, cioè, la loro intensità può diminuire con il passare del tempo. Ad ogni campo è possibile associare una serie di informazioni, semplici come può essere il colore o complesse come dei messaggi ACL (Agent Communication Language).

Le classi che costituiscono il package field sono le seguenti:

FieldId è l’identificativo univoco di ogni campo emesso. È costituito da un identificatore di spazio, del posto sorgente e da un numero progressivo di campi emessi.

Field questa classe, di cui tutti gli attributi sono immutabili (frozen), contiene tutte le informazioni statiche su di un determinato tipo di campo. Le informazioni che contiene sono il nome, l’intensità iniziale, i gradi di evaporazione e di attenuazione, il colore. Estendendo questa classe è anche possibile definire delle funzioni per l’evaporazione e l’attenuazione. Una volta che un oggetto di questo tipo è stato creato, non è più possibile modificare il valore sui attributi.

ActiveFieldInfo rappresenta un campo emesso, cioè un campo posizionato su di un posto. Ad ogni istanza sono associati un oggetto Field ed un identificativo (FieldId). Questa classe fornisce i metodi per ottenere i valori correnti dell’intensità del campo, del suo colore e delle sue proprietà. Ciò, man mano che un campo
CAPITOLO 4. AMBIENTE SOFTWARE PER IL MODELLO MMASS

si modifica nel tempo (ad esempio perché si attenua) questa classe fornisce differenti valori, anche se la struttura che contiene i dati del campo (Field) non viene mai modificata. Ciò permette di condividere una singola istanza di Field tra più istanze di campi emessi.

**FieldGroup** questa classe, che estende ActiveFieldInfo, rappresenta un gruppo di campi emessi dello stesso tipo. Permette di ottenere raggruppati tutti i dati riguardanti un dato tipo di campo.

**FieldsManager** si occupa della gestione dei campi presenti su di un determinato posto, operando in modo asincrono. Per ogni posto è presente un’ istanza di questa classe che ha l’ incarico di processare i campi appena arrivati, diffondere i campi ai siti vicini e di rimuoverli quando il valore della loro intensità diviene nullo.

Figura 4.14: Classi che compongono il package field

Per emettere un campo, un agente deve creare una nuova istanza della classe Field. Il costruttore della classe, che permette di specificare i valori che caratterizzeranno il campo, prevede che vengano specificati i seguenti parametri:

- **type** nome del campo;
- **intensity** valore iniziale dell’ intensità del campo;
- **attenuation** valore di attenuazione dovuto alla diffusione del campo nello spazio.

Questo valore viene sottratto all’ intensità attuale del campo ogni volta che viene diffuso verso un nuovo posto;
evaporationRate permette di specificare come varia l’ intensità del campo in base al tempo;

maxIntensity massimo valore d’ intensità raggiungibile da questo campo (serve per calcolare il colore attuale del campo in rapporto con l’ attuale intensità);

color colore del campo. Può essere ignorato se non si ha la necessità di visualizzare graficamente il campo;

properties permette di specificare delle proprietà aggiuntive (cioè è un elenco di nomi e di un valori). Non è necessario che questo parametro sia presente.

4.4.1 Diffusione

Per la diffusione dei campi nello spazio possono essere adottate differenti strategie, che dipendono dall’ uso che vogliamo fare dei campi e dal tipo di struttura spaziale su cui si diffondono. Si è quindi scelto di realizzare un sistema che permettesse differenti infrastrutture di diffusione, in modo che per ogni situazione possa essere scelto l’ approccio più appropriato.

L’infrastruttura di diffusione è il sistema che si occupa di diffondere i campi tra i posti che compongono lo spazio ed ad ogni spazio è possibile associare una differente politica di diffusione. Le politiche di diffusione sono definite estendendo la classe astratta AbstractDiffusion. Quando c’ è un campo da diffondere, viene chiamato il metodo diffuse, che vuole come parametri il posto da cui deve venire diffusio il campo e il campo stesso. Attraverso le differenti implementazioni di questo metodo, si possono definire diverse politiche di diffusione.

Figura 4.15: Le classi che compongono il package diffusion
4.4.1.1 Diffusione nulla

Attraverso la classe NullDiffusion è stata implementata una semplice politica di diffusione, in cui i campi non vengono diffusi, ma rimangono solo sul posto su cui sono stati emessi.

Questa classe è molto semplice: in pratica il metodo diffuse non fa assolutamente nulla. Questa classe può essere utilizzata con qualunque tipo di spazio.

4.4.1.2 Floodding corretto

La diffusione dei campi tramite flooding prevede che ogni posto, dopo aver ricevuto un campo, lo diffonda a tutti i posti vicini con l’ esclusione di quello da cui è arrivato.

Questo metodo di diffusione ha il vantaggio di non utilizzare alcuna struttura di supporto per l’ instradamento, ma è soggetto ad alcuni svantaggi, come la possibilità di avere dei campi duplicati presenti nello stesso posto, la non determinabilità del percorso di diffusione e la terminazione non garantita.

Alcuni di questi svantaggi possono essere eliminati con una serie di controlli che devono essere eseguiti prima di diffondere il campo. Questi controlli si basano sul fatto che ogni campo può essere individuato univocamente da un identificativo (FieldId) generato a partire dall’ identificativo del posto su cui è stato emesso il campo e da un numero progressivo di campi generati su quel posto.

Grazie a questo id è possibile determinare se un campo è già presente in un dato posto. In caso affermativo, viene eseguito un controllo sull’intensità: se è minore od uguale significa che il campo è giunto attraverso un percorso più lungo (o di uguale lunghezza) e quindi deve essere scartato, se è maggiore verrà invece inoltrato ai posti vicini (ad esclusione del posto da cui è arrivato).

Uno dei problemi di questo approccio è che lo stesso campo può giungere sullo stesso posto in momenti diversi e questo significa che si deve diffondere più volte lo stesso campo. Anche se ciò rimane un problema, generalmente non accade, dato i campi che percorrono il percorso più breve (e quindi hanno intensità maggiore) solitamente arrivano prima di quelli che compiono un percorso più lungo.

Il problema della non terminazione è risolto imponendo al campo un valore di attenuazione di diffusione maggiore di 0. In questo modo è garantito che il processo di diffusione termini in quanto o l’ intensità del campo raggiunge lo 0 (e quindi smette di diffondersi) o si diffonde su tutti i siti (ed anche in questo caso, termina).
La classe che implementa questo tipo di diffusione è la *FloadDiffusion* e può essere utilizzata per qualunque tipo di spazio.

### 4.4.1.3 Diffusione per spazi regolari

Per gli spazi regolari, è possibile definire delle politiche di diffusione che sfruttino la conoscenza a priori della struttura dello spazio. In questo modo è possibile, senza appoggiarsi a nessuna struttura esterna, definire in modo univoco per ogni posto e per ogni sorgente, verso quali posti diffondere i campi.

Un tipo di diffusione per spazi regolari che è stata implementata è la *VonNeumannDiffusion* che si occupa della diffusione di campi su griglie con l’ intorno di VonNeumann.

Negli spazi di questo tipo, ogni posto è identificato da due coordinate (x, y) ed è a partire dalla posizione della sorgente rispetto al nodo da cui si vuole diffondere un campo che si calcolano i posti verso cui verrà diffuso in campo. Le informazioni su cui l’ algoritmo si basa sono:

- coordinate del posto da cui si vuole diffondere il campo;
- coordinate del posto sorgente del campo;
- dimensioni dello spazio.

L’ algoritmo può essere così espresso in pseudocodice:

```plaintext
if (y == sourceY) && (x > 0) && (x <= sourceX)
    diffuse to site (x-1,y)
if (y == sourceY) && (x < spaceWidth-1) && (x >= sourceX)
    diffuse to site (x+1,y)
if (y < spaceHeight-1) && (y >= sourceY)
    diffuse to site (x,y+1)
if (y > 0) && (y <= sourceY)
    diffuse to site (x,y-1)
```

In pratica, tutti i posti che si trovano sulla stessa riga della sorgente (cioè che hanno la loro coordinat y uguale a quella della sorgente) diffondono il campo verso destra o verso sinistra a seconda di se si trovano a destra o a sinistra della sorgente (ad esempio, un posto che si trova a sinistra della sorgente diffonderà i campi verso il sito che si trova alla sua sinistra). Tutti i posti che hanno una y maggiore od uguale a quella della
sorgente diffondono anche verso il posto che hanno immediatamente sotto e i posti che hanno una \( y \) minore od uguale diffondono verso il posto che hanno sopra.

