ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

INTEGRAZIONE DI TECNOLOGIE PER LA DISTRIBUZIONE DI CONTENUTI WEB IN SISTEMI MULTI-AGENTE:UN APPROCCIO BASATO SULL'INFRASTRUTTURA TUCSON

Elaborata nel corso di: Sistemi Distribuiti

Tesi di Laurea di:
STEFANO MONTINI

Relatore:
Prof. ALESSANDRO RICCI

ANNO ACCADEMICO 2004–2005
SESSIONE II

Indice

Introduzione **v**

1 Feed RSS **1**
1.1 Cos'è RSS 1
1.1.1 Un po' di storia 2
1.1.2 Cosa sono i Feed 3
1.1.3 Come possono essere usati 4
1.2 Strutttura di un feed 5
1.2.1 L'elemento *channel* 6
1.2.2 L'elemento *image* 6
1.2.3 L'elemento *item* 7
1.2.4 L'elemento *textInput* 8
1.3 Formati e Moduli 8
1.3.1 RSS 2.0 9
1.3.2 RSS 1.0 10
1.3.3 Atom 12

2 Agenti e Tucson **15**
2.1 Agenti come paradigmi dell'ingegneria del software . . 16
2.1.1 Cosa sono e loro caratteristiche 17
2.1.2 Tipologie e ambienti in cui operano 19
2.1.3 Agenti V.S. Oggetti 20
2.2 Agenti e MAS: comunicazione, coordinazione e organizzazione 21
2.2.1 Interazione/Comunicazione: diretta e mediata . 23
2.3 Infrastruttura di coordinazione Tucson 24
2.3.1 Tuple Centre 24

2.3.2 ReSpecT 27

3 RSS syndication in TuCSoN **29**
3.1 Analisi 29
3.2 Progetto 31

4 Implementazione e testing **35**

5 Conclusioni **45**

Introduzione

In tempi recenti si è assistito ad un enorme sviluppo di tecnologie e applicazioni Web-based per la diffusione e distribuzione di contenuti sul Web. Soffermiamoci per un attimo a pensare al numero di informazioni che è possibile reperire in Internet. Probabilmente non basterebbero ore per stilare una lista di tutto il materiale che la rete mette a disposizione, anche a causa del numero di informazioni non necessarie che spesso ci troviamo di fronte quando navighiamo alla ricerca di un qualche contenuto. Non è difficile comprendere come possa essere lento e dispendioso ogni volta accedere al sito, verificare la presenza degli ultimi aggiornamenti e leggerli.
È proprio qui che ci viene in aiuto RSS: esso nasce, infatti, con l'intento di permettere rapidi e veloci aggiornamenti da più fonti senza la necessità di dover navigare singolarmente ciascun sito, ma semplicemente verificando una lista dei contenuti pubblicati offerta dal sito stesso. Le notizie sono inserite dai gestori dei siti in appositi canali, in cui l'utente può accedere attraverso l'utilizzo di particolari strumenti che gli permettono di conoscere gli ultimi eventi disponibili. Nel capitolo 1 si analizza la struttura e il funzionamento di questa particolare tecnologia.
Partendo da questa innovativa applicazione che si sta diffondendo velocemente in tutto il Web e che è ormai conosciuta dalla maggior parte dei visitatori della rete, si è pensato di utilizzarla all'interno di sistemi complessi realizzati attravero il paradigma ad agenti, il cui interesse sta aumentando continuamente.
Questo nuovo paradigma che sta piano piano diffondendosi, si mostra sicuramente adatto allo sviluppo di applicazioni complesse e dinamiche. Esso è visto non soltanto come una naturale evoluzione del paradigma orientato agli oggetti, ma rappresenta una vera e propria rivoluzione,

che cambia radicalmente il modo di sviluppare il software. Si può già osservare la grande quantità di sistemi distribuiti complessi basati sugli agenti che sono nati e si sono sviluppati nei più disparati campi scientifici e industriali. Nel Capitolo 2 è descritto in modo più approfondito il concetto di agente e di sistema multi-agente.
In generale, un qualunque sistema automatico complesso è in grado di osservare eventi pubblicati via Web e di poterli manipolare secondo le proprie esigenze. Non è più l'utente ad osservare e gestire direttamente le informazioni trovate nella rete, ma vengono realizzate vere e proprie entità che manipolano e organizzano questi eventi. In questo modo il sistema è in grado di osservare le informazioni pubblicate via Web e ragionare su queste per svolgere particolari azioni coordinate. Le news sono a disposizione di agenti che le gestiscono e le utilizzano per raggiungere un certo insieme di obiettivi. Si possono così realizzare sistemi complessi di ogni tipo che recuperano i nuovi aggiornamenti e che, attraverso processi automatici di gestione, li manipolano e, a seconda delle notizie osservate, agiscono di conseguenza. Gli agenti utilizzano quindi le informazioni come punto di partenza per eseguire determinate attività: una volta ricevuto un determinato evento, è possibile attivare l'esecuzione dell'agente in attesa di quel particolare aggiornamento che metterà in esecuzione altri agenti fino all'attivazione dell'intero sistema che raggiungerà così l'obiettivo per cui è stato realizzato.
Un sistema costruito con queste tecnologie potrebbe essere collegato con un sito che informa gli utenti dell'uscita degli ultimi film nelle sale cinematografiche. Sarà presente un agente che osserverà periodicamente se un nuovo film di un determinato regista è stato inserito nel sito. L'agente controllerà solo i films di quel particolare regista: tralascierà quelli realizzati da altri, non informando il sistema della loro presenza. Ogni volta che un suo film viene inserito sul sito, l'agente avvisa la comunità di agenti di questo nuovo aggiornamento. Gli altri agenti potrebbero essere realizzati in modo tale da avvisare l'utente tramite un'email il giorno stesso in cui il film uscirà.
I sistemi possono anche essere realizzati in maniera più complessa, con l'utilizzo di agenti che coordinano automaticamente le loro attività e attendono di ricevere particolari informazioni da altre entità.
Il sistema potrebbe essere costituito da più agenti che osservano aste

online e recuperano e gestiscono informazioni sugli oggetti venduti. Gli agenti sono pronti ad informare il sistema ogni volta che viene messo all'asta un determinato oggetto: ognuno di questi agenti recupererà inoltre tutte le informazioni relative a quell'oggetto quali caratteristiche e prezzo. Il sistema riceverà da essi le diverse proprietà di ogni oggetto presente nelle diverse aste e ragionerà su queste caratteristiche: alcuni agenti, ad esempio, potrebbero confrontare il prezzo di tutti gli oggetti con le stesse caratteristiche; altri potrebbero stabilire quale sia l'oggetto migliore tra tutti quelli che hanno lo stesso prezzo ma caratteristiche diverse; altri ancora potrebbero essere dotati di regole che stabiliscono se un'offerta di un oggetto è migliore di un'altra dello stesso oggetto che ha però caratteristiche e prezzo diversi. Si potrebbe, inoltre, una volta stabilito l'oggetto migliore in termini di qualità e di prezzo, affidare ad un'agente il compito di rilanciare il prezzo dell'oggetto in quella determinata asta.
Con questi semplici esempi è possibile immaginare come sia utile la realizzazione di questi particolari tipi di sistemi. Non importa quello che farà il sistema ogni volta che riceverà una particolare notizia. Quello che interessa è riuscire a creare il punto di partenza per mettere in esecuzione l'intero sistema: esso consiste nella realizzazione di un'entità capace di osservare e recuperare le informazioni su un nuovo evento pubblicato via Web e di informare le altre entità della presenza di tale aggiornamento.
In particolare si cercherà di realizzare un'applicazione che sia in grado di interagire con file RSS prelevando da essi tutto ciò che può servire per identificare e descrivere un'informazione. Verrà creato un agente che dovrà raccogliere queste notizie ed informare tutti gli altri agenti che fanno parte del sistema sulla disponibilità di un nuovo aggiornamento. Quello che farannno poi gli altri agenti sarà solamente una verifica del corretto funzionamento dell'applicazione.

Capitolo 1

Feed RSS

Lo sviluppo continuo del Web come raccolta di informazioni provenienti da tutto il mondo ha portato la nascita di numerose tecnologie per la disribuzione e diffusione di contenuto informativo di ogni tipo. Puoi 'sfogliare' pagine Web per soddisfare la tua voglia e la tua curiosità di conoscere tutto di tutti, per imparare ciò che più ti interessa, per essere in ogni momento aggiornato su cosa succede nel mondo. I 'lettori', a causa dell'eccessiva quantità di informazioni sparse in tutta la rete, sono però costretti a visitare una grandissima quantità di siti prima di trovare ciò che cercano. RSS è uno dei più popolari formati per la distibuzione di contenuti sul Web che è riuscito a superare questo problema.

1.1 Cos'è RSS

Rich Site Summary (RSS) è un formato che permette la diffusione e distribuzione su diversi canali di liste di link, titoli e sommary di news. È questa una delle tante definizioni che si può trovare in rete e che fornisce, in maniera molto semplice e generale, un'idea di che cosa si intende per RSS. Si tratta di un formato basato su XML che garantisce la *Syndication* di liste di brevi notizie sul Web [7]. Per *Syndication* si intende l'affidamento di testi ad un'agenzia che provvede a distribuirli a riviste e a siti. La parola è usata anche per un sistema di collaborazione e collegamento tra reti televisive o radiofoniche. Una rete produce un programma e poi lo distribuisce su un certo numero di

stazioni locali. Essa è molto simile all'offerta proposta da molti siti di inviare periodicamente via e-mail nuove notizie su argomenti che interessano i loro visitatori. La grande differenza, consiste nel fatto che i lettori non devono fornire il loro indirizzo e-mail, riducendo in questo modo la privacy dell'individuo: sono i siti che mettono a disposizione file RSS che contengono una lista di tutte le notizie aggiornate in tempo reale. Chiunque, attraverso il proprio computer, può regolarmente utilizzare questi file e prendere da essi le informazioni più recenti in modo tale da essere sempre aggiornato su tutto.
Oggi RSS è lo standard per l'esportazione di contenuti Web. I principali siti di informazione, i quotidiani online, i fornitori di contenuti: tutti sembrano aver adottato il formato RSS. Gli utenti possono oggi accedere a migliaia di feed RSS.

