TOOLKIT PER SISTEMI MULTIAGENTE

Francesco Di Giunta

Cos’è un toolkit per MAS?

E’ un sistema software che contiene:

- Un ambiente run time in cui ciascun agente può essere eseguito e in cui gli agenti possono interagire (“sistema operativo per MAS”).
- Degli strumenti CASE per il progetto e la costruzione di MAS.
- Degli strumenti per la gestione e il monitoraggio di MAS.

Molti toolkit esistenti

- IMPACT, consorzio interuniversitario, http://www.cs.umd.edu/projects/impact/
- JACK, Agent Oriented Software Group, http://www.agent-software.com/
- JADE, Telecom Italia Labs, http://jade.tilab.com/
- RETSINA, Carnagie Mellon, http://www.cs.cmu.edu/~softagents/retsina.html
- Zeus, British Telecommunications, http://labs.bt.com/projects/agents/zeus/
- …

Infrastruttura in cui un MAS può operare

- Perché i toolkit forniscono un'infrastruttura?
- Per la stessa ragione per cui esistono i sistemi operativi: evitare che il progettista di un MAS debba partire ogni volta da zero.
- In particolare: evitare che il progettista debba sviluppare le parti di un MAS specifico che hanno a che vedere con il concetto di MAS generico:
  - Architettura di un agente singolo generico
  - Modalità di interazione fra agenti (ad esempio il fatto che gli agenti comunicano tramite un ACL)

Infrastruttura in cui un MAS può operare

- Toolkit diversi suggeriscono o impongono differenti
  - architetture di agente singolo
  - architetture di MAS (quali interazioni permettere).
- Quindi: toolkit diversi per esigenze o preferenze progettuali diverse.

Architettura di un agente singolo indotta da un toolkit

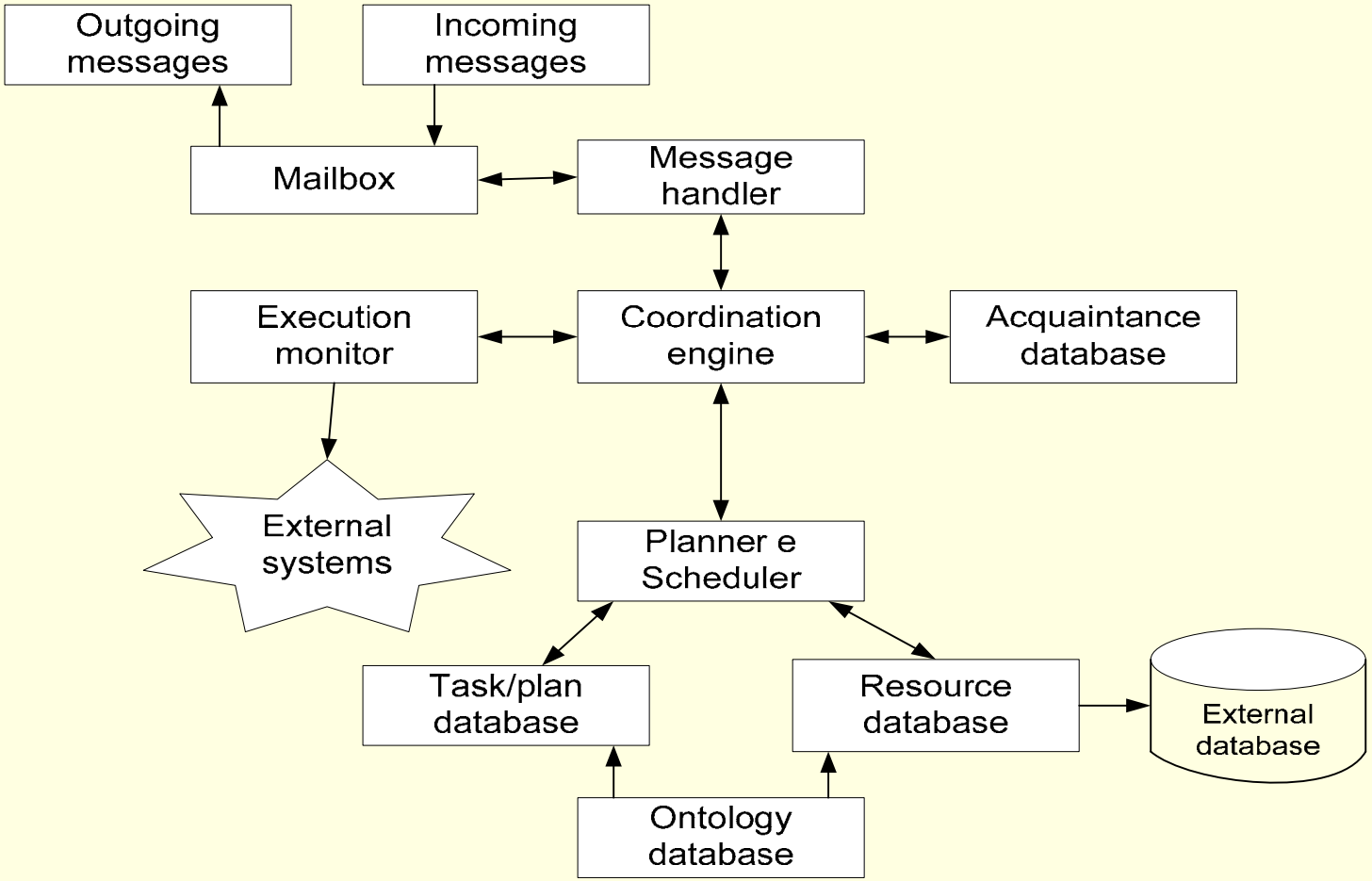
- Modelli “forti”: il toolkit ingloba una definizione rigida di agente ed il progettista deve solo definire i particolari strettamente dipendenti dall’applicazione (es.: Zeus):
  - V: facilità di progetto
  - V: soluzioni di provata affidabilità
  - S: adatti a certe applicazioni e non ad altre
  - S: scarsa flessibilità
- Modelli “deboli”: il toolkit ingloba una definizione “lasca” di agente; cosa è un agente è definito dal progettista dell’applicazione (es. JADE):
  - V: grande flessibilità
  - V: agenti per ogni tipo di applicazione
  - S: carico progettuale quasi interamente sul progettista dell’applicazione
  - S: è facile commettere errori