![Figura 4.16: Sequenza di diffusione data dall’ algoritmo per la diffusione in griglie con l’ intorno di Von Neumann](image)

L’ algoritmo può essere adattato anche per occuparsi della diffusione su griglie toroidali con l’ intorno di Von Neumann:

\[
\begin{align*}
\text{dx} &= \frac{w}{2} - \text{sourceX}; \\
\text{dy} &= \frac{h}{2} - \text{sourceY}; \\
\text{if} \ (y+dy == \text{sourceY}+dy) \ && (x+dx > 0) \ && (x+dx <= \text{sourceX}+dx) \\
&\quad \text{diffuse to site} \ ((x-1+\text{spaceWidth}) \ mod \ \text{spaceWidth}, \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (y+\text{spaceHeight}) \ mod \ \text{spaceHeight}) \\
\text{if} \ (y+dy == \text{sourceY}+dy) \ && (x+dx < \text{spaceWidth}-1) \ && (x+dx >= \text{sourceX}+dx) \\
&\quad \text{diffuse to site} \ ((x+1+\text{spaceWidth}) \ mod \ \text{spaceWidth}, \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (y+\text{spaceHeight}) \ mod \ \text{spaceHeight}) \\
\text{if} \ (y+dy < \text{spaceHeight}-1) \ && (y+dy >= \text{sourceY}+dy) \\
&\quad \text{diffuse to site} \ ((x+\text{spaceWidth}) \ mod \ \text{spaceWidth}, \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (y+1+\text{spaceHeight}) \ mod \ \text{spaceHeight}) \\
\text{if} \ (y+dy > 0) \ && (y+dy <= \text{sourceY}+dy) \\
&\quad \text{diffuse to site} \ ((x+\text{spaceWidth}) \ mod \ \text{spaceWidth}, \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (y-1+\text{spaceHeight}) \ mod \ \text{spaceHeight})
\end{align*}
\]

L’ algoritmo rimane sostanzialmente invariato, opera solamente come se la sorgente del campo fosse translata al centro dello spazio (sommando \( dx \) e \( dy \) a tutte le coordinate). Vengono effettuate anche ulteriori operazioni che hanno lo scopo di riportare i valori delle coordinate all’interno dell’intervallo consentito (cioè \([0, \text{spaceWidth}-1]\) per quanto riguarda la \( x \) e \([0, \text{spaceHeight}-1]\) per la \( y \)).

### 4.5 Strumenti

Oltre agli strumenti specifici per la visualizzazione degli spazi, dei campi e degli agenti, la piattaforma mette a disposizione anche un piccolo insieme di tool che sono utili per esaminare lo stato del sistema ed effettuare il tuning. Gli strumenti che sono attualmente presenti nella piattaforma sono i seguenti:
- Object Inspector;
- Console;
- Memory Monitor.

Questi strumenti sono integrati nell’ interfaccia grafica in modo da permettere il loro utilizzo attraverso pochi click del mouse.

4.5.1 Object Inspector

L’ Object Inspector è uno strumento che permette di visualizzare tutte le proprietà di un qualunque oggetto. Le proprietà di un oggetto Java vengono accedute attraverso i metodi *getXXX()* e *setXXX()*, dove *XXX* è il nome della proprietà.

Il funzionamento dell’ Inspector prevede che, una volta ottenuto l’ elenco di tutti i metodi *getXXX()* di un oggetto, questi vengano invocati uno per uno e vengono visualizzati i valori di ritorno.

La procedura scritta in Java risulta la seguente:

```java
BeanInfo bi = java.beans.Introspector.getBeanInfo(object.getClass());
// Ottiene l’elenco delle proprietà dell’oggetto
PropertyDescriptor[] pds = bi.getPropertyDescriptors();
for (int i = 0; i < pds.length; i++) {
    PropertyDescriptor pd = pds[i];
    // Ottiene il nome della proprietà
    String name = pd.getName();
    Method getMethod = pd.getReadMethod();
    // Invoca il metodo getXXX()
    Object o = getMethod.invoke(object, null);
    if (o != null) {
        // Visualizza il nome della proprietà ed il valore
        ...
    }
}
```

L’ utilizzo più comune che viene fatto dell’ inspector è la visualizzazione delle proprietà dei siti e degli agenti. Se viene utilizzata *GridSpace* o *GraphSpace* per visualizzare
uno spazio, è possibile cliccare su di un sito per visualizzare le sue proprietà, mentre si può visualizzare quelle di un agente a partire dall’ elenco degli agenti del sistema.

![Inspector](image1)


### 4.5.2 Console

Questo strumento intercetta tutto ciò che viene scritto sullo *standard output* e sullo *standard error* e lo visualizza all’interno di una finestra grafica. In questo modo è possibile visualizzare tutti i messaggi scritti dall’applicazione o dagli agenti tramite *System.out.println* o *System.err.println* direttamente all’interno della finestra dell’applicazione, senza dover andare a vedere la console.

L’utilizzo di questi stream può risultare utile nella fase di debug di una applicazione per verificare che tutto funzioni a dovere. L’implementazione della console deriva da un’estensione della *Simple Java Console* di RJHM Van Den Bergh.

![Console](image2)

Figura 4.18: Screenshot della Console.
4.5.3 Memory Monitor

Si tratta di un semplice tool che visualizza un monitor che permette di tenere traccia della quantità di memoria utilizzata. In questo modo si può verificare, ad esempio, che l’occupazione di memoria non 'esploda' con il passare del tempo (anche grazie alla visualizzazione sotto forma di grafico).

![Figure 4.19: Screenshot del Memory Monitor; la barra verticale a sinistra indica la memoria attualmente utilizzata, mentre il grafico a destra mostra l’andamento nel tempo](image)

4.6 Esempio di utilizzo

In questo paragrafo verrà descritta l’implementazione di una semplice applicazione di prova della piattaforma. Questo esempio può servire come base per realizzare delle applicazioni reali basate.

L’applicazione di esempio ha uno spazio regolare con l’intorno di Von Neumann e 3 agenti: un agente di tipo A immobile che emette un campo e due agenti di tipo B che si muovono alla ricerca dell’intensità massima del campo emesso da A.

L’esempio è costituito da 3 classi: AgentA, AgentB e Test.

Le prime due sono le classi che determinano il comportamento degli agenti, mentre la terza si occupa dell’inizializzazione dello spazio e degli agenti.

Il primo agente (AgentA) è immobile e genera, al momento della sua inizializzazione, un campo di tipo A e di colore rosso con una data intensità, secondo un parametro dato.

```java
public class AgentA extends SingleLayerAgent {
    private int intensity;

    public AgentA(String name, Site site, int intensity) {
        super(name, site);
        this.intensity=intensity;
    }
```
Il secondo agente (AgentB) si muove verso i posti in cui l’ intensità di campo di tipo A è maggiore. Nel caso l’ intensità del campo sia uguale su due posti diversi, ne viene scelto a caso uno dei due. Anche nel caso in cui l’ intensità del campo sia nulla per tutti i posti vicini ne viene scelto uno a caso verso cui muoversi.

Dopo essersi mosso, l’ agente emette un campo che evapora di tipo B.

La classe principale dell’ applicazione (che estende mmass.Main) si occupa di inizializzare lo spazio e gli agenti e di creare una finestra grafica per la loro visualizzazione.

Lo spazio è una griglia regolare con l’ intorno di Von Neumann di dimensione 15 x 15 e gli agenti vengono posizionati a caso all’ interno di questo spazio.