1.1.1 Un po' di storia

Come molti argomenti dell'informatica, anche la storia di RSS è fatta di rivalità, piccole ripicche e gruppi contrapposti. Essa può essere ricondotta al 1997 con la creazione di *Resource Description Framework* (RDF), un linguaggio di markup utilizzato per rappresentare sul Web le informazioni e i legami che intercorrono tra di esse. RDF è molto simile a RSS a tal punto che molti definiscono RSS come *RDF Site Summary* per evidenziare l'origine di questo particolare formato [5].
Nel 1999 Netscape creò uno standard chiamato RSS versione 0.90: fu questo l'inizio di quello che noi conosciamo come RSS. Erano gli anni in cui Netscape puntava sulla creazione di portali personalizzati (MyNetscape) da riempire con liste di link o di news selezionate dall'utente in base alle sue preferenze o esigenze. Il formato permetteva la visualizzazione sul portale di titoli e link relativi a notizie pubblicate su altri siti e rese disponibili attenendosi a specifiche ben precise. Fu subito un grande successo: in poco tempo, centinaia di fornitori di contenuti aderirono all'iniziativa e il portale My Netscape potè beneficiare di una vasta raccolta di notizie a disposizione dei propri utenti registrati.
In seguito, un dipendente della Netscape, Dan Libby, perfezionò la versione realizzando la 0.91.
Ma Netscape non era soddisfatta: il formato era troppo complicato

per l'obiettivo che cercavano di raggiungere; decise quindi di abbandonare lo sviluppo di RSS.
Il formato venne allora adottato da Dane Winer della Userland, come base dei sistemi di *content management* dell'azienda, che rilasciò una versione di RSS, figlia del formato utilizzato da Netscape, denominata 0.92: le sue specifiche erano leggermente differenti da quelle della versione 0.91.
Intanto, un altro gruppo di programmatori diede vita alla versione 1.0, incompatibile con la precedente: l'obiettivo era lo stesso ma le specifiche erano molto differenti in quanto conformi allo standard RDF. Quest'ultima versione prese l'acronimo di RDF Site Summary.
Sentitosi scavalcato e deprecando certi aspetti di complessità di RSS 1.0, Dave Winer proseguì per la sua via, dando nuova vita al vecchio formato e sfornando, nell'ordine le versioni RSS 0.92, 0.93, 0.94, fino a quella attuale: RSS 2.0 . Per rimarcare la distanza da RSS 1.0 e sottolineare il carattere di semplicità della sua proposta, Winer ha dato al suo formato il nome di Really Simple Syndication. Per far in modo che le nuove specifiche siano accettate e utilizzate da tutti, essa fu donata alla scuola di Harward Law la quale è ora responsabile del suo sviluppo futuro.

1.1.2 Cosa sono i Feed

I Feed RSS o più comunemente chiamati file RSS sono un tipo di metadati. Per metadati si intende unità di informazioni comunemente usate per descrivere informazioni riguardanti contenuti, contesti e caratteristiche di altri dati; per esempio se consideriamo un articolo o una notizia, i metadati potrebbero essere l'autore, la lingua, la copyright e tutte le informazioni inerenti all'articolo o alla singola notizia. Sebbene questa spiegazione può risultare contorta e di difficile comprensione, si vedrà più avanti come sia semplice capire la struttura di questi file. Intanto è opportuno accennare qualcosa su di essi. Si tratta di file XML che contengono una lista di *items* o *entries* che corrispondono alle notizie, ciascuna delle quali è caratterizzata da un link, un titolo, una descrizione e altre informazioni utili per la sua identificazione. È possibile vedere un semplice esempio di questo nella pagina seguente. Inoltre ciascun feed può avere, oltre a questi, altri

elementi ciascuno con i propri campi per caratterizzare maggiormente la sua struttura e renderlo così più dettagliato e completo.

```
<item>
    <title>Amazon.com vende porzioni di libro</title>
    <link>http://webnews.html.it/news/3120.htm</link>
    <description>Amazon.com scommette sul mercato dei libri digitalizzati
    proponendo fin da subito una importante novità:
    i libri sono acquistabili anche nelle loro singole porzioni
    quali singoli capitoli o singole pagine. </description>
</item>
```

Figura 1.1: Un item in RSS

1.1.3 Come possono essere usati

Gli aggregatori sono il più comune e diffuso mezzo per utilizzare i feeds. Un news aggregator (detto anche feed aggregator o newsreader) è un applicativo in grado di leggere periodicamente un flusso di notizie in formato RSS e di presentarle con un'interfaccia grafica simile, nella maggioranza dei casi, a quella di un programma di posta elettronica. Esistono fondamentalmente due tipi di news aggregator: on line (Web Aggregators) e desktop. I primi sono veri e propri siti, che permettono all'utente di selezionare una lista di feed RSS e di visualizzarli in una pagina che viene aggiornata di tanto in tanto (ad esempio *my Yahoo*). Gli altri sono veri e propri programmi che l'utente scarica dalla Rete e installa sulla propria macchina: alcuni di questi sono software autonomi altri sono integrati in programmi già utilizzati come Microsoft Outlook e il motore Mozilla. Gli aggregatori possono offrire una varietà di caratteristiche come la possibilità di visualizzare la lista di tutte le notizie, nascondendo quelle che il visitatore ha già visto e suddividendole in categorie a seconda del loro argomento. È necessario inserire nella lista dell'aggregatore l'indirizzo del feed RSS che contiene le notizie desiderate, in modo tale che esso rilevi se il produttore del feed ha effettuato aggiornamenti su di stesso, effettuando così il

download delle notizie ad intervalli di tempo regolari. In questo modo l'utente può essere informato quasi in tempo reale quando un sito è stato aggiornato, evitando dunque di dover visitare, uno per uno, i siti da cui provengono le notizie stesse. Per trovare il link del feed RSS, basta cercare l'icona XML o RSS e poi copiare e incollare il link del canale nell'apposita lista dell'aggregatore.
Un altro utilizzo dei feeds è la possibilità di ricercare siti attraverso motori di ricerca basati su RSS. Questo è un grande vantaggio in quanto essi forniscono singoli items anziche pagine che contengono una gran quantità di notizie.
È possibile inoltre ripubblicare nel proprio sito feeds presenti in altri siti: in questo modo ciascuno può rappresentare come meglio vuole la lista delle notizie.
Essendo un file RSS molto facile da utilizzare e vedendo nella prossima sezione la semplicità della sua struttura, molte persone creano personalmente il proprio feed utilizzando qualsiasi tipo di editor XML. Una volta realizzato è necessario validarlo per assicurarsi che venga interpretato dai lettori: proporre un feed non valido, ovvero non conforme alle specifiche del formato adottato, equivale a non averlo per niente. Solo a qeusto punto è possibile pubblicarlo sul proprio sito.

1.2 Strutttura di un feed

In questo sezione, viene descritta la struttura generale di un feed RSS, analizzando gli elementi principali e comuni ad entrambe le versioni RSS 1.0 e 2.0 viste successivamente all'interno del capitolo [6].
Innanzitutto, come ho più volte accennato, RSS è una applicazione XML: per questo tutti i feed realizzati possono presentare (non è obbligatorio ma quasi tutti ce l'hanno) il classico prologo tipico di questo linguaggio. Successivamente è presente l'elemento radice che contiene tutti gli altri elementi. Esso è obbligatorio e definisce l'inizio del file RSS: è qui specificata la versione del formato utilizzato.
All'interno dell'elemento radice sono presenti quattro principali sezioni: *channel, item, image* e *textInput*, ciascuna con i prorpi sottoelementi.

1.2.1 L'elemento *channel*

L'elemento *channel* contiene metadati che descrivono il canale stesso, spiegando chi lo ha creato e fornendo un'introduzione su quello che contiene. Esso è obbligatorio e include tre sottoelementi, anch'essi obbligatori, che sono il titolo descrittivo del canale, l'URL del sito HTML che corrisponde ad esso e una breve descrizione del suo contenuto. L'immagine mostra un esempio di questo.

```
<channel>
  <title>Canale di esempio</title>
  <link>http://esempio.com</link>
  <description>Il mio canale contiene notizie, interviste
   e molto altro</description>
</channel>
```

Figura 1.2: L'elemento channel

All'interno di esso, oltre a questi, trovano posto l'*image* (il logo del canale), l'*item* (la notizia) e il *textInput*.

1.2.2 L'elemento *image*

L'elemento *image* è un un metadato opzionale utilizzato per includere un'immagine da associare alla rappresentazione HTML del canale. L'immagine dovrebbe essere di un formato supportato dalla maggior parte dei browser Web. Le dimensioni di default sono di 88 pixels in larghezza e 31 pixels in altezza. È possibile tuttavia definire le sue dimensioni attraverso i due elementi opzionali corrispondenti: i valori della larghezza devono essere compresi tra 1 e 144 mentre quelli per l'altezza tra 1 e 400.
I suoi tre metadati obbligatori sono: *title* che descrive l'immagine del canale e che è associato all'attributo ALT del tag <img> nella sua rappresentazione HTML; *url* è l'URL dell'immagine GIF, JPEG o PNG; *link* è l'URL del sito HTML che corrisponde al canale.

```
<image>
  <title>HTML.it - News</title>
  <url>http://www.html.it/img/88x30a.gif</url>
  <link>http://webnews.html.it/</link>
  <width>88</width>
  <height>31</height>
</image>
```

Figura 1.3: L'elemento image

1.2.3 L'elemento *item*

L'*item*, il metadato più importante del canale, costituisce la parte dinamica del file RSS. Mentre channel, image e textInput rappresentano l'identità del canale e restano la stesso periodo di tempo all'interno di esso, gli items si susseguono continuamente: la frequenza con la quale ciascun item viene sostituito da un altro costituisce il valore del canale. Esso rappresenta la notizia dell'ultima ora: all'interno del canale sono

```
<item>
    <title>Amazon.com vende porzioni di libro</title>
    <link>http://webnews.html.it/news/3120.htm</link>
    <description>Amazon.com scommette sul mercato dei libri digitalizzati
    proponendo fin da subito una importante novità:
    i libri sono acquistabili anche nelle loro singole porzioni
    quali singoli capitoli o singole pagine. </description>
</item>
```

Figura 1.4: L'elemento item

presenti un numero limitato di items (circa 15) dal momento che si cerca di distribuire solo le notizie più recenti e quindi più interessanti per i visitatori. L'item contiene il titolo della notizia (*title*), l'URL della pagina in cui essa è contenuta (*link*) e una sintesi contenente la

prima parte o il testo completo dell'articolo (*description*). Nessuno di questi elementi è obbligatorio, ma lo schema tipico è quello riportato nell'immagine.

1.2.4 L'elemento *textInput*

L'elemento *textInput* fornisce un metodo per inviare dei dati mediante una form verso un URL stabilito, generalmente situato nel sito principale. È un elemento opzionale e tipicamente è usato come area di ricerca o form di iscrizione. Contiene quattro metadati obbligatori: *title* che rappresenta il nome del pulsante submit associato all'elemento; *description* che fornisce una breve descrizione dello scopo del campo; *name* che rappresenta il nome del campo di testo e descrive brevemente il suo contenuto; *link* che fornisce l'URL verso il quale un eventuale submit del textinput sarà indirizzato.

```
<textinput>
  <title>Send</title>
  <description>Comments about MozillaZine</description>
  <name>responseText</name>
  <link>http://www.mozillazine.org/cgi-bin/sampleonly.cgi</link>
</textinput>
```

Figura 1.5: L'elemento textInput

Sono questi i quattro elementi principali di un feed RSS. Nella prossima sezione vedremo come sia possibile, utilizzando le diverse versioni di RSS, estendere questi file con l'inserimento di metadati aggiuntivi: questi ulteriori elementi o sono già presenti nelle specifiche oppure vengono importati attraverso l'utilizzo di moduli.