Architettura di un agente singolo indotta da un toolkit

- Agenti behavior directed (es. Jade)
- Agenti goal-directed + pianificazione (es. Zeus)
- Agenti BDI (es. JACK)
- Architetture ad hoc (es. IMPACT)
- Non mi risulta esistano architetture esplicitamente di tipo MDP (ma sono implementabili da architetture generiche).

Architettura di un agente singolo indotta da un toolkit

Esempio 1: architettura di un agente in Zeus:



Architettura di un agente singolo indotta da un toolkit

Esempio 2: architettura di un agente in JADE (cenni):

- Architettura generica behavior directed
  - Un behavior (o task) è formato da un'azione (codice generico) e da una condizione di terminazione.
  - Un agente ha vari behavior alcuni dei quali attivi, inseriti in una lista.
  - L'agente esegue l'azione del primo behavior attivo. Quando l'azione finisce si verifica la condizione di terminazione: se è vera il behavior viene disattivato. Se no viene rimesso in coda alla lista. Fatto questo, si procede col successivo behavior della lista.
  - I behavior possono essere attivati dall'azione di un altro behavior.

Servizi per MAS forniti dai toolkit

- Servizi di basso livello: generici per sistemi distribuiti
- Servizi di alto livello: specifici per MAS, sono principalmente servizi per l'interazione fra agenti
- Differenze fra toolkit:
  - Quali servizi fornire
  - Quali soluzioni adottare per fornire un servizio

Servizi di basso livello

- Divisione logico-fisica degli ambienti (per esempio i “container” in JADE)
- Meccanismi di comunicazione in rete, forniti da tutti i toolkits (es. TCP/IP in Zeus, RMI in JADE)
- Sicurezza (es. chiave pubblica, SSL)
- Load balancing, scaling, migrazione di agenti
- Replicazione di componenti
- Invio persistente di messaggi
- Interfacciamento con altri sistemi (es. DBMS)

Influenzano portabilità, prestazioni, affidabilità.

Servizi di alto livello

- Iscrizione e information discovery. Servono a notificare l'ingresso di un agente in un MAS e a far sì che ciascun agente sappia quali sono gli altri agenti presenti
- Meccanismi di comunicazione. Definiscono e implementano le modalità con cui gli agenti possono comunicare fra loro.
- Ontologie. Per la definizione dei termini usati nei messaggi.
- …

Servizi di alto livello: iscrizione e information discovery

- Modalità:
  - White pages (nome e indirizzo); JADE: AMS; Zeus: Name server
  - Yellow pages (nome, indirizzo, servizi offerti); JADE: Directory Facilitator; Zeus: Facilitator
  - Yellow pages con rilassamento (softmatching)
  - Linguaggi di descrizione di servizi e similarity matching (IMPACT)
- Iscrizione obbligatoria (JADE, Zeus) vs facoltativa