La visualizzazione è effettuata tramite la classe GridView, a cui vengono impostati dei parametri che riguardano la dimensione e la distanza dei nodi.
addAgent(new AgentB("B1", space.randomSite()));
addAgent(new AgentB("B2", space.randomSite()));

// Visualizzazione
GridView view = new GridView(mainFrame, space, 200, 200);
view.setNodeDistance(8);
view.getDrawNode().setNodeRadius(4);
view.getDrawAgent().setAgentRadius(2);
mainFrame.addSpaceWindow(view, 50, 50, 200, 200);
}

public static void main(String[] argv) {
    new Test();
}
}

Se si desidera visualizzare lo spazio come una griglia isometrica, è sufficiente utilizzare le seguenti linee di codice (le due differenti visualizzazioni possono anche tranquillamente coesistere perché per ogni spazio è possibile avere zero, una o più viste):

IsoGridView isoView = new IsoGridView(mainFrame, space);
mainFrame.addSpaceWindow(isoView, 350, 50, 600, 480);

Figura 4.20: Due finestre di visualizzazione dello spazio dell’ applicazione di prova
Capitolo 5

WebSpace

In questo capitolo verrà descritta WebSpace, un’ applicazione che sfrutta la modellazione del Web come un sistema ad agenti e la piattaforma realizzata per fornire contenuti adattivi agli utilizzatori di un sito Web e per fornire supporto ai Web Master.

5.1 Descrizione dell’ applicazione

WebSpace è un software, operante a lato Server, che si occupa di monitorare la navigazione degli utenti su di un sito Web e di utilizzare le informazioni raccolte sia per fornire delle informazioni utili al Web Master che per generare contenuti adattivi.

La generazione dei contenuti adattivi si limita al suggerimento, attraverso un box inserito all’ interno delle pagine del sito, di una serie di link a pagine a cui gli utenti potrebbero essere interessati, scelti perché altri utenti si sono dimostrati interessati a quelle pagine.

I dati raccolti per generare i contenuti adattivi sono gli stessi che vengono utilizzati per generare le statistiche per l’ amministratore del sito Web. Anche in questo caso si tratta di uno strumento di supporto, solo che qui lo scopo è mettere in evidenza del modo più chiaro possibile delle informazione e lasciare che sia una persona ad analizzarle e a prendere delle decisioni.

Il programma opera utilizzando una rappresentazione della struttura del sito Web come un grafo, in ad ogni pagina del sito è associato un nodo del grafo e ad ogni link un collegamento tra due siti. Questo spazio, come verrà descritto con maggior dettaglio
in seguito, può essere creato tutto in una volta al momento dell’ inizializzazione del programma o dinamicamente durante la sua esecuzione, a seconda della strategia più conveniente secondo la tipologia di sito Web su cui l’ applicazione è utilizzata.

Ad ogni utente che si collega al sito Web, l’ applicazione associa un agente valido per tutta la sua sessione di navigazione. Questo agente viene creato la prima volta che l’ utente si collega (la prima volta di ogni sessione di collegamento, non solamente la prima volta in assoluto) e l’ associazione avviene tramite una mappa tra l’ identificativo della sessione e l’ agente stesso. Ad ogni richiesta di pagina da parte dell’ utente, il suo agente viene spostato sul nodo corrispondente alla pagina richiesta. Se per un certo periodo non vengono ricevuti più messaggi relativi ad un dato utente, la sua sessione scade ed l’ agente viene rimosso.

**Figura 5.1: Schema di funzionamento di WebSpace alla ricezione di una richiesta di pagina da parte di un utente**

### 5.2 Creazione della struttura spaziale

Nella fase di progettazione dell’ applicazione, sono stati analizzati differenti modalità per la generazione della struttura che rappresenta un sito Web.

L’ approccio più semplice consiste nel creare “a mano” la struttura spaziale, inserendo, tramite l’ editor dei grafì, tutte le pagine ed i link. Questo sistema è applicabile solo a siti formati da poche pagine e che vengono modificati raramente, perché altrimenti diventa difficile mantenere una rappresentazione accurata e aggiornata.
Figura 5.2: Struttura di un sito Web disegnata tramite GraphEditor

Per un utilizzo reale del programma è necessario che la struttura spaziale sia generata automaticamente, in modo da non richiedere nessun intervento ogni volta che viene effettuata una modifica del sito.

Nello sviluppo dell’ applicazione sono state individuate differenti procedure per acquisire la struttura; quelle considerate più promettenti sono le seguenti:

- creazione della struttura basandosi su dati contenuti in un database o comunque memorizzati in modo esplicito;
- acquisizione della struttura del sito durante l’ inizializzazione dell’ applicazione tramite un processo di crawling;
- generazione dinamica della struttura spaziale secondo la navigazione degli utenti;

Nei paragrafi successivi verranno esaminati in dettaglio i vari approcci.

### 5.2.1 Creazione da struttura nota

Per i siti Web basati su CMS (Content Management System) è abbastanza semplice generare la struttura spaziale sfruttando i dati memorizzati nel CMS stesso.

I CMS sono sistemi usati per gestire dinamicamente i contenuti dei siti Web che semplificano l’ inserimento e la modifica dei contenuti.
Tabella 5.1: Confronto tra i diversi approcci di generazione dello spazio

<table>
<thead>
<tr>
<th></th>
<th>Creazione da struttura nota</th>
<th>Acquisizione</th>
<th>Generazione a runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contiene tutti le pagine raggiungibili del sito</td>
<td>No</td>
<td>Sì</td>
<td>No, solo le visitate</td>
</tr>
<tr>
<td>Permette di conoscere l’intera struttura del sito</td>
<td>Sì</td>
<td>Sì</td>
<td>No</td>
</tr>
<tr>
<td>Permette di non visualizzare pagine di servizio</td>
<td>Sì</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Viene modificato dinamicamente</td>
<td>No</td>
<td>No</td>
<td>Sì</td>
</tr>
<tr>
<td>Approccio adatto ad ogni tipo di sito</td>
<td>No</td>
<td>Sì</td>
<td>Sì</td>
</tr>
<tr>
<td>Tempo di creazione dello spazio</td>
<td>Basso</td>
<td>Dipende dal numero di pagine</td>
<td>Basso</td>
</tr>
</tbody>
</table>

La caratteristiche dei sistemi CMS variano da prodotto a prodotto, ma la maggior parte includono un sistema di pubblicazione basato su Web, la gestione delle versioni dei documenti, l’ indicizzazione e la ricerca. Il sistema di pubblicazione basato sul Web permette a chi si occupa dell’inserimento dei contenuti di usare dei modelli, dei *wizard* ed altri strumenti per creare e modificare i contenuti.

Tipicamente, un CMS consiste di tre elementi: il sistema di gestione dei contenuti CMA (Content Management Application), il motore di pubblicazione delle pagine CDA (Content Delivery Application) e la base dati contenente i contenuti.

Il database è l’archivio dei contenuti che saranno pubblicati sul sito e normalmente contiene la struttura della parte dinamica del sito, i contenuti di tutte le pagine e i link tra le pagine.

Il database contiene tutte le informazioni necessarie per la generazione della struttura spaziale, si tratta solamente di estrarle e di generare lo spazio. Il problema è che il suo formato varia secondo il CMS usato, per cui non è possibile realizzare un modulo universale che acceda al database e generi la struttura spaziale.

Per ovviare a questo inconveniente si è definito un formato XML con cui può essere descritta la struttura di un sito. In questo modo, per interfacciare un CMS con WebSpace, è sufficiente estrarre i dati presenti nel database e convertirli in questo formato.

Il formato XML scelto non è altro che un’estensione del formato utilizzato dalla piattaforma per gestire gli spazi, che è descritto in 4.2.3.3.
5.2.2 Acquisizione della struttura di un sito Web

Lo spazio che rappresenta la struttura di un sito Web può essere generato come risultato di un processo di *crawling*\(^1\), cioè di un processo automatizzato in cui viene scaricata una pagina Web, vengono seguiti i link per individuare nuove pagine da scaricare e viene ripetuto fino a quando ci sono pagine da acquisire.

Il *crawling* è l’ approccio seguito dai motori di ricerca per l’ acquisizione delle pagine da indicizzare, ma, a differenza dei motori di ricerca, in questo caso vengono esaminati solo i link interni, cioè quelli che puntano verso pagine dello stesso sito Web.