1.3 Formati e Moduli

Come abbiamo visto precedentemente parlando della nascita e sviluppo di RSS, ci sono state diverse versioni del formato ma sono due quelle

che vengono maggiormente utilizzate nei giorni nostri: RSS 1.0 e RSS 2.0 [8]. Non bisogna pensare, però, che le versioni attualmente superate siano scomparse. I feed in formato RSS 0.91 o 0.92 sono molto utilizzati, principalmente per la loro semplicità e per il fatto di essere più che sufficienti per una syndication di base, ridotta all'essenziale. Dal momento che l'ultima versione realizzata, la 2.0, ha praticamente le stesse specifiche di quelle della serie 0.9, con qualche perfezionamento, vedremo come le caratteristiche di questa sono molto diverse da quelle della 1.0, realizzata da un altro gruppo di programmatori.
La situazione, dunque, è che abbiamo due formati concorrenti per fare essenzialmente la stessa cosa. Entrambi sono basati su XML e hanno la stessa struttura di base. Ciò che cambia è la filosofia di fondo. RSS 2.0 punta alla semplicità, RSS 1.0 guarda alla ricchezza semantica e all'estendibilità con moduli esterni (cose comunque possibili anche per il 'rivale').
Inoltre, è presente un'altra possibile scelta: Atom. Molto simile a RSS, Atom fu creato da persone che volevano migliorare e standardizzare il formato precedente, realizzando una specifica di distribuzione di notizie maggiormente documentata e in grado di diffondere informazioni più complesse.
Ognuno di questi tre formati ha le sue caratteristiche e specifiche, vantaggi e svantaggi: qui di seguito viene riportata una breve descrizione dei tre modelli in modo da evidenziare le particolarità di ciascuno di essi.
La scelta di uno tra questi tre formati non dipende dalla maggiore funzionalità e diffusione di uno rispetto ad un altro: gli aggregatori e altri software riescono a gestire qualsiasi tipo di feed indipendentemente dalle specifiche utilizzate. È il gusto e le preferenze personali che spingono a scegliere quale formato utilizzare [1].

1.3.1 RSS 2.0

È questa la versione realizzata da Dane Winer: essa è basata sulla 0.91 creata da Netscape e ridefinita da Userland. In questa versione RSS sta per *Really Simple Syndication* e le sue peculiarità sono la semplicità e l'intuizione.
Come prima cosa, è importante sottolineare la grande quantità di ele-

menti previsti dalla specifica. Oltre a quelli principali, visti precedentemente, che caratterizzano il canale (*link, title, description, image, item*), è possibile specificare all'interno di ciascun feed, altri particolari metadati [14].
Alcuni di questi sono: *webMaster* e *managingEditor* che permettono di identificare e contattare il responsabile del feed; *lastBuildDate* che mostra la data e l'ora dell'ultima volta in cui è stato aggiornato il file; *docs* che contiene l'indirizzo della pagina contenente la documentazione del formato utilizzato. Anche per l'elemento *item* sono stati inclusi ulteriori metadati: ad esempio *enclosure* che permette di collegare un link ad un file, creando un allegato o *guid* che identifica univocamente il feed.
Proprio perché supporta un gran numero di elementi e sottoelementi, non è necessario difinire all'interno di ciascun feed alcun tipo di Namespace predefinito (come invece è necessario nei feed RSS 1.0). All'interno dei file è comunque possibile trovare Namespace XML: infatti , essendo RSS un formato modulare, si possono aggiungere tranquillamente nuovi elementi per qualsiasi esigenza. Per fare questo è necessario specificare il Namespace utilizzato per evitare conflitti nei loro nomi: nell'esempio viene utilizzato nell'ultimo item il modulo book; questo è possibile in quanto è stato definito il Namespace corrispondente.
Osservando l'immagine e confrontandola con un file RSS 1.0 è possibile notare due piccole ma importantissimi differenze. La prima riguarda la definizione dell'elemento radice: qui *<rss version='2.0'>*. La seconda invece riguarda la posizione degli item che sono inseriti all'interno dell'elemento channel e non al di fuori di esso.

1.3.2 RSS 1.0

RSS 1.0 sta per *RDF Site Summary*. Come abbiamo accennato precedentemente, questa versione deriva dal linguaggio standard RDF utilizzato per la descrizione delle risorse sul Web. Discendendo dal modello standard, i feed RSS 1.0 si diffusero velocemente e diventarono facilmente parte integrante del Web. Questi sono molto simili ai feed della versione RSS 2.0 precedentemente analizzata anche se mostrano alcuni elementi differenti [13]:

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Canale di esempio</title>
    <link>http://esempio.com</link>
    <description>Il mio canale</description>
    <item>
        <title>Notizie del primo di dicembre</title>
        <link>http://esempio.com/2005/12/1</link>
        <description>L'inizio di dicembre</description>
    </item>
    <item>
        <title>Notizie del secondo di dicembre</title>
        <link>http://esempio.com/2005/12/2</link>
        <description>Siamo già al 2</description>
    </item>
    <item xmlns:book="http://namespace.esempio.com/book/1.0">
        <title>Assassinio</title>
        <link>http://www.amazon.com/exec/obidos/tg/detail/-/0553575376</link>
        <book:isbn>0553575376</book:isbn>
    </item>
  </channel>
</rss>
```

Figura 1.6: RSS 2.0 feed

- L'intero contenuto del feed è racchiuso dall'elemento radice *<rdf:RDF>...</rdf:RDF>* che evidenzia la sua appartenenza al modello RDF

- Viene utilizzato l'atributo *rdf:about*: associato al canale esso contiene l'URL che lo identifica e che deve essere unico all'interno del documento RSS; riferito a ciascun item, è di solito uguale al valore del sottoelemento *<link>* .

- All'interno del canale è presente l'elemento *<items>* che con-

tiene la lista di tutti gli items del canale, in modo tale che qualsiasi elaboratore RDF possa tenere traccia delle loro relazioni.

- Ciascun item si trova all'esterno del canale.
- Alcuni metadati utilizzano l'attributo *rdf:resource* per specificare il link, evitando così di metterlo all'interno dell'elemento

Queste particolari caratteristiche si possono osservare dalla figura presente nella pagina successiva. Inoltre è possibile notare l'utilizzo di particolari Namespace. La caratteristica principale di RSS 1.0 è, infatti, l'estensibilità. Per garantire questo, RSS 1.0 utilizza moduli, basati ciascuno sul proprio Namespace, che consentono di estendere l'RSS senza la necessità di riscritture iterative della specifica centrale ma soprattutto senza interferenze di nomenclatura. Gli unici moduli integrati con l'RSS 1.0 sono Dublin Core e Syndication. È Dublin Core quello più utilizzato: è una raccolta di elementi che inseriscono informazioni aggiuntive sia all'interno dei feeds (nell'elemento canale) sia all'interno dei singoli items. Alcuni elementi sono: *dc:date* che associa la data di pubblicazione all'item; *dc:subject* che fornisce una categoria ai feeds o agli items; *dc:creator* che associa l'autore a ciascun items. Nell'immagine si può notare che è stato necessario dichiarare all'inizio del file il Namespace ad esso associato per poter utilizzare questi particolari elementi.
Da quanto detto, si può vedere come RSS 1.0 sia più prolisso del 2.0, in quanto, per essere compatibile con le altre versione di RSS, esso deve contenere i markup richiesti dagli elaboratori RDF.

1.3.3 Atom

Per introdurre una certa stabilità nella distribuzione di informazioni sul Web, un gruppo di lavoro realizzò un formato chiamato Atom. Molto simile ad entrambe le specifiche RSS, è stato approvato dalla IETF (una comunità aperta di specialisti e ricercatori interessati all'evoluzione tecnica e tecnologica di Internet) ed è diventato uno standard per la sottoscrizione di contenuti Web. È questa la principale differenza rispetto alle versioni non ufficiali di RSS 1.0 e 2.0: le sue specifiche non possono essere modificate e le procedure standard garantiscono una completa stabilità del formato.

```
<?xml version="1.0">
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://purl.org/rss/1.0/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
>
 <channel rdf:about="http://esempio.com/news.rss">
  <title>Canale di esempio</title>
  <link>http://esempio.com</link>
  <description>Il mio canale</description>
  <items>
    <rdf:Seq>
      <rdf:li rdf:resource="http://esempio.com/2005/12/1" />
      <rdf:li rdf:resource="http://esempio.com/2005/12/2" />
    </rdf:Seq>
  </items>
 </channel>
 <item rdf:about="http://esempio.com/2005/12/1">
  <title>Notizie del primo di dicembre</title>
  <link>http://esempio.com/2005/12/1</link>
  <description>L'inizio di dicembre</description>
  <dc:subject>NewTeam</dc:subject>
  <dc:creator>Stefano Montini</dc:creator>
  <dc:date>2005-12-1</dc:date>
 </item>
 <item rdf:about="http://esempio.com/2005/12/2">
  <title>Notizie del secondo di dicembre</title>
  <link>http://esempio.com/2005/12/2</link>
  <description>Siamo già al 2</description>
  <dc:subject>Allsystem</dc:subject>
  <dc:creator>Stefano Montini</dc:creator>
  <dc:date>2005-12-2</dc:date>
 </item>
</rdf:RDF>
```

Figura 1.7: RSS 1.0 feed

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
</feed>
```

Figura 1.8: Atom feed

Come si può vedere dall'immagine, è anch'esso un formato basato su XML e caratterizzato da particolari metadati: l'elemento *feed* (analogo al *channel* di RSS) che contiene tutti gli elementi necessari come ad esempio *entrys* (analogo di *items*) e *entry* che rappresentano le notizie. Le diverse *entry* contengono a loro volta metadati che descrivono, come nel formato RSS, il titolo (*title*), il link (*link*) e la descrizione (*summary*).
Come abbiamo visto, è molto simile ad RSS ma, essendo un formato relativamente nuovo (approvato nel 2004), non è ampiamente supportato come le specifiche RSS ed è per questo meno utilizzato dai produttori di notizie.

Dopo questa panoramica sul significato e sull'utilizzo di RSS, analizzerò nel prossimo capitolo le caratteristiche e le funzionalità di sistemi distribuiti complessi realizzati attraverso il paradigma ad agenti.

Capitolo 2

Agenti e Tucson

Una delle maggiori difficoltà nel mondo dell'ingegneria del software è la realizzazione e lo sviluppo di applicazioni di alta qualità in grado di migliorare e semplificare l'interazione e la coordinazione delle diverse parti che costituiscono un sistema. Ogni giorno, un grande numero di ricercatori si occupano di trovare e perfezionare tecniche per la gestione di sistemi distribuiti complessi in cui garantire un buon livello di comunicazione e interoperabilità. Una delle maggiori teorie proposte in questo ambito è sicuramente la teoria degli agenti.
Un agente è considerato come una particolare entità software in grado di agire in modo autonomo prelevando informazioni dall'ambiente in cui si trova e agendo secondo la propria base di conoscenze, di scambiare informazioni con altri agenti o con esseri umani, e di prendere l'iniziativa per soddisfare i propri obiettivi. Un agente è detto autonomo o intelligente (autonomous agent o intelligent agent) quando ha necessità di operare in un ambiente dinamico ed imprevedibile, nel quale le azioni hanno una certa probabilità di fallire e devono essere prese rapidamente. La realizzazione e lo sviluppo di sistemi attraverso l'utilizzo di agenti autonomi, che interagiscono tra loro e con l'ambiente esterno (sistemi multi-agente), ha portato ad un notevole sviluppo dei processi nel campo dei sistemi intelligenti distribuiti.
In questo capitolo verrà spiegato e illustrato il concetto di agente e sistema multi-agente sottolineando le principali caratteristiche che hanno favorito l'utilizzo di questa particolare tecnologia. Inoltre si analizzerà una particolare infrastruttura per la comunicazione e coordinazione

degli agenti: l'infrastruttura Tucson.

2.1 Agenti come paradigmi dell'ingegneria del software

I sistemi software svolgono un ruolo molto importante nello sviluppo e nella diffusione di informazioni all'interno della nostra società. Basta vedere la grande quantità di tecnologie per l'informazione che vengono realizzate sia in specifici campi quali le telecomunicazioni e il controllo industriale, sia nella vita di tutti i giorni. La realizzazione di questi software è però molto complessa a causa delle numerose funzionalità che essi devono garantire e a causa della flessibilità ai possibili cambiamenti che devono supportare.
Il ruolo dell'ingengeria del software è quello di trovare strutture e tecniche che rendano più facile la manipolazione di questa complessità. Una delle soluzioni più diffuse è quella di suddividere il sistema in parti più piccole e quindi maggiormente gestibili: ciascuna di queste può così essere trattata separatamente, indipendentemente dalle altre. Questo limita notevolmente il raggio d'azione dell'ingegnere il quale incontrerà minore difficoltà a realizzare ciascun pezzo del sistema.
Un altro elemento importante è l'astrazione: la definizione di un modello semplificato del sistema in cui vengono privilegiate alcune sue proprietà rispetto ad altre. Il sistema viene così semplificato in quanto l'attenzione di chi lo realizza si concentra maggiormente sulle parti più importanti e più necessarie per la sua costruzione.
Anche l'organizzazione gioca un ruolo fondamentale nella semplicità di un sistema: specificando e manipolando le relazioni tra le varie parti che lo costituiscono, gli ingegneri possono riunire i componenti base in unità di più alto livello e analizzare e descrivere adeguatamente le relazioni tra queste unità.
Infine risulta indispensabile la definizione delle interazioni che si verificano tra i diversi componenti del sistema, specificando i loro ruoli sociali e le norme che governano il loro comportamento. Non solo si analizzano le relazioni statiche tra le singole parti del sistema ma l'attenzione si sofferma soprattutto sulla dinamicità delle relazioni dando maggior importanza all'interazione e alla coordinazione dei diversi

componenti.
I tradizionali paradigmi adottati dall'ingegneria del software (ad es. object-oriented) sono considerati non completamente adeguati per affrontare questi punti e quindi non adatti alla realizzazione di sistemi complessi. In particolare questi paradigmi non sono in grado di garantire, ad un giusto livello di astrazione, alcune indispensabili caratteristiche come la distribuzione del controllo e la dinamicità dei componenti.
Un adeguato livello di astrazione è invece garantito dal paradigma ad agenti e sistemi multi-agente che caratterizza molti dei campi più sviluppati nei giorni nostri, dall'ingegneria del software all'intelligenza artificiale distribuita.

2.1.1 Cosa sono e loro caratteristiche

Uno dei principali elementi necessari per comprendere il significato del paradigma ad agenti è quello di capire il concetto di agente.
An encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment in order to meet its design objective.
È questa una delle definizioni più diffuse nel mondo ingegneristico che fornisce, in maniera semplice e generale, un'idea di che cosa si intende per Agente: un'entità (computazionale) immersa in un ambiente con cui interagisce al fine di raggiungere un determinato obiettivo, svolgendo una o più attività [2].
Da questa definizione si possono notare le principali proprietà che caratterizzano l'agente:

- è situato in un particolare ambiente con cui interagisce: lo osserva, riceve informazioni sul suo stato (tramite sensori) e compie azioni su di esso (tramite attuatori);
- è creato per coprire un determinato ruolo al fine di raggiungere particolari obiettivi;
- è autonomo (intelligente) in quanto incapsula e controlla personalmente il suo stato interno e il proprio comportamento in modo tale da reagire, senza l'aiuto dell'utente, ad un ambiente dinamico ed imprevedibile;

- è flessibile e quindi capace di adattarsi ad ogni specifica situazione risolvendo qualsiasi tipo di problema che interferisce sul raggiungimento dell'obiettivo per cui è stato progettato;

- può sia agire conseguentemente ad uno stimolo esterno (reattività) sia prendere lui stesso l'iniziativa per soddisfare i propri obiettivi (proattività).

Vi sono inoltre altre importanti caratteristiche che un agente deve avere per raggiungere nel miglior modo possibile l'obiettivo preposto [11]:

- abilità a comunicare: gli agenti devono accedere ad informazioni provenienti da altre sorgenti (non necessariamente altri agenti, ma anche basi di dati, pagine web, persone, directory service, ecc.) circa lo stato di ambienti esterni ed essere in grado di scambiarsele;

- capacità di cooperazione: per eseguire processi complessi e perseguire obiettivi comuni gli agenti devono lavorare insieme;

- capacità di ragionamento: un agente deve possedere l'abilità di decidere sulla base della sua conoscenza ed esperienza;

- capacità di adattamento: l'agente deve avere dei meccanismi per giudicare lo stato attuale del suo dominio esterno ed incorporare questo all'interno di decisioni circa azioni future (tale caratteristica è nota anche come apprendimento);

- sincerità: l'utente e gli altri agenti devono essere sicuri che un dato agente agirà e riporterà in maniera del tutto sincera l'informazione a sua disposizione.

Sono pochi gli agenti che supportano tutte queste caratteristiche: la maggior parte sono specializzati per eseguire determinate azioni necessarie per perseguire l'obiettivo richiesto: a seconda di quali capacità posseggono, sono state definite diverse tipologie di agenti.

2.1.2 Tipologie e ambienti in cui operano

Un'interessante documento che definisce una buona classificazione delle diverse tipologie di agenti che si possono incontrare è sicuramente 'Software Agents: An Overview, Knowledge Engineering Review' di Nwana, H.S. Gli agenti possono essere *collaborativi* (Collaborative agents) cioè autonomi e con la capacità di cooperare con altri agenti; *d'interfaccia* (Interface agents) che hanno capacità di apprendimento e hanno il compito di gestire il dialogo con l'utente; *d'informazione* (Information agents) che gestiscono informazioni provenienti da varie fonti (ad es. Internet).
Un argomento di ricerca molto importante sono gli *agenti mobili* (Mobile Agents): utilizzando modelli computazionali classici (ad es. object-oriented) per definire il loro comportamento, sono in grado di spostarsi (migrare) da un computer all'altro (attraverso una rete) durante la loro stessa esecuzione.
Sebbene questi tipi di agenti forniscono importanti funzionalità utili per numerosissimi obiettivi, sono tre le tipologie di agenti più diffuse:

- *Agenti reattivi*
- *Agenti logici*
- *Agenti ibridi*

Gli agenti reattivi sono agenti dotati soltanto di reattività e non di proatttività: non possiedono una rappresentazione interna simbolica dell'ambiente esterno, ma agiscono solo secondo il principio richiesta-risposta. Questo significa che non hanno alcuna forma di rappresentazione della conoscenza ma ogni loro comportamento è esecuzione di un'azione in corrispondenza di una percezione. Sono semplici ed eleganti ma incapaci di affrontare problemi che richiedono la scelta di decisioni a partire da informazioni non recuperate da interazioni con l'ambiente.
Questa è invece una caratteristica molto importante degli agenti logici, agenti dotati di autonomia, capacità di collaborazioe e di apprendimento. Essi hanno una rappresentazione esplicita simbolica della conocenza in termini di teorie logiche e deduttive. Il loro comportamento non è una pura e semplice reazione a determinate percezioni

sull'ambiente, ma è guidato da ragionamenti e decisioni sulla conoscenza e sulle interazioni con il mondo esterno. Grazie alle informazioni possedute ed ad una certa capacità percettiva e attuativa, sono in grado di operare in maniera flessibile e razionale in un certo numero di circostanze ambientali: accessibili o non accessibili, statiche o dinamiche. Il più diffuso e adottato modello per questo tipo di agenti è il modello BDI (Belief Desire Intention): un agente è modellato in termini di conoscenze e credenze (Belief) che ha del mondo, di opzioni a disposizione per raggiungere gli obiettivi (Desire) e dei piani adottati per perseguire questi obiettivi (Intention) [12].
Ci sono inoltre agenti che combinano queste due tipologie: gli agenti ibridi. Sono costituiti da una parte deliberativa che contiene la conoscenza dell'ambiente esterno e una parte reattiva in grado di reagire ad eventi senza la necessità di ragionamenti complessi.

2.1.3 Agenti V.S. Oggetti

Gli agenti e i sistemi multi-agente stanno sviluppandosi praticamente in ogni settore della nostra società a discapito dei sistemi orientati agli oggetti che cositutivano uno dei più importanti traguardi raggiunti nel campo informatico.
La programmazione object-oriented è un paradigma di programmazione, che prevede di raggruppare in un'unica entità (la classe) sia le strutture dati che le procedure che operano su di esse, creando per l'appunto un oggetto software dotato di proprietà (dati) e metodi (procedure) che operano sui dati dell'oggetto stesso. Gli oggetti sono però molto differenti dal concetto di agente [4]. Innanzitutto gli oggetti sono entità passive: necessitano di avere un istruzione prima di diventare attivi; inoltre essi incapsulano stato e comportamento ma non il controllo: quest'ultimo fluisce da un oggetto ad un altro, con l'invocazione di metodi. Gli agenti invece, come abbiamo visto precedentemente, sono entità attive che interagiscono con l'ambiente senza alcun intervento esterno e che stabiliscono loro stessi quale azione eseguire in base a conoscenze interne della realtà. Non hanno interfacce in quanto violerebbero l'incapsulamento del controllo e sono dinamici e pronti a qualsiasi tipo di imprevisto: le pre-condizioni di una richiesta possono cambiare anche durante l'attività di soddisfacimento della richiesta

stessa.
Gli oggetti, le classi e i moduli non riescono a raggiungere, a differenza degli agenti, livelli di astrazione ottimali per realizzare e modellare sistemi sempre più complessi: infatti la granularità del comportamento e l'invocazione di metodi rappresentano un meccanismo troppo superficiale per descrivere i diversi tipi di interazioni tra queste entità.
Inoltre l'approccio object-oriented fornisce solo un minimo supporto per specificare e manipolare le relazioni sempre più complesse tra le varie parti che costituiscono il sistema.