Per la realizzazione dell’ applicazione si è scelto di utilizzare, come *Crawler* per acquisire le pagine, WebSPHINX\(^2\). Si tratta di un *Web Crawler*, scritto in Java, realizzato presso la Carnegie Mellon University che presenta le seguenti caratteristiche:

- sistema multithreaded di acquisizione delle pagine;
- modello ad oggetti in cui vengono rappresentati esplicitamente pagine e link;
- sistema di classificazione delle pagine;
- parser HTML tollerante agli errori;
- supporto per il *Robot Exclusion Protocol*;
- pattern matching con espressioni regolari, *wildcards*;
- modifica contenuto pagine HTML (concatenamento di pagine, salvataggio, rinominazione link).

---

\(^1\) Maggiori informazioni sul *crawling* possono essere trovate su [Ferguson 2002](http://example.com/ferguson2002) [Heaton 2002](http://example.com/heaton2002).

Per utilizzare WebSPHINX è stato necessario implementare una classe che estende \texttt{websphinx.Crawler} e si occupa di popolare lo spazio con le informazioni acquisite tramite il \texttt{Crawler}:

```java
public class MMASSSpider extends Crawler {
    /**
     * Callback for testing whether a link should be traversed.
     * @param l Link encountered by the crawler
     * @return true if link should be followed,
     * false if it should be ignored.
     */
    public boolean shouldVisit(Link l) {
        return l.getHost().equalsIgnoreCase(host);
    }

    /**
     * Callback for visiting a page.
     * @param page Page retrieved by the crawler
     */
    public void visit (Page page) {
        if (page.isHTML()) {
            WebSpaceSite site=webSpace.addSite(page.getURL());
            site.setProperty("title",page.getTitle());
            // Links
            websphinx.Link links[]=page.getLinks();
            for (int i=0; i<links.length; i++)
                tempLinks.add(new LinkFrom(page,links[i]));
        }
    }

    /**
     * Run
     */
    public void run() {
        super.run();
        // Links
        Iterator it=tempLinks.iterator();
        while (it.hasNext()) {
            LinkFrom linkFrom=(LinkFrom)it.next();
            if (linkFrom != null)
                linkSites(linkFrom.from,linkFrom.to);
        }
    }
}
```
CAPITOLO 5. WEBSPACE

Grazie all’utilizzo del Robot Exclusion Protocol è possibile specificare delle parti del sito che verranno ignorate da WebSPHINX. Per utilizzare il Robot Exclusion Protocol è necessario creare un file chiamato robot.txt nella directory principale del sito. Questo file ha una struttura simile al seguente:

```
User-agent: *
Disallow: /jsp
Disallow: /tmp
```

In questo esempio si specifica che tutti i Robots non devono tener conto dei documenti che si trovano nei percorsi /jsp e /tmp. In questo modo è possibile, ad esempio, ignorare tutte le pagine di servizio.

5.2.3 Generazione dinamica dello spazio secondo la navigazione degli utenti

Un differente approccio per la generazione della struttura spaziale prevede che lo spazio delle pagine venga generato man mano che gli utenti esplorano un sito Web.

In prima analisi si potrebbe pensare che la procedura di generazione dello spazio possa essere la seguente: ogni volta che un utente richiede una determinata pagina, si controlla se è già presente nello spazio e, in caso non sia presente, viene aggiunta.

Purtroppo questo approccio non è praticabile, perché pone dei problemi con i collegamenti tra le pagine nello spazio. Per capire qual’è il problema basti pensare che, quando lo spazio è ancora vuoto, un utente richieda una pagina; a questo punto si dovrebbe creare un posto nello spazio relativo a questa pagina e creare tutti i collegamenti che rappresentano i link presenti nella pagina. Il problema è che le pagine di destinazione non saranno presenti e quindi non potranno essere creati i collegamenti. Anche se man mano lo spazio andrebbe a popolarsi di pagine, ci sarebbero sempre molti collegamenti mancanti.

Per ovviare a questo inconveniente, si è pensato di avere due differenti tipologie di posti nello spazio: i posti normali e i posti non visitati. I primi sono i posti relativi alle pagine richieste dagli utenti, mentre i secondi sono relativi a pagine non visitate ma sono raggiunti da qualche collegamento proveniente da pagine visitate. Per ogni posto relativo ad una pagina vista sono presenti tutti i collegamenti che partono da lì, mentre per gli altri posti sono presenti solo i collegamenti che arrivano da pagine già visitate.

3http://www.robotstxt.org/wc/norobots-rfc.html
Quando un utente richiede una pagina che non è già presente nello spazio, si procede nel seguente modo:

- viene creato nello spazio un posto corrispondente alla pagina richiesta;
- per ogni destinazione dei link presenti nella pagina, se non è già presente, viene creato un posto che viene marcato come non-visitato;
- vengono creati dei collegamenti tra il posto ed i posti relativi alle destinazioni dei link.

Se invece viene richiesta una pagina il cui posto è già presente nello spazio ma è marcato come non-visitato, si procede così:

- se non è già presente, viene creato un posto marcato come non-visitato per ogni destinazione di link presente all’interno della pagina;
- se non sono già presenti, vengono creati i collegamenti tra il posto ed i posti relativi alle destinazioni dei link;
- il posto corrispondente alla pagine viene marcato come visitato;

Un esempio di 3 momenti della generazione di uno spazio è mostrato nella figura 5.4.

![Figura 5.4: Esempio di generazione dello spazio secondo la navigazione degli utenti](image)

La prima figura si riferisce ad un sito Web non ancora esplorato, in cui la struttura spaziale non contiene nessun posto.

Nella seconda figura un agente entra nel posto A, che viene creato nello spazio. Oltre al posto A, vengono aggiunti anche i posti a lui vicini ed i collegamenti verso di essi. I posti vicini (rappresentati in giallo) sono già presenti nello spazio, ma non sono state generate tutte le informazioni che li riguardano (cioè tutti i collegamenti in uscita da essi).
L’ultima figura mostra l’ agente che si è spostato sul posto C che a questo punto viene integrato delle informazioni mancanti e viene contrassegnato come acquisito. Tutti i posti vicini vengono acquisiti e vengono generati i collegamenti verso di loro.

5.3 Raccolta di informazioni: le tracce degli utenti

Generalmente, le applicazioni che si occupano di analizzare il comportamente degli utenti di un sito Web, si basano sull’ analisi dei file di log, che sono dei file di cui viene memorizzata dai Web Server ogni richiesta proveniente dagli utenti.

Il formato di questi file è stato standardizzato con il nome di Common Logfile Formato (CLF) e prevede che in ogni riga del file sia presente una sola richiesta, rappresentata tramite la seguente struttura:

```
remotehost rfc931 authuser [date] "request" status bytes
```

- `remotehost` nome o indirizzo IP del client;
- `rfc931` nome dell’ utente sulla macchina che ha effettuato la richiesta (ottenuto tramite `identd`);
- `authuser` nome con cui l’ utente si è autenticato sul server;
- `[date]` data e orario della richiesta (tra parentesi quadre);
- "request" richiesta del client (tra virgolette);
- `status` codice di stato HTTP;
- `content-length` lunghezza in bytes del documento inviato.

Un esempio di una riga di un file di log di un Web Server, è il seguente:

```
127.0.0.1 - - [01/Sep/2004:19:48:17 +0200] "GET /images/macosxlogo.gif HTTP/1.1" 200 2829
```

Per indicare che alcune informazioni non sono disponibili viene utilizzato il carattere ‘.’. In questo caso mancano il nome dell’ utente sulla macchina remota e il nome

4http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format
CAPITOLO 5. WEBSpace

dell’ utente autenticato. Questa richiesta è stata fatta il 1 Settembre 2004 alle 19:48, è relativa al file /images/macosxlogo.gif e ha avuto esito positivo (status 200).

Il file di log può raggiungere dimensioni molto considerevoli (i siti più frequentati contano anche milioni di accessi al mese).

Esistono pertanto degli appositi programmi che hanno lo scopo di analizzare i log file per sintetizzarne il contenuto in una forma più utile. Questi programmi, che devono essere installati sul Web Server che ospita il sito, sono chiamati Web Analyzer [Polillo 2004].

Questi programmi vengono generalmente eseguiti ad intervalli regolari, ad esempio ogni giorno, e producono un’ insieme di dati aggregati aggiornati al momento della loro esecuzione.