2.2 Agenti e MAS: comunicazione, coordinazione e organizzazione

Gli agenti sono entità autonome che interagiscono con l'ambiente al fine di raggiungere i propri obiettivi. Da sole, però non riescono a rappresentare completamente le numerose prospettive e i numerosi obiettivi che un sistema complesso deve soddisfare. Proprio per questo si è passato dall'astrazione del singolo agente a sistemi con più agenti definiti *Multi-Agent System* (MAS).
I MAS sono insiemi di agenti con ruoli diversi che interagiscono in modo fruttuoso in un certo ambiente al fine di realizzare scopi del sistema nel suo complesso. Possono essere visti come veri e propri contesti in cui un agente vive, costituiti da risorse, strutture e processi che caratterizzano sia la singola entità sia la comunità degli agenti.
Un sistema multi-agente deve mantenere una coerenza globale senza un esplicito controllo centralizzato: gli agenti devono essere capaci di determinare per conto proprio gli obiettivi che condividono con altri agenti, i compiti comuni, evitare conflitti non necessari e mettere in comune la conoscenza. Per questo ciascuna parte del sistema lavora cooperativamente per raggiungere le funzionalità globali del loro sistema 'padre'.
Due degli aspetti centrali che caratterizzano i sistemi multi-agente sono l'organizzazione e la coordinazione: la prima riguarda la gestione della stuttura del sistema in termini di parti e relazioni tra le singole parti; la seconda invece descrive i processi che legano tra loro le singole entità, processi necessari per il raggiungimento degli obiettivi

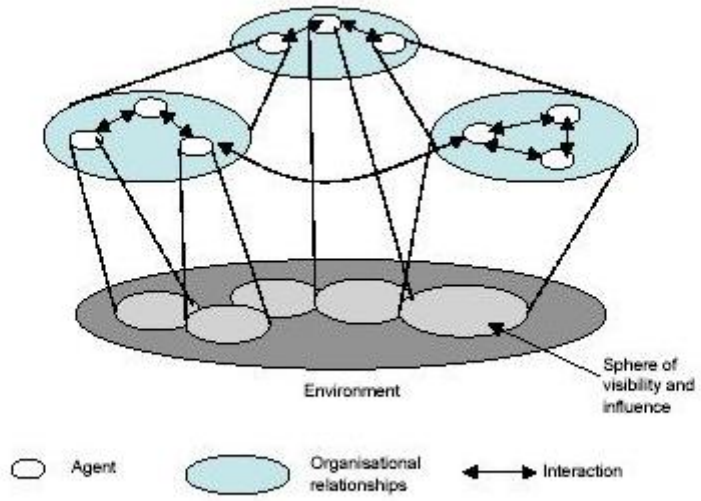

Figura 2.1: Rappresentazione di un sistema Multi-Agente

collettivi. Grazie ad un buon livello di coordinazione, gli agenti sincronizzano le rispettive attività e condividono e accedono alle risorse senza entrare in conflitto: questo permette di perseguire l'obiettivo del sistema senza incorrere in problemi inaspettati e diffici da risolvere.
La coordinazione è concepita da due punti di vista diversi. Da un lato essa è totalmente a carico dei singoli agenti e delle loro capacità di osservare e percepire il comportamento degli altri agenti e l'evoluzione dell'ambiente in cui operano; dall'altro non sono i singoli agenti a gestire la coordinazione ma opportune astrazioni di artefatti che osservano e regolano 'dall'esterno' il comportamento di tutti gli agenti e la loro interazione con l'ambiente. Seguendo questa seconda via, sono stati realizzate vere e proprie infrastrutture specializzate nel fornire servizi a supporto delle attività di coordinazione: condivisione delle risorse, gestione del flusso delle informazioni, sincronizzazione. Più avanti verrà analizzata e descritta uno delle più diffuse infrastrutture di coordinazione: l'infrastruttura Tucson.

Ma tutte queste infrastrutture forniscono anche servizi non solo per la coordinazione, ma anche per l'organizzazione, la sicurezza e soprattutto la comunicazione tra agenti.

2.2.1 Interazione/Comunicazione: diretta e mediata

L'interazione è la caratteristica principale dei sistemi multi-agente. Gli agenti necessitano di interagire tra loro e con l'ambiente esterno per eseguire i loro obiettivi individuali e per prendere parte ai processi collettivi che regolano la vita e lo sviluppo dell'intero MAS. Esistono differenti tipi di modelli di interazione adottati per garantire la comunicazione all'interno del sistema: possono essere tradizionali interazioni client-server o vere e proprie comunicazioni sociali caratterizzate da cooperazione, coordinazione e negoziazione tra i vari agenti. A parte questa distinzione, ci sono alcuni importanti punti che differenziano qualitativamente l'interazione tra agenti da quella presente in altri paradigmi dell'ingegneria del software. Nel caso degli agenti, questa avviene attraverso un linguaggio di comunicazione di alto livello basato su scambi diretti di messaggi che specificano azioni (KQLM e FIPA ACL)[bibliografia]: un agente esegue un atto di comunicazione al fine di modificare lo stato conoscitivo di un altro agente che lo ascolta. Ne segue che le interazioni sono realizzate secondo livelli di conoscenza a seconda di quali obiettivi devono essere seguiti, quando devono essere realizzati e da chi devono essere raggiunti. Inoltre, dal momento che gli agenti sono flessibili e sempre pronti a risolvere i problemi che si verificano all'interno dell'ambiente in cui operano, le diverse interazioni vengono manipolate con la stessa flessibilità. In questo modo, ciascu atto di comunicazione dipende dall'ambiente in cui viene eseguito e viene adattato secondo le caratteristiche del contesto in cui l'agente si trova.
L'interazione può avvenire in maniera diretta cioè attraverso scambi di messaggi direttamente tra gli agenti (si astrae dal mezzo che realizza la comunicazione); ma può avvenire anche in modo mediato attraverso artefatti, come lavagne e spazi di tuple, che rappresentano esplicitamente le astrazioni utilizzate dagli agenti per comunicare. Gli spazi delle tuple sono infatti spazi di informazione condivisi dove gli agenti

inseriscono e prelevano informazioni in modo associativo. Grazie ad essi un agente può comunicare con un altro anche se quest'ultimo non è presente: basta che l'agente mittente lasci il messaggio in questo spazio condiviso e l'entità interessata lo leggerà non appena cercherà un'informazione (a lui diretta) che abbia quel determinato template.
Questi artefatti sono supportati da apposite infrastrutture che forniscono, oltre a questo, altri servizi a disposizione degli agenti per supportare le loro attività: organizzazione, sicurezza, mobilità, coordinazione.
Nella successiva sezione si analizzerà la particolare infrastruttura Tucson necessaria per la realizzazione dell'obiettivo di questa tesi.

2.3 Infrastruttura di coordinazione Tucson

Come accennato precedentemente, gli agenti non interagiscono attraverso invocazioni di metodi che causano il passaggio del flusso di controllo da un'entità ad un'altra; essi utilizzano invece modelli di comunicazione e di coordinazioni basati sullo scambio di messaggi o diretto o tramite astrazioni condivise (spazi di tuple) Uno di questi modelli è TuCSoN (Tuple Centres Spread over Networks) [3].
TuCSoN è un'infrastruttura che fornisce servizi per la comunicazione e la coordinazione in sistemi multi-agente. Questi servizi sono realizzati utilizzando centri di tuple (Tuple Centre) distribuiti sui nodi dell'infrastruttura, dove gli agenti inseriscono, recuperano e leggono le informazioni sotto forma di tuple. Le tuple sono una collezione ordinata di pezzi (possibilemte eterogenei) di informazioni che vengono inseriti nei tuple centre e prelevati in modo associativo: questi centri permettono agli agenti di comunicare e coordinarsi tra loro.

2.3.1 Tuple Centre

I Tuple Centres (centri di tuple) sono spazi di tuple programmabili grazie ai quali gli agenti possono interagire secondo particolari comportamenti di coordinazione che possono essere programmati dinamicamente sia dall'utente sia dagli agenti stessi. I centri di tuple es-

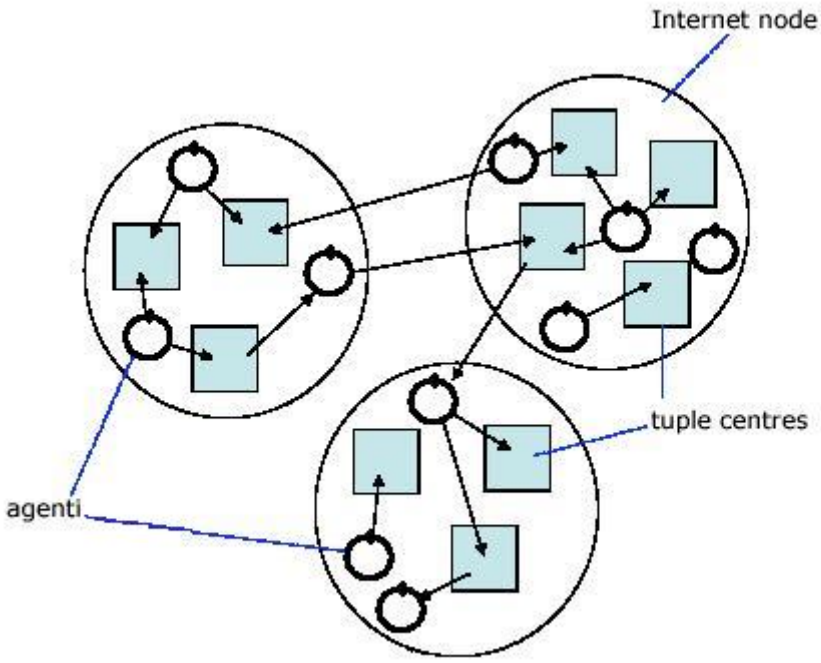

Figura 2.2: Vista logica di TuCSoN

tendono e specializzano gli spazi di tuple usando tuple logiche come linguaggio di comunicazione [9].
Gli spazi di tuple sono stati introdotti originariamente come un mezzo potente per realizzare la comunicazione e la coordinazione in sistemi concorrenti. I maggiori benefici forniti da questo particolare modello sono stati la separazione della comunicazione dalla coordinazione, la generative comunication e l'accesso associativo nello spazio di interazione. La prima caratteristica fornisce un'architettura ad agenti più pulita, ditinguendo chiaramente l'interazione dalla coordinazione. Grazie alla seconda, gli agenti comunicano creando e recuperando tuple la cui esistenza è totalmente indipendente dalle tuple generate da altri agenti. Questo permette di ottenere proprietà spaziali e temporali distinte in modo tale da semplificare la realizzazione di spazi di interazione per gli agenti. Infine l'accesso associativo alle informazioni

rende questi spazi più adatti per realizzare sistemi di gestione di informazioni dinamiche ed eterogenee.
Ma i tuple centres non sono semplici spazi di tuple: essi sono spazi estesi in cui il comportamento del centro è stato programmato dinamicamente in termini di transizioni di stato governate da determinati atti di comunicazione. A differenza degli spazi di tuple in cui tale comportamento è fissato una volta per tutte, nei tuple centre può essere manipolato e cambiato a seconda delle politiche di coordinazione richieste. In questo modo le leggi di coordinazione tra i vari agenti sono direttamente incapsulate all'interno dell'infrastruttura.
Il centro è programmato in modo tale da essere sempre pronto a reagire a qualsiati tipo di azione svolta dall'agente: grazie all'utilizzo di tuple logiche come linguaggio di comunicazione, ciascun agente è in grado di interagire con gli altri sincronizzando le proprie azioni e non interferendo con quelle delle altre entità.

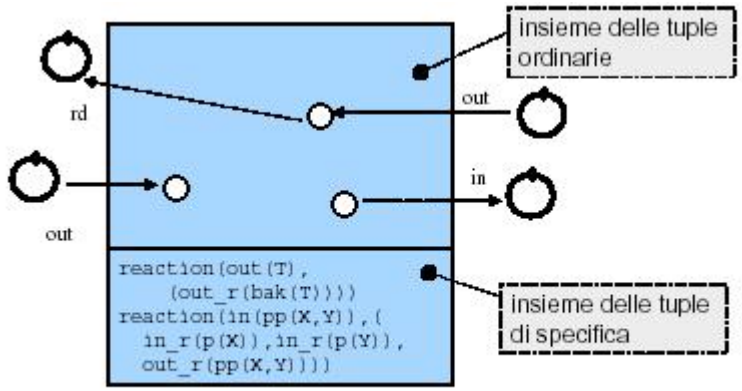

Figura 2.3: Rappresentazione del TupleCentre

Ciascun centro di tuple è associato ad un nodo ed è denotato localmente da un identificatore unico. Quest'ultimo è rappresentato da un nome che può essere assoluto o relativo. Più precisamente un tuple centre *tc* presente in un nodo Internet *node* può essere rappresentato o attaverso il nome assoluto *tc@node* da chiunque presente nella rete o attraverso il nome relatico *tc* nel contesto del nodo *node*. Ad esempio

access@lia.deis.unibo.it rappresenta il tiple centre *access* del nodo Internet *lia.deis.unibo.it*. Attraverso l'identificativo del tuple centre, gli agenti possono interagire con esso attraverso l'utilizzo di diversi tipi di operazioni. La forma di qualsiasi azione svolta dall'agente sul centro di tuple è *tc@node?operation(tuple)*: interagendo con il tuple centre *tc* nel nodo *node*, l'agente esegue l'operazione *operation* usando la tupla *tuple* [10].
Esistono diverse primitive di coordinazione (out, in, rd, inp, rdp) utilizzate come operazioni per inserire, leggere e rimuovere associativamente le tuple. La primitiva *out(T)* inserisce una tupla T nel Tuple Centre; *in(TT)* rimuove la prima tupla disponibile che fa match con il template TT e la consegna all'agente; *rd(TT)* legge la prima tupla disponibile che fa match con il template TT e la consegna all'agente senza rimuoverla dal tuple centre; *inp(TT)* rimuove la tupla con eventuale fallimento; *rdp(TT)* legge la tupla con eventuale fallimento.
Ciascuna di queste operazioni viene associata ad una determinata azione detta 'reaction' in modo tale da garantire vere e proprie leggi di coordinazione. Il linguaggio utilizzato per fornire al tuple centre queste regole di coordinazione è il ReSpecT che specifica quale deve essere il comportamento del tuple centre.

2.3.2 ReSpecT

I tuple centre sono spazi di tuple programmabili: il loro comportamento può essere programmato utilizzando una logica basata sul linguaggio ReSpecT che definisce vere e propire leggi di coordinazione.
Esso definisce un insieme di reazioni ciascuna delle quali specifica un insieme di azioni con cui manipolare l'insieme delle tuple. Una reazione è rappresentata nella forma:

reaction (Event, (Body))

Event è un evento comunicativo, che include l'esecuzione di primitive di coordinazione base (out, in, rd, inp, rdp) e la successiva esecuzione di alcune primitive ReSpecT (out_r, in_r, rd_r, no_r).
Body è una sequenza di predicati ReSpecT. I principali predicati sono out_r, in_r, rd_r, no_r: grazie a questi è possibile osservare e modificare il contenuto dell'insieme di tuple inserendo, leggendo e recuperando in modo associativo le tuple.

Out_r ha sempre successo e ha come effetto l'inserimento della tupla T nell'insieme delle tuple; *in_r* ha successo solo se la tupla che fa match con il template TT è presente nel centro di tuple e ha come effetto che la tupla viene rimossa e unificata con TT; se ciò non accade il predicato fallisce; *rd_r* ha successo solo se la tupla che fa match con il template TT è presente nel centro di tuple e ha come effetto che la tupla viene unificata con TT; se ciò non accade il predicato fallisce; *no_r* ha successo solo se la tupla che fa match con il template TT non è presente nel centro di tuple; se ciò non accade il predicato fallisce.
Ogni volta che si verifica un evento all'interno del tuple centre (ad esempio l'esecuzione di primitive di coordinazione da parte di un agente), viene eseguito il body di ciascuna reazione provocata da quel determinato evento. Le reazioni sono eseguite una dopo l'altra e l'ordine dell'esecuzione non è deterministico. Un semplice esempio di reaction è il seguente:

reaction(out(p(X)),(
out_r(backup(p(X)))))

In questo esempio, quando una tupla che fa match con il template p(X) viene inserita nel tuple centre, viene creata la tupla backup(p(X)) come copia di quella inserita.
L'esecuzione di un predicato (contenuto in Body) può aver successo o fallire causa il fallimento di una delle azioni delle primitive: ad esempio quando cerco di recuperare una determinata tupla con in_r e nessuna tupla con quel determinato template è presente nel tuple centre. In questo caso la reazione fallisce atomicamente cioè viene annullato ogni cambiamento effettuato sull'insieme di tuple dall'esecuzione del corpo della reazione: il contenuto del tuple centre viene riportato allo stato in cui si trovava prima dell'esecuzione della reazione.
Inoltre le reazioni possono essere anche scatenate dal successo dell'esecuzione di alcune primitive ReSpecT. Quando i predicati vengono eseguiti con successo, le reazioni ad essi associati vengono aggiunte all'insieme delle reazioni pronte per essere eseguite. Se una reazione fallisce, anche tutte le reazioni legate all'esecuzione dei suoi predicati vengono cancellate.

Capitolo 3

RSS syndication in TuCSoN

Dopo aver analizzato le due tecnologie necessarie per raggiungere l'obiettivo proposto, in questo capitolo illustrerò in che cosa consiste e in che modo è stata realizzata l'applicazione. Essa permette di manipolare e utilizzare un feed RSS in modo tale da informare il sistema ad agenti della presenza di una nuova notizia e di eseguire determinate azioni a seconda di come il sistema è stato progettato.

3.1 Analisi

L'obiettivo della tesi è quello di riuscire a integrare tecnologie Web-based che permettono di diffondere notizie e informazioni, a chiunque sia interessato, con sistemi distribuiti complessi, in cui viene utilizzato il paradigma ad agenti. In particolare, attraverso l'utilizzo dei feed RSS, ogni nuova informazione è disponibile su un determinato canale pronta per essere osservata e catturata. In questo modo, qualsiasi sistema automatico complesso ad agenti può osservare e recuperare questi eventi, ragionare su di essi e compiere determinate azioni. Comunicando e cooperando colletivamente, ciascun agente si comporterà nel modo più opportuno per raggiungere un certo insieme di obiettivi comuni all'intero sistema. Gli agenti agiranno a seconda di quale aggiornamento è stato osservato e segnalato e coordineranno automaticamente le loro azioni per garantire la realizzazione dell'obiettivo preposto.
Per informare l'intero sistema ad agenti della disponibilità di un nuo-

vo evento è necessario l'utilizzo di un agente in grado di recuperare le news aggiornate presenti in un feed RSS. Un feed RSS, come abbiamo visto, è un file XML, messo a disposizione da determinati siti, caratterizzato da particolari tag che forniscono una semplice descrizione delle nuove notizie. L'agente dovrà quindi collegarsi al canale messo a disposizione e, attraverso appositi strumenti per la gestione di documenti XML, 'catturare' gli elementi necessari (quali titolo, link e descrizione) per identificare e rintracciare le singole notizie. L'agente, per essere informato su news di diversa natura, può anche ricercare gli eventi su differenti canali: per questo è conveniente dotare l'applicazione della possibilità di scegliere l'indirizzo del canale desiderato in modo tale da avere svariate notizie su molteplici argomenti.
Gli altri agenti devono essere informati di queste notizie: l'agente manterrà le principali informazioni che descrivono ciascun evento e distribuirà queste informazioni, attraverso l'utilizzo di apposite tuple, all'interno di un tuple centre. In questo modo tutti gli altri agenti possono vedere quando è arrivata una nuova news e agire di conseguenza. È conveniente fornire la possibilità di stabilire il tuple centre desiderato, specificando il nome che lo identifica. L'agente sarà così in grado di fornire le informazioni anche a centri di tuple non presenti nel nodo locale in modo tale da rendere la notizia pubblica a chiunque.
Per rendere l'agente ancora più autonomo e indipendente, è possibile fare in modo che esso controlli il feed RSS ad intervalli di tempo. In questo modo l'utente non deve ogni volta avviare la sua esecuzione ma, una volta fatto partire, l'agente controllerà periodicamente se sono presenti nuovi aggiornamenti: l'utente, al massimo, deciderà l'intervallo di tempo tra due successivi controlli.
Anche se non indispensabile per il raggiungimento dell'obiettivo descritto, si potrebbe fornire l'applicazione di un'interfaccia grafica con cui l'utente possa interagire facilmente con l'agente e possa manipolare e modificare i principali elementi che lo caratterizzano come ad esempio l'indirizzo del canale del feed o il nome che identifica un particolare tuple centre.

3.2 Progetto

Il progettto consiste nella creazione di tutti i componenti necessari per la realizzazione dell'obiettivo descritto, secondo le osservazioni fatte durante la fase di analisi. Il linguaggio utilizzato per realizzare questi componenti è il linguaggio Java. Si è scelto di utilizzare questo linguaggio in quanto maggiormente conosciuto e diffuso nelle più importanti applicazioni informatiche. Inoltre, grazie alla presenza di classi e interfacce che costituiscono semplici template per agenti (Agent) e che interagiscono con l'infrastruttura TuCSoN (Tucson, TupleCentreID) utilizzata per la comunicazione e coordinazione degli agenti, la realizzazione dell'applicazione è risultata più semplice.
Poiché lo scopo è costruire una applicazione Java, l'architettura logica può essere immediatamente tradotta in una architettura di sistema associando ad ogni parte una classe che si assume la responsabilità di realizzare quella parte.
Con riferimento al pattern Model-View-Controller (MVC), si può dire che:

- Model: è rappresentato dall'agente che controlla e recupera le notizie
- View: è rappresentata dalla possibile interfaccia grafica che rappresenta il mezzo mediante il quale l'utente interagisce con il sistema
- Controller: è rappresentato dall'entità che trasforma le interazioni dell'utente attraverso la View in azioni eseguite dal Model

Dal momento che ciascun feed ha una struttura semplice da realizzare e manipolare, l'utilizzo di un unico agente in grado di decifrare i tag XML mi è sembrata la strada più semplice e più veloce per raggiungere l'obiettivo richiesto. Qualcuno potrebbe pensare che l'agente in questione debba sobbarcarsi un peso troppo gravoso e quindi sia realizzato in maniera molto complessa: oltre a interagire con il particolare TupleCentre e informare tutti gli altri agenti della presenza di nuove notizie, deve essere capace anche di poter leggere e decifrare gli elementi di un feed RSS. In realtà, la struttura di questi particolari file è semplice da capire, progettare e manipolare, e quindi l'utilizzo di un

unico agente è stata il punto di partenza per costruire l'applicazione. L'immagine che segue mostra, in maniera semplice e schematica, che cosa deve fare questa particolare entità