Figura 5.5: Analisi dei file di log di un Web Server

Purtroppo questo approccio non è funzionale all’ applicazione che si sta realizzando, in quanto si desidera avere dati aggiornati ’in tempo reale’, che sono necessari, ad esempio, per visualizzare se ci sono altri utenti presenti nel momento in cui si consulta il sito.

Un ulteriore problema di difficile soluzione è tener traccia degli utenti, dato che nei log vengono riportati solo i nomi degli utenti remoti (e questa informazione non è disponibile praticamente mai) ed i nomi degli utenti autenticati. Non c’è un modo di riconoscere degli utenti non autenticati, se non dall’ indirizzo IP, ma questo sistema non è molto affidabile, in quanto alcune tipologie di rete fanno in modo che le richieste di un singolo utente arrivino con indirizzi IP differenti o che più utenti accedano con un singolo indirizzo IP.

6Un esempio di programma per l’ analisi dei file di log è Webalizer, un software Open Source che può essere trovato sul sito http://www.mrunix.net/webalizer/
Per risolvere questi problemi, si è deciso di provare altri approcci per tenere traccia dell’attività degli utenti. Le modalità che sono state analizzate e ritenute valide e applicabili sono le seguenti:

- creazione di un *Proxy* tra l’utente e il *Web Server* che intercetti ogni richiesta e ne traccia;
- realizzazione di un filtro per un Web Server in modo che comunichi ogni accesso che viene effettuato;
- inserimento di un *tracker* nelle pagine da monitorare.

Nei paragrafi successivi queste tecniche verranno analizzate in dettaglio.

### 5.3.1 Proxy

L’analisi dell’attività degli utenti tramite un *Proxy* prevede la presenza di un componente (il *Proxy*, appunto) che filtra ogni richiesta che viene inviata al *Web Server*.

Rispetto all’analisi dei log, questo approccio è più versatile, in quanto permette di tracciare ogni singolo utente (tramite dei cookie) e fornisce informazioni “in tempo reale” su ciò che stanno facendo i vari utenti.

Il *Proxy* opera nel seguente modo:

1. Riceve una richiesta da un utente utente.
2. Analizza la richiesta per determinare l’identità dell’utente.
3. Invia la richiesta al *Web Server*.
4. Riceve la risposta del web server.
5. Invia la risposta ricevuta al client.

La realizzazione di un semplice *Proxy* non è particolarmente complicata (le classi standard di Java prevedono quasi tutte queste funzionalità), ma può presentare dei seri problemi per quanto riguarda le *performance*, dato ogni richiesta e ogni risposta che viene inviata o proveniente dal Web Server, deve passare per questo componente, che rischia di diventare un collo di bottiglia.
Un ulteriore problema è che se, per un qualunque motivo, il Proxy smettesse di funzionare, il sito Web non sarebbe più accessibile.

![Figura 5.6: Architettura del proxy](image.png)

### 5.3.2 Filtro Web Server

Un’ altra possibile strada per monitorare l’ attività degli utenti durante la navigazione di un sito, è programmare il Web Server in modo che generi un messaggio da inviare all’ applicazione ogni volta che un utente richiede una pagina. Come ciò possa essere fatto, dipende dal Web Server utilizzato e potrebbe essere difficile da implementare su alcuni server.

Per i Web Server compatibili con le specifiche Java Servlet Container (come Tomcat\(^7\), Jetty\(^8\), JRun\(^9\), Resin\(^10\)) non è molto difficile generare un messaggio ogni volta che un utente richiede una pagina, basta definire un filtro.

I filtri vengono realizzati implementando l’ interfaccia `javax.servlet.Filter`\(^11\). L’ implementazione del filtro è molto semplice ed è simile alla seguente:

```java
public class MonitorFilter implements Filter {
    public void doFilter(ServletRequest request,
                         ServletResponse response, FilterChain chain)
                throws IOException, ServletException {
        // Invia la richiesta dell’utente all’applicazione
        ...
        // Procede con l’invio della pagina al client
        chain.doFilter(request, response);
    }
}
```

\(^7\)[http://jakarta.apache.org/tomcat/](http://jakarta.apache.org/tomcat/)
\(^8\)[http://mortbay.org/jetty/](http://mortbay.org/jetty/)
\(^11\)[http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/Filter.html](http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/Filter.html)
Per utilizzare il filtro, è necessario inserire le seguenti linee di configurazione nel file WEB-INF/web.xml della Web Application:

```xml
<filter>
    <filter-name>monitor</filter-name>
    <filter-class>webspace.MonitorFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>monitor</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

A differenza di un Proxy separato dal Web Server, questo approccio non ha influenze negative sulle prestazioni del Web Server, ma è maggiormente dipendente da uno specifico tipo di Web Server.

![Diagram of filter configuration](image)

Figura 5.7: Esempio di filtri come descritti nella specifica Java Servlet

### 5.3.3 Tracker

Con il termine Tracker si intende una pagina dinamica che viene richiamata da un' altra e serve a monitorare gli accessi che vengono fatti. In pratica, ogni volta che viene caricata la pagina principale, viene richiamato anche il Tracker. Questo componente viene visualizzato come un' immagine con il numero di persone che hanno visto la pagina o, se è più “discreto”, come una piccola immagine invisibile. In genere, nelle pagine principali dei siti viene riportato il numero dei visitatori, mentre nelle altre pagine, è presente solo sotto forma di immagine invisibile, solo per poter contare il numero di accessi.
CAPITOLO 5. WEBSpace

Per poter identificare gli utenti e tenere traccia delle pagine richieste, viene fatto uso delle informazioni presenti nell’ header del protocollo HTTP12.

Attraverso l’intestazione Referer, presente nell’header, il Tracker è in grado di determinare a quale pagina il contatore si deve riferire e grazie all’utilizzo dei cookie è possibile tracciare ogni singolo utente. Un esempio di risposta di un Web Server alla richiesta di una pagina da parte di un client, è simile al seguente:

HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=A15888227AF642FA1135075D27AE8CEB; Path=/
Date: Sun, 12 Dec 2004 21:23:21 GMT
Server: Apache-Coyote/1.1
Referer: http://localhost/index.html
Connection: close

In questo esempio il server invia al client il cookie JSESSIONID=A15888227AF642FA1135075D27AE8CEB, che il client includerà in ogni futura richiesta che farà a questo Web Server e che permetterà al server di tracciarlo, mentre il parametro Referer indica che l’URL da cui è stata effettuata la richiesta di questo documento è http://localhost/index.html.

L’utilizzo di un tracker è abbastanza flessibile, in quanto è possibile decidere quali pagine vogliamo che vengono monitorate (per esempio, lasciando senza contatori le pagine di servizio). Un altro vantaggio è che può fisicamente risiedere su un sito diverso rispetto a quello che si vuole monitorare.

Nell’applicazione è stato implementato un tracker attraverso l’utilizzo della tecnologia Java Servlet.

Uno degli aspetti che si sono dovuti tenere in considerazione è la generazione dell’immagine da inviare all’utente, perché questo, su siti molto visitati, può essere un collo di bottiglia, in quanto tutti gli utenti richiamano molte volte il tracker.

Per velocizzare la generazione dell’immagine da inviare all’utente, al posto di utilizzare le classi Java per creare l’immagine, o di caricarla ogni volta da un file, si è preferito crearla una sola volta utilizzando un programma di grafica e copiare all’interno del codice Java i byte che compongono il file.

Si è realizzata una immagine GIF (Graphics Interchange Format) di dimensioni 1 x 1 vuota con lo sfondo trasparente, in modo che non venga visualizzato nulla dall’utente qualunque sia il colore di sfondo della pagina. E’ stato scelto GIF come formato dell’

12Le informazioni sul protocollo HTTP 1.1 e sul suo header si possono trovare nel documento RFC 2616 - http://www.ietf.org/rfc/rfc2616.txt
immagine perché è supportato da tutti i Browser Web. L’ immagine memorizzata è stata successivamente trasformata nel suo equivalente esadecimale utilizzando il comando `hexdump`:

```
andreax@andreaxmac:~$ hexdump -C empty.gif
00000000  47 49 46 38 39 61 01 00  01 00 80 00 00 ff ff  |GIF89a...........
00000010  00 00 00 21 f9 04 01 00  00 00 00 2c 00 00 00  |....!.............
00000020  01 00 01 00 00 02 02 44  01 00 3b   ........D.;|
0000002b
```

L’ output del programma è stato copiato in un array di 43 byte che viene inviato come risposta ogni volta che l’ utente richiama il contatore. In questo modo l’ output generato sarà considerato dal browser come un’ immagine.