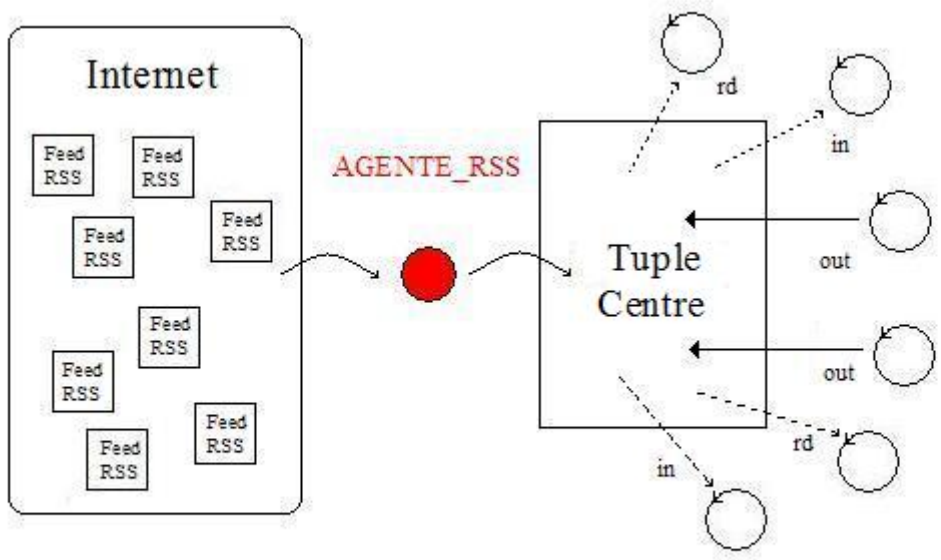

Figura 3.1: Rappresentazione dell'applicazione

L'agente in questione è realizzato attraverso l'utilizzo della classe *AgenteRSS*. Essa estende la classe *Agent* necessaria per la costruzione di semplici agenti TuCSoN: dotata di un proprio flusso di controllo, la classe base permette di accedere all'infrastruttura TuCSoN e di eseguire su di essa qualsiasi primitiva di coordinazione.
L'agente si collega all'indirizzo di un canale in cui è presente un determinato feed RSS e recupera da esso le notizie in esso contenute. Per compiere questo utilizza una particolare API, detta JDOM, adottata per gestire i documenti XML. Si è scelto di utilizzare questo particolare strumento in quanto fornisce meccanismi per recuperare tag XML più facili e immediati da capire in quanto coerenti con l'ottica degli sviluppatori Java. Infatti è possibile navigare la struttura di un

documento in maniera molto semplice, creando facilmente gli elementi che rappresentano i diversi tag incontrati e recuperando da essi i corrispondenti valori.
Una volta recuperate tutte le informazioni che descrivono ciascuna particolare notizia, viene creata una tupla che informerà il tuple centre e tutti gli altri agenti della presenza di un aggiornamento: da notare che sarà creata una tupla per ogni singola notizia e quindi per ogni singolo item presente nel feed RSS. Tale tupla è identificata da un nome univoco (rss_item) in modo tale che una volta inserita nel centro di tuple, gli altri agenti che interagiscono con questo tuple centre siano in grado di riconoscerla e di eseguire determinate azioni a seconda di come sono stati progettati.
Ciascuna tupla contiene il nome e l'indirizzo del canale seguiti dagli elementi principali che identificano la notizia: titolo, indirizzo e breve descrizione.
L'agente inoltre è realizzato in modo tale che, una volta lanciato, esso rimanga in esecuzione finché non viene stoppato dall'utente. In questo modo esegue, a intervalli di tempo regolari, il codice precedentemente illustrato: dopo aver controllato se all'interno del feed sono presenti nuove notizie, attende un certo periodo di tempo dopo il quale legge nuovamente il file RSS per vedere se sono stati inseriti nuovi aggiornamenti. Se è così, esso recupera solo le notizie che sono state inserite dopo l'ultima news recuperata nell'accesso precedente. Per realizzare questo, l'agente controlla la data di pubblicazione dell'ultimo item inserito (esso ha la data più recente tra tutte le notizie presenti nel feed): al successivo controllo, confronta tale data con quella del possibile nuovo ultimo item e considera solo le notizie più recenti rispetto a questa data, creando per ognuna di queste la tupla corrispondente. In questo modo si evita di creare ed inserire tuple che sono già presenti nel tuple centre.
L'applicazione è dotata inoltre di una semplice interfaccia grafica necessaria per permettere all'utente di settare i parametri indispensabili per l'esecuzione dell'agente. L'interfaccia è realizzata utilizzando le due classi *Console* e *MainPanel*: la prima costruisce e configura la finestra che contiene il pannello realizzato con la seconda classe. Il pannello contiene tutti gli elementi necessari per modificare i parametri utili per il funzionamento dell'agente e per avviare e interrompere la

sua esecuzione.
L'utente può decidere l'indirizzo del canale in cui è presente il feed RSS interessato e stabilire l'identificatore del tuple centre in cui inserire le tuple create. L'identificatore può essere semplicemente il nome del centro di tuple del nodo locale, ma può anche essere un nome assoluto che specifica, oltre al tuple centre, anche il nodo internet in cui esso si trova. L'utente può settare anche l'intervallo di tempo, espresso in millesecondi, che l'agente dovrà aspettare prima di ricontrollare il contenuto del feed.
Nell'interfaccia sono inoltre presenti due pulsanti per avviare ed interrompere l'esecuzione dell'agente: è stato utilizzato come listener per entrambi i pulsanti il pannello stesso. Questo in quanto i due pulsanti non devono rispettivamente far altro che creare e bloccare l'agente. Esso una volta avviato con il rispettivo pulsante, come prima operazione recupera i parametri settati dall'utente grazie all'interfaccia grafica: a seconda di quali valori sono stati specificati, esegue il controllo del feed RSS corrispondente e inserisce ciascuna tupla creata nel tuple centre specificato. L'utente può decidere di bloccare quando vuole l'agente utilizzando l'apposito pulsante presente nell'interfaccia: esso si bloccherà non appena avrà terminato il controllo completo dell'intero file RSS, evitando così di rimanere in sospeso tra una notizia e l'altra.
È presente inoltre un ultima classe *ApplicazioneRSS* che contiene solo il metodo main e che testa le tre classi descritte precedentemente.

Capitolo 4

Implementazione e testing

L'applicazione è costituita da quattro classi: *AgenteRSS*, *MainPanel*, *Console* e *ApplicazioneRSS*.
La classe *AgenteRSS* è quella principale: essa rappresenta l'agente che, una volta recuperati i parametri definiti dall'utente, si collega al feed RSS specificato, recupera tutte le informazioni relative a ciascuna nuova notizia in esso presente e crea una tupla per ogni evento trovato. Tale classe è collegata direttamente al pannello *MainPanel* da cui recupera l'indirizzo del feed RSS, l'identificativo del tuple centre e l'intervallo di tempo che l'agente deve aspettare prima di effettuare il successivo controllo del file considerato.
La seconda classe *MainPanel* (rappresenta la View del pattern MVC) rappresenta il pannello che ospita tutti i componenti necessari per permettere all'utente di interagire con l'agente. L'utente può modificare i parametri necessari per la sua esecuzione e avviare e bloccare l'agente stesso.
La terza classe *Console* rappresenta la finestra che viene visualizzata all'utente, contenente il pannello attraverso il quale si interagisce con l'agente.
L'ultima classe *ApplicazioneRSS* contiene il metodo main in cui vengono create le tre precedenti entità per testare l'esecuzione dell'applicazione.
Nelle pagine seguenti sono riportati gli interi codici delle classi appena descritte, commentate per rendere più veloce e chiara la comprensione del loro funzionamento.

Per testare la solidità dell'applicazione realizzata, si sono eseguiti diversi test che hanno permesso di verificare il corretto funzionamento delle classi create ma soprattutto l'esecuzione della classe *AgenteRSS*, fulcro dell'intero sistema.
Si sono scelti diversi indirizzi di feed RSS per vedere se l'agente considerava tutte le possibili notizie all'interno di questi file ed inseriva le corrispondenti tuple all'interno del tuple centre specificato. Si è provato a cambiare l'identificativo del centro di tuple, considerando sia nomi locali sia nomi di centri presenti su altri nodi Internet. Infine sono stati scelti diversi intervalli di tempo per vedere se l'agente aspettava quel determinato tempo prima di controllare nuovamente il feed RSS specificato.
Inoltre, grazie all'utilizzo del tool Inspector di TuCSoN che permette di monitorare e controllare lo stato di un tuple centre, si è potuto osservare direttamente l'inserimento delle tuple corrispondenti a ciascuna notizia recuperata dall'agente. Di seguito è possibile vedere l'interfaccia utente in cui sono stati specificati i parametri necessari all'agente: l'indirizzo è quello del feed RSS che contiene le notizie di sport del Resto del Carlino e il centro di tuple è quello di default del nodo locale. L'agente controlla se ci sono nuovi aggiornamenti ogni 5 minuti.

Figura 4.1: Interfaccia grafica

Dall'immagine che segue, che rappresenta il centro di tuple di default

del nodo locale, si possono vedere le tuple create e inserite dall'agente.

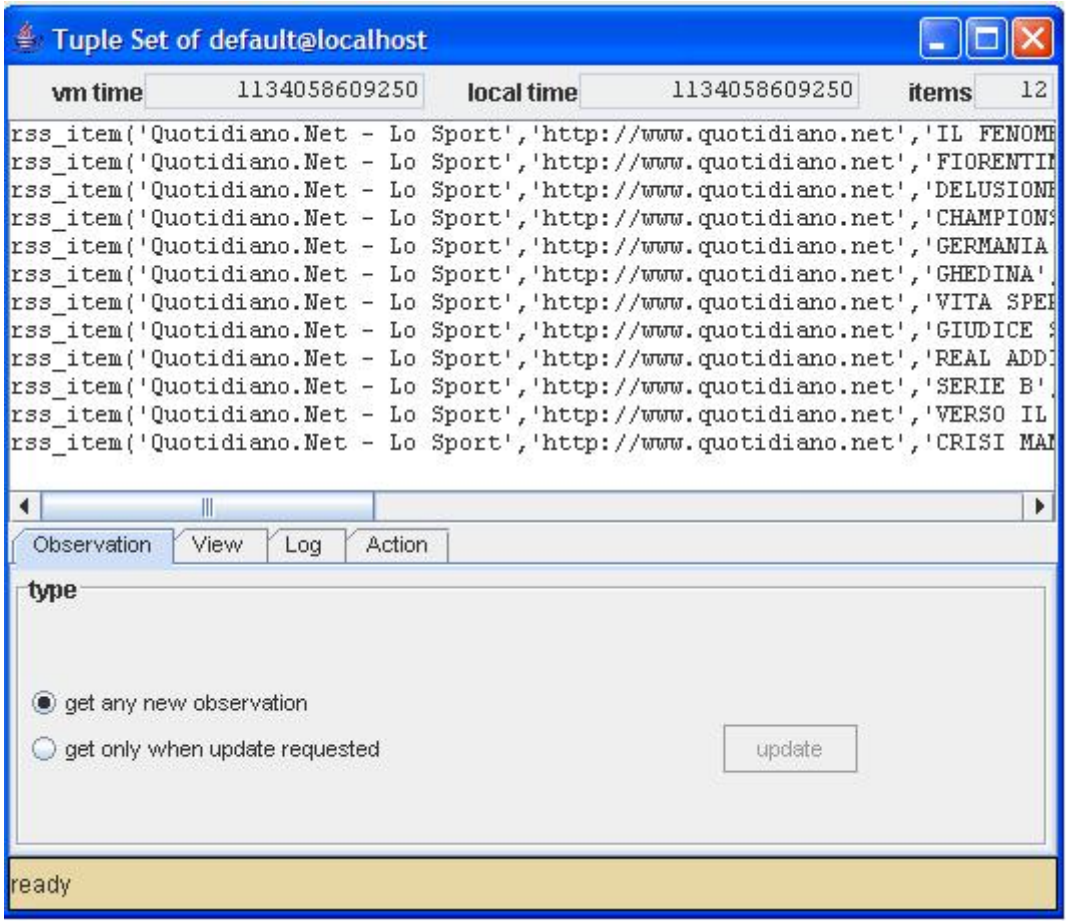

Figura 4.2: Il tuple centre default del nodo locale

```
import java.util.*;
import java.net.*;
import alice.tucson.api.*;
import alice.logictuple.*;
import org.jdom.*;
import org.jdom.input.SAXBuilder;

public class AgenteRSS extends Agent{

    private boolean finished = true;
    private String strurl;
    private String strtid;
    private long timeOut;
    private MainPanel prova;
    private AgentId id;
    private long firstItemTime;
    private long lastTime;
    private TupleCentreId tid;

    public AgenteRSS(AgentId id, String strurl, String strtid,
            long timeOut)throws Exception{
        super(id);
        this.strtid = strtid;
        this.strurl = strurl;
        this.timeOut = timeOut;
        //creo il TupleCentreId
        tid = new TupleCentreId(strtid);
        lastTime = 0;
    }

    protected synchronized void body(){
        //creo un documento JDOM fornendo l'URL
        SAXBuilder builder = new SAXBuilder();

        while (finished){
            try {
                //creo un oggetto URL
                URL url = new URL(strurl);
                Document doc = builder.build(url);
```