```java
private char[] emptyGif = {
    'G','I','F','8','9','a',0x01,0x00,
    0x01,0x00,0x80,0x00,0x0f,0x0f,0xff,0xff,
    0x00,0x00,0x00,0x21,0xf9,0x04,0x01,0x00,
    0x00,0x00,0x00,0x02,0x02,0x44,0x01,0x03b
};
```

Per assicurarsi che l’ immagine non venga memorizzata nella cache del browser dell’ utente (e che quindi il contatore venga richiamato meno spesso del dovuto, falsando le misurazioni) sono state aggiunte delle opportune istruzioni.

Nel codice HTML delle pagine che verranno monitorate, è necessario inserire un tag che richiami il `tracker`, come il seguente:

```html
<img src="http://server/counter" alt="" />
```

Figura 5.8: Esempio di funzionamento del `tracker`
5.4 Generazione dei contenuti adattivi

Progettare un sito Web complesso che permetta di accedere velocemente alle informazioni contenute è un compito difficile. Chi lo progetta deve anticipare i bisogni degli utenti e, in base a questi, definire la struttura del sito. Gli utenti possono avere molte viste differenti sulle informazioni del sito e queste possono cambiare nel tempo, in modi che il progettista non si aspettava. [Perkowitz Etzioni 2002]

Capire i bisogni degli utenti significa capire come gli utenti accedono ai dati disponibili e per poter fare questo è necessario studiare come gli utenti si muovono all’interno del sito. Una prima possibilità può essere prendere un gruppo di persone e farle navigare sul sito e analizzando i loro spostamenti e le loro reazioni. Successivamente si procede adattando il sito Web secondo le informazioni ricavate. Questo approccio non è sempre applicabile e, per siti Web di grandi dimensioni, diventa lungo e costoso effettuare questi test. Un’altro problema è che la procedura comporta un tempo per sistemare manualmente il sito, rendendo scarsamente reattiva la struttura al rapido variare delle richieste degli utenti.

Un approccio differente consiste nel realizzare siti Web che abbiamo la capacità di adattarsi secondo la navigazione degli utenti, ad esempio, mettendo automaticamente in evidenza le pagine che sono maggiormente visitate.

Ci sono varie tipologie di siti Web adattivi che si caratterizzano per le differenti modalità utilizzate per personalizzare i contenuti. La personalizzazione delle informazioni, ad esempio, può essere fatta per ogni singolo utente (cioè fornendo ad ogni utente una personalizzazione ad-hoc) oppure per tutti gli utenti del sito; questi due approcci spesso convivono: alcuni contenuti vengono personalizzati per ogni singolo utente mentre altri sono generati uguali per tutti gli utenti di un sito.

Un altro aspetto riguarda quali contenuti può essere personalizzato; in genere vengono proposti dei link, oppure vengono evidenziati dei testi o cambiato il formato a delle pagine. Proporre dei link è usato spesso per evidenziare dei contenuti a cui l’utente o altri utenti accedono frequentemente.
CAPITOLO 5. WEBSPACE

Figura 5.9: Nel sito di aste www.ebay.com viene presentato nella prima pagina l’ elenco degli ultimi prodotti visti dall’utente.

Un altro importante aspetto a cui prestare attenzione è la scelta dei dati su cui basarsi per la generazione dei contenuti adattivi. Le due alternative (anche se una non esclude necessariamente l’altra) sono basarsi sul contenuto delle pagine o sulla navigazione degli utenti. Molti siti di e-commerce mostrano come informazioni adattive gli ultimi prodotti visti. In questo caso l’adattamento del sito è basato sia sulla navigazione degli utenti che sul contenuto delle pagine.

Figura 5.10: Nel sito di e-commerce www.chl.it sono presenti dei box in cui vengono proposti i link verso le pagine dei prodotti più venduti e di quelli maggiormente visti dagli utenti.

Utilizzare il contenuto delle pagine spesso non è facile, perché questo è espresso in un linguaggio difficilmente interpretabile in modo automatico senza supervisione da parte di un utente, mentre diventa un approccio più pratico se ai documenti sono associate delle keyword che ne specificano il contenuto. L’altro approccio prevede l’analisi della navigazione degli utenti, al fine di determinare interessi comuni a più utenti e sfruttare queste informazioni per la generazione dei contenuti adattivi. L’idea è che se un dato
capitolo 5. webspacE

contenuto è risultato interessante per un gruppo di utenti, questo può essere proposto agli altri utenti perché anche loro potrebbero ritenerlo interessante. Questo secondo approccio è quello utilizzato da WebSpace per la generazione dei suggerimenti.

5.4.1 Contenuti adattivi di WebSpace

WebSpace permette di ottimizzare un sito Web suggerendo ai visitatori dei link a contenuti a cui potrebbero essere interessati. Vengono proposti dei link uguali per tutti gli utenti del sito (in questo caso si tratta quindi di ottimizzazione della struttura del sito) che variano da pagina a pagina. In ogni pagina vengono proposti dei link verso le pagine vicine maggiormente visitate. Per pagine vicine si intendono quelle raggiungibili attraverso non più di due link, cioè la pagina A è vicina alla pagina B se c’è un link che le collega oppure se la pagina A è collegata alla pagina C che a sua volta è collegata alla pagina B.

Nella figura 5.11 viene mostrato un esempio delle pagine che vengono selezionate per essere proposte come contenuto adattivo a partire dalla pagina corrente. Per ogni pagina del sito che viene visualizzata, WebSpace genera un elenco delle pagine vicine maggiormente viste, le ordina secondo il numero di visite e mostra i link verso le prime sei pagine.

Le pagine proposte possono variare nel tempo, ad esempio, se molti utenti cliccano uno dei link consigliati, il link assumerà presto maggiore importanza (perché il contatore relativo al numero di utenti che hanno visto la pagina aumenterà) ed il link verrà proposto prima degli altri.

Figura 5.11: Parte della struttura di un sito Web, con il numero di visite per ogni pagina. Vengono evidenziate con il colore giallo le pagine suggerite a partire dalla pagina A.

Il lavoro svolto per proporre contenuti adattivi può essere considerato come una base su cui sviluppare futuri sistemi per realizzare siti adattivi. In questo caso lo scopo era
solo mostrare come poteva essere utilizzata la modellazione del Web come sistema ad agenti proposto nei capitoli precedenti per fornire contenuti adattivi in un sito Web.

## 5.5 Funzionalità di monitoraggio

Oltre che per generare contenuti adattivi per gli utenti, i dati raccolti da WebSpace vengono utilizzati per fornire un supporto a chi si occupa della gestione del sito, fornendo informazioni su come gli utenti utilizzano il sito.

Per determinare le funzionalità necessarie sono state analizzate alcune applicazioni di monitoraggio di siti Web esistenti e successivamente si è verificato che WebSpace fosse in grado di fornire almeno una parte di queste informazioni (almeno quelle maggiormente significative).

![Figura 5.12: Interfaccia Web di BBClone](image)

I software che sono stati analizzati per determinare le funzionalità richieste sono Webalizer e BBClone. Webalizer è uno strumento che permette di monitorare l’ andamento di un sito Web attraverso l’ analisi dei dati registrati dal Web Server. BBClone fornisce informazioni simili ma si basa su un approccio differente: prevede che vengano modificate le pagine del sito di cui si vogliono collezionare informazioni statistiche inserendo dei tag che richiamano un contatore. Ogni volta che viene richiamato il contatore da

---


14. [http://bbclone.de](http://bbclone.de)
CAPITOLO 5. WEBSPACE

una pagina, BBClone registra sul suo database interno le informazioni relative alla richiesta; in questo modo è possibile decidere di monitorare solo alcune pagine del sito o decidere di richiamare contatori differenti secondo lo stato dell’utente. Tutti i programmi analizzati dispongono di un’interfaccia Web in cui vengono presentate le informazioni statistiche sui dati raccolti.

Indipendentemente dal modo in cui le informazioni relative alle richieste vengano collezionate, i principali dati relativi ad ogni richiesta che vengono presi in considerazione dai programmi di monitoraggio sono i seguenti:

- indirizzo IP del client;
- pagina richiesta;
- referring URI\(^{15}\);
- data e ora.

Oltre a queste informazioni possono eventualmente esserne registrate delle altre, come il nome e la versione del sistema operativo del client, il browser utilizzato, la dimensione dello schermo, le lingue preferite, il supporto di Java, Flash o altri formati.

Altre informazioni che vengono presentate sono ricavate da questi dati: ad esempio, tramite l’indirizzo IP è possibile determinare la nazione da cui avviene il collegamento o analizzando tutte le connessioni di un singolo utente è possibile stabilire il suo tempo di permanenza su un sito Web.

Una caratteristica importante di questi programmi è che non si limitano a fornire queste informazioni per ogni richiesta o per ogni sessione, ma generano dei dati aggregati che risultano essere molto più significativi, perché racchiudono in una serie di numeri o in un grafico le informazioni relative non ad un singolo utente ma a un vasto numero di utenti.

\(^{15}\)Il referring URL è (l’eventuale) indirizzo da cui è stata richiamata una pagina o, più in generale, un qualunque contenuto di un sito Web. Ad esempio, per una immagine contenuta in una pagina, il referring URL è l’indirizzo della pagina che contiene l’immagine.
CAPITOLO 5. WEBSpace

5.6 Visualizzazione della struttura spaziale

Un altro aspetto che è stato preso in considerazione nella realizzazione di WebSpace è stata la visualizzazione della struttura spaziale di un sito Web. Grazie all’accuratezza e alla velocità con cui il sistema visivo umano funziona, le rappresentazioni grafiche rendono possibile visualizzare in poco spazio una grande quantità di informazioni.

Un esempio, tratto da [Bertin 1983], è riprodotto nella figura 5.14 dove, ambedue la mappa mostrano il valore di una data caratteristica nelle varie zone della Francia.

Nell’immagine di sinistra sono utilizzati i numeri per rappresentare i differenti valori, mentre in quella di destra vengono visualizzati attraverso la variazione dell’intensità di un pattern di punti.

A differenza dell’immagine con i numeri, la seconda immagine è immediatamente leggibile e aiuta a fare delle analisi sui dati (ad esempio, si può vede chiaramente che i valori più alti sono concentrati nella zona in alto a sinistra).
Questo esempio aiuta a capire l’importanza della visualizzazione grafica per presentare un insieme di dati. Proprio per questo motivo si è dovuto prestare particolare attenzione nella realizzazione dell’interfaccia di visualizzazione dei dati di WebSpace.

Nei seguenti paragrafi verranno illustrati differenti approcci adattati alla visualizzazione della struttura di un sito Web tra cui quello che è stato adottato per la realizzazione di WebSpace.

5.6.1 Mappe disegnate a mano

Il tipo più semplice di mappa di un sito Web è quello in cui sono riportate in una tabella le varie sezioni del sito, suddivise gerarchicamente. Queste mappe risultano intuitive da usare per gli utenti: basta cliccare su un argomento per visualizzare la pagina corrispondente.

Lo svantaggio è che devono essere aggiornate manualmente e che un sito può avere un numero di sezioni troppo grande per essere visualizzato in una mappa sola (basta pensare alle directory di Yahoo!). Un ulteriore problema è che spesso la struttura di un sito Web non è così strettamente gerarchica come si potrebbe immaginare.

Le mappe di questo tipo sono quelle che visivamente risultano più differenti le une dalle altre: si spazia da semplici elenchi testuali a complesse rappresentazione grafiche.

Figura 5.15: Mappa del sito di Pirelli (http://www.pirelli.it) in cui sono mostrate in forma tabellare le varie sezioni che compongono il sito

Figura 5.17: Mappa grafica del sito del LIntAr (http://www.lintar.disco.unimib.it) in cui vengono visualizzate solo le sezioni principali, ogni una contrassegnata da un colore differente.

5.6.2 Mappe a grafo

La visualizzazione della mappa di un sito come una serie di nodi collegati da archi è uno dei più semplici ed intuitivi sistemi di visualizzazione della struttura di un sito Web. Queste mappe vengono generate automaticamente a partire dalla struttura del sito stesso e quindi risultano essere sempre aggiornate e rispecchiano fedelmente la struttura del sito, con tutte le pagine presenti ed i link tra di esse.

In questi tipi di visualizzazione il più grande problema da risolvere è la disposizione dei nodi nello spazio. Se per un sito formato da poche pagine o dove i contenuti sono...
strettamente gerarchici questo non costituisce un problema, per siti formati da molte pagine con molti link diventa difficile generare dei grafi comprensibili.

Questo tipo di visualizzazione è spesso utilizzata nei software per la gestione dei siti, che sono progettati per aiutare i Web Master a visualizzare e gestire siti grandi e complessi. Un esempio di uno di questi tool è Mapuccino è un applicazione, sviluppata presso l’ IBM Research Lab ad Haifa, che permette di costruire dinamicamente mappe visuali a grafo interattive di siti Web. Altri esempi di applicazioni simili sono Astra SiteManager e ClearWeb.

Figura 5.18: Mapuccino
Anche per WebSpace è stata studiata una visualizzazione a grafo della struttura del sito, in cui i nodi vengono disposti a random e successivamente, tramite un algoritmo di tipo Spring-Forces vengono disposti in modo che nessun nodo sia sovrapposto ad un altro e che i nodi collegati siano il più possibile vicini. I nomi delle pagine sono...
visualizzati solo a *mouse-over* in modo da non occupare spazio sopra la mappa e da permettere di disegnare i nodi i più piccoli possibili. Attraverso una finestra con una riproduzione in piccolo della mappa (in alto a destra) è possibile farla scorrere, per avere una superficie di visualizzazione maggiore. Per permettere agli utenti di personalizzare la visualizzazione è stata data la possibilità di trascinare i nodi per disporli secondo lo schema desiderato. Nonostante questi accorgimenti, questo tipo di visuale si è dimostrato non adatta già per siti anche per siti con solo 50 pagine e numerosi link, per cui è da considerare totalmente inutilizzabile per siti di grandi dimensioni.

Figura 5.21: Visualizzazione a grafo di WebSpace; nella seconda immagine è selezionato un nodo e, in basso a sinistra, viene visualizzato il titolo della pagina corrispondente

5.6.3 Mappe fish-eye e ipperboliche

Le visualizzazione di tipo fish-eye e ipperboliche costituiscono una variante delle visuali a grafo e utilizzano una visione grafica interattiva per mostrare la struttura di un sito Web. Queste tecniche si basano sulla visualizzazione della struttura come se si trovasse sotto una lente di ingrandimento, con gli elementi al centro visualizzati più in grande e quelli alle estremità più in piccolo. Generalmente queste mappe sono interattive e gli utenti possono trascinare la mappa per portare al centro (e quindi visualizzare in grande) una parte differente della struttura. Questi tipi di visualizzazione permettono di avere mappe sempre aggiornate dei siti (dato che si basano sulla struttura del sito stesso) e di rappresentare in uno spazio ridotto molte informazioni, ma spesso rischiano di essere poco intuitive e di confondere gli utenti.

Le mappe fish-eye e ipperboliche sono utilizzate sia in programmi di aiuto nella gestione di siti Web che per generare mappe per gli utenti finali dei siti. Un esempio di applicazione che utilizza una visione ipperbolica è Footprints, che è descritta nel paragrafo 2.1.2.3.
Figura 5.22: Mappa ipperbolica del Xerox Reserce Centre Europe (http://www.xrce.xerox.com), attraverso l’ utilizzo del mouse gli utenti possono interagire con la mappa, spostando il fuoco su altre pagine e modificando la lunghezza degli archi.

Figura 5.23: Microsoft WebMapper, software per la gestione di siti Web
CAPITOLO 5. WEBSpace

5.6.4 Mappe Land-Use