Figura 4.3: Codice della classe AgenteRSS

```
//Verifico la versione di RSS
Element rss = doc.getRootElement();
if(rss.getName() != "rss"){
    System.out.println("L'URL non e' un feed RSS 2.0");
}else{
    //Prendo il tag channel e i suoi elementi obblogatori
    Element channel = rss.getChild("channel");
    String title_channel = channel.getChildText("title");
    String link_channel = channel.getChildText("link");
    String descr_channel = channel.getChildText("description");

    //Prendo tutte le notizie
    List items = channel.getChildren("item");
    String title, link, description, date;
    Iterator iterator = items.iterator();

    if (iterator.hasNext()){
        //Recupero il valore della data e lo trasformo in millisec
        Element item = (Element)iterator.next();
        date = item.getChildText("pubDate");
        firstItemTime = manData(date);

        if (firstItemTime>lastTime) {
            insertItem(item,title_channel,link_channel);

            while(iterator.hasNext()) {
                //Recupero e trafsormo il valore della data
                item = (Element)iterator.next();
                date = item.getChildText("pubDate");
                long tempo = manData(date);

                if (tempo>lastTime) {
                    insertItem(item,title_channel,link_channel);
                } else {
                    break;
                }
            }//while
            lastTime = firstItemTime;
        }//if
    }//if
```

Figura 4.4: Codice della classe AgenteRSS

```
      }//else
      //l'agente aspetta l'intervallo di tempo specificato
      Thread.sleep(timeOut);
    } catch(Exception e) {
        System.err.println("Errore");
        e.printStackTrace();
    }
  }//while
}
//Blocca l'esecuzione dell'agente
public void stop(){
     finished = false;
}

private void insertItem(Element item, String title_channel,
        String link_channel)throws Exception {
    String title, link, description, date;
    title = item.getChildText("title");
    link = item.getChildText("link");
    description = item.getChildText("description");
    //inserisco per ogni item una tupla nel TupleCentre di
    //con il titolo del canale e i sottoelemti considerati
    out(tid, new LogicTuple("rss_item",new Value(title_channel),
            new Value(link_channel),new Value(title),
            new Value(link),new Value(description)));
    System.out.println("\tTitolo: " + title);
    System.out.println("\tLink: " + link);
    System.out.println("\tDescrizione: " + description);
}
//Controlla la data dell'item piu' recente
private long manData(String data){
    StringTokenizer tok = new StringTokenizer(data,",: ");
    tok.nextToken();
    int giorno = Integer.parseInt(tok.nextToken());
    String mesePar = tok.nextToken();
    int mese = 0;
    if(mesePar.equals("Jan"))
        mese = 0;
    if(mesePar.equals("Feb"))
        mese = 1;
```

Figura 4.5: Codice della classe AgenteRSS

```
        if(mesePar.equals("Mar"))
            mese = 2;
        if(mesePar.equals("Apr"))
            mese = 3;
        if(mesePar.equals("May"))
            mese = 4;
        if(mesePar.equals("Jun"))
            mese = 5;
        if(mesePar.equals("Jul"))
            mese = 6;
        if(mesePar.equals("Aug"))
            mese = 7;
        if(mesePar.equals("Sep"))
            mese = 8;
        if(mesePar.equals("Oct"))
            mese = 9;
        if(mesePar.equals("Nov"))
            mese = 10;
        if(mesePar.equals("Dec"))
            mese = 11;
        int anno = Integer.parseInt(tok.nextToken());
        int ore = Integer.parseInt(tok.nextToken());
        int minuti = Integer.parseInt(tok.nextToken());
        int sec = Integer.parseInt(tok.nextToken());
        Calendar calendar = Calendar.getInstance();
        calendar.set(anno, mese, giorno, ore, minuti, sec);
        calendar.set(Calendar.MILLISECOND,0);
        long numSec  = calendar.getTimeInMillis();
        return numSec;

    }
}
```

Figura 4.6: Codice della classe AgenteRSS

```
import java.awt.event.*;
import javax.swing.*;
import alice.tucson.api.*;

public class MainPanel extends JPanel implements ActionListener{

    private JLabel istr1, istr2, istr3;
    private JTextField indirizzo, tucsonId, intervallo;
    private JButton modifica, ferma;
    private AgenteRSS Agente;

    public MainPanel(){
        super();
        creazioniEntita();
        configura();
    }
    //Creo tutti gli elementi necessari all'utente per interagire con l'agente
    private void creazioniEntita() {
        istr1 = new JLabel("Indirizzo feed RSS:");
        istr2 = new JLabel("Identificatore TupleCentre" +
                           "(es. docs @ '192.0.0.3'):");
        istr3 = new JLabel("Intervallo di tempo tra le ricerche " +
                           "di nuovi aggiornamenti (millisec):");
        indirizzo = new JTextField(50);
        tucsonId = new JTextField(30);
        tucsonId.setText("default @ 'localhost'");
        intervallo = new JTextField(15);
        intervallo.setText("300000");
        modifica = new JButton("Avvia Agente RSS");
        ferma = new JButton("Interrompi Agente RSS");
        ferma.setEnabled(false);
        modifica.addActionListener(this);
        ferma.addActionListener(this);
    }
    //Aggiungo al pannello ciascun elemento
    private void configura(){
        add(istr1);
        add(indirizzo);
        add(istr2);
        add(tucsonId);
        add(istr3);
        add(intervallo);
        add(modifica);
        add(ferma);
    }
```

Figura 4.7: Codice della classe MainPanel

```
    //Fornisce l'indirizzo del feed specificato dall'utente
    private String getIndirizzo(){
        return indirizzo.getText();
    }
    //Fornisce l'identificatore del tuple centre specificato
    private String getId(){
        return tucsonId.getText();
    }
    //Fornisce l'intervallo di tempo di attesa dell'agente
    private long getIntervallo(){
        return Long.parseLong(intervallo.getText());
    }
    //Avvia e blocca l'esecuzione dell'agente
    public void actionPerformed(ActionEvent e) {
        AgentId aid = null;
        Object src = e.getSource();
        if (src == modifica){
            try {
                aid = new AgentId("rss");
                String ind = getIndirizzo();
                String id = getId();
                long interv = getIntervallo();
                Agente = new AgenteRSS(aid, ind, id, interv);
                Agente.spawn();
                ferma.setEnabled(true);
            }catch (Exception e1) {
                e1.printStackTrace();
            }
        }//if
        if (src == ferma)
            Agente.stop();
    }
}
```

Figura 4.8: Codice della classe MainPanel

```
import javax.swing.*;

public class Console extends JFrame{

    public Console(String titolo){
        super(titolo);
        configura();
    }
    //Configuro le dimensioni e la posizione della finestra
    private void configura() {
        setBounds(200,100,600,200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

Figura 4.9: Codice della classe Console

```
import javax.swing.*;
import java.awt.*;

public class ApplicazioneRSS {

    public static void main (String[] args) {

        JFrame Test = new Console ("Settaggio Parametri");
        Container c = Test.getContentPane();
        JPanel pannello = new MainPanel();
        c.add(pannello);
        Test.setVisible(true);
    }
}
```

Figura 4.10: Codice della classe ApplicazioneRSS

Capitolo 5

Conclusioni

L'utilizzo dei feed RSS sta diffondendosi velocemente tra tutti gli utenti desiderosi di recuperare e ricevere notizie via Web. Esso fornisce agli utenti la possibilità di accedere all'apposito canale che contiene il feed e osservare le singole notizie senza la necessità di dover navigare interi siti alla ricerca dell'evento desiderato.
Grazie all'utilizzo di sistemi automatici complessi è possibile osservare questi eventi e manipolarli secondo le proprie esigenze. L'utente non deve più recuperare direttamente le informazioni trovate ma sono utilizzate vere e proprie entità che gestiscono e catturano queste informazioni. Si possono così realizzare sistemi complessi di ogni tipo che recuperano i nuovi aggiornamenti e che, attraverso processi automatici di gestione, svolgono particolari attività a seconda della notizia osservata. I sistemi complessi realizzati attraverso il paradigma ad agenti stanno avendo molto successo grazie al comportamento autonomo delle singole entità decisionali che comunicano e si coordinano per raggiungere l'obiettivo globale. Ciascuna entità ha un particolare compito, necessario per il perseguimento dell'obiettivo dell'intero sistema.
In questa tesi sono stati integrati questi due concetti, la tecnologia RSS e il paradigma ad agenti. È stato realizzato un agente in grado di recuperare le notizie da un qualunque feed RSS e di creare, per ognuna di esse, una particolare tupla che viene inserita in un determinato tuple centre. L'applicazione è anche dotata di un interfaccia grafica attraverso la quale l'utente può modificare i parametri neces-

sari al corretto funzionamento dell'agente.
Allo stato attuale, il sistema funziona correttamente: esso è però solo un punto di partenza per la realizzazione di sistemi automatici più complessi. Ulteriori sviluppi futuri comprendono la programmazione delle altre entità che comunicano con il centro di tuple che viene avvertito della presenza di un nuovo evento: in questo modo ciascun agente, non appena viene inserita la tupla corrispondente ad una particolare notizia, è pronto per eseguire determinate azioni che porteranno al raggiungimento dell'obiettivo finale del sistema. Un sistema complesso di questo tipo potrebbe recuperare, attraverso l'agente realizzato, i titoli degli ultimi libri pubblicati e affidare ad altri agenti il compito di acquistare solo quelli di un determinato scrittore.

5.1 Ringraziamenti

Bibliografia

[1] S. Carletti. La guerra dei formati. RSS o Atom: questo è il problema, Aug. 2005.

[2] P. Copello. *Un sistema multi-agente per lo scheduling on-line in ambito manifatturiero*. PhD thesis, Università degli Studi di Genova, Facolta di Ingegneria, 2003.

[3] U. d. B. DEIS. *TuCSoN Guide*, Dec. 2004. TuCSoN version: 1.4.0.

[4] N. R. Jennings. Building complex software systems. *Comunication of th Act*, 44(4):35–41, Apr. 2001.

[5] C. Lamanna. Introduzione a rss. Web Writing, Mar. 2003.

[6] J. Lewin. An introduction to rss news feeds using openformats for content syndication. This article includes sample code that demonstrates elements of an RSS file, Nov. 2000.

[7] M. Nottingham. What is an rss channel, anyway? Syndication, Nov. 2002.

[8] M. Nottingham. Rss tutorial. Feature and benefits of RSS and a technical overview of it, Sept. 2005.

[9] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[10] A. Omicini and F. Zambonelli. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2:251–269, 1999.

[11] A. W. P. Maes. Software agents. Caratteristiche agenti, 1996.

[12] A. Ricci. *The Agent Paradigm*. PhD thesis, Università degli Studi di Bologna, Facolta di Ingegneria, 2003.

[13] A. Swartz. *RDF Site Summary (RSS) 1.0*.

[14] D. Winer. *RSS 2.0 Specification*.