Un differente approccio per mostrare la struttura di un sito Web è costituito dalle mappe di tipo Land-Use. Un esempio di mappa di questo tipo è mostrato nella figura 5.25 e si riferisce all’applicazione ET-Map che è stata sviluppata da Hsinchun Chen a dal team di ricerca del Artificial Intelligence Lab della University of Arizona.

La figura 5.25 mostra la spazializzazione di oltre 110.000 pagine della categoria intrattenimento di Yahoo!, ed è un esempio di come è possibile ricercare informazioni sulla musica jazz in Yahoo!. Ogni mappa mostra un gruppo di pagine Web come una forma regolare associata ad un dato argomento. La dimensione di ogni area è data dal numero di pagine contenuto nella categoria [Dodge Kitchim 2001]. Le mappe generate da ET-Map possono essere esplorate interattivamente, cliccando sulle aree relative alle categorie a cui si è interessati. Le mappe vengono create tramite reti neurali Kohonene Self-Organizing map (SOM).

5.6.5 Mappe WebSpace

Dopo vari studi sulla migliore visualizzazione possibile della struttura spaziale, si è scelto di realizzare mappe interattive di tipo Land-Use, ispirate a ET-Map (vedi 5.6.4). Le principali differenze tra la visualizzazione di WebSpace di ET-Map sono il diverso utilizzo che viene fatto del colore nelle mappe e l’ algoritmo utilizzato per la generazione delle stesse.

Mentre ET-Map utilizza differenti colori solo per evidenziare differenti categorie, nella mappe generate da WebSpace i colori vengono utilizzati per mappare il dato relativo alla popolarità della pagina. Le pagine che sono state visitate da molti utenti vengono visualizzate in un verde acceso, che diventa sempre più tenue fino a raggiungere il bianco per le pagine meno visitate. Questa modalità di visualizzazione si ispira a The Map of the Market, dove con i differenti colori veniva mappato il cambiamento del valore delle azioni di una società.

Figura 5.26: Map of the Market
CAPITOLO 5. WEBSPACE

Figura 5.27: Mappe delle pagine Start e Progetti del sito del LIntAr

Un’ ulteriore importante differenza riguarda l’ algoritmo utilizzato per la generazione della mappa. In ET-Map viene utilizzata una rete neurali Kohonen Self-Organizing Map (SOM), mentre per WebSpace si è scelto l’ algoritmo Treemap.

L’ algoritmo Treemap utilizza come dati in ingresso la radice di un albero ed un’ area rettangolare definita dalle coordinate dei punti in alto a sinistra ed in basso a destra P1(x1,y1), Q1(x2,y2). Il numero di suddivisione che il rettangolo subirà dipendono dal numero di archi uscenti dalla radice, mentre l’ area di ogni suddivisione sarà in rapporto al numero di elementi contenuti. L’ algoritmo procede ricorsivamente su ogni sottoalbero utilizzando il rettangolo corrispondente.

Figura 5.28: Confronto tra rappresentazioni ad albero e Treemap
Esistono diverse varianti dell’ algoritmo originale (descritto in [Shneiderman 1992]) che risultano essere più o meno stabili (in rapporto al variare della visualizzazione grafica e dei dati in ingresso), leggibili e in cui l’ ordine degli elementi viene o meno preservato. Maggiori informazioni sui diversi algoritmi possono essere trovate in [Bederson et al 2001].

Dato che gli algoritmi Treemap operano su degli alberi e non su dei grafi (come invece è la struttura di un sito Web), per ogni mappa che viene generata si costruisce un albero che ha come radice la pagina relativa alla mappa che viene generata e che ha come profondità massima 2 (cioè che contiene tutte le pagine raggiungibili attraverso uno o due collegamenti). Tutti i cicli eventualmente presenti vengono eliminati duplicando i nodi che fanno parte dei cicli. L’ albero e di conseguenza la mappa posso quindi presentare più volte, a livelli differenti, la stessa pagina. Ciò non è un problema perché serve a rappresentare la struttura a grafo di un sito Web.

![Figura 5.29: Mappe della pagina Progetti in cui vengono evidenziati in rosso tutti i link che portano alla pagina stessa](image)

5.7 **Sperimentazione**

Dopo essere stata realizzate ed essere stata testata con dei siti Web di poche pagine realizzati unicamente per i test, l’ applicazione WebSpace è stata impiegata in un test su di un sito Web realmente in uso.

Per questa prova, l’ applicazione utilizza il tracker come modalità per tener traccia delle richieste degli utenti e la generazione della struttura del sito Web viene fatta tramite l’ acquisizione del sito al momento dell’ inizializzazione e successivamente ad intervalli regolari per adattare la struttura alle modifiche del sito.
Il sito che è stato utilizzato per la dimostrazione dell’ applicazione è quello del LIntAr (Laboratorio di Intelligenza Artificiale) che si basa sul CMS (Content Management System) SnipSnap. Dato che tutte le pagine del sito sono gestite tramite il CMS è stato sufficiente modificare il template di pagina per inserire il tracker. Questo ha permesso di verificare che l’ applicazione funzionasse, successivamente è stato modificato ancora il template per visualizzare, in un blocco presente in ogni pagina, l’ elenco dei link proposti (cioè i contenuti adattivi).

Questa sperimentazione ha dimostrato che è possibile utilizzare con successo l’ applicazione in un sito Web reale formato da circa 200 pagine.

![Image](image.png)

Figura 5.30: Pagina del sito del sito Web del LIntAr. Si può notare, nella parte bassa a sinistra, il box con le pagine suggerite

16http://www.lintar.disco.unimib.it
17http://www.snipsnap.org
Capitolo 6

Sviluppi futuri

I possibili sviluppi futuri di questo lavoro di tesi potranno riguardare tutte le aree su cui si estende, cioè la modellazione del Web come un sistema multi-agente basato sul modello MMASS, la piattaforma ad agenti e l’ applicazione WebSpace.

**Modello**
Per quanto riguarda il modello, potrebbe essere utile definire una semantica di alto livello per descrivere i contenuti delle pagine che potrà essere utilizzata dalle applicazioni che si occupano dell’ analisi dei contenuti dei siti Web e della generazione di contenuti adattivi.

**Piattaforma**
Per quanto riguarda la piattaforma, dovranno essere sviluppatte alcune parti relative al modello MMASS che attualmente non sono state implementate, come il modello di reazione sincrona tra agenti situati in posti adiacenti e le interfacce per la diffusione di campi tra spazi differenti.

Dovrà essere definito un insieme di API che siano stabili anche per le future versioni e che siano adattabili anche ad altri linguaggi di programmazione, per permettere di realizzare implementazioni in altri linguaggi di piattaforme per il modello MMASS.

Sarà necessario creare nuovi strumenti che semplifichino agli sviluppatori la creazione di applicazioni.

Una condizione necessaria per il completamento della piattaforma e la creazione di un ambiente software utile agli sviluppatori, sarà l’ implementazione di applicazioni che faranno riferimento ad altri domini applicativi reali differenti dalla modellazione del Web come un sistema ad agenti.
**WebSpace** L'applicazione WebSpace potrà essere estesa inserendo il supporto per la multicanalità, in modo da permettere agli utenti di un sito Web di comunicare tra loro attraverso mezzi differenti (chat via Web, mail, messanger).

Figura 6.1: Estensione dell’architettura di WebSpace per supportare media differenti

Figura 6.2: Esempio di comunicazione tra due utenti

Potranno essere sviluppati dei meccanismi più avanzati per la generazione di contenuti adattivi da proporre agli utenti, prevedendo sia la *customizzazione* che l’*ottimizzazione*.

Dovrà essere prevista la possibilità per il Web Master di visualizzare informazioni personalizzate, ad esempio visualizzando le informazioni solo su un sottogruppo di utenti, in modo che lo strumento di supporto ai gestori di siti possa fornire informazioni maggiormente utili.
Bibliografia


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
</table>
BIBLIOGRAFIA


[httpd@w3.org 1995] httpd@w3.org, *Logging Control In W3C httpd*, http://www.w3.org/Daemon/User/Config/Logging.html, 1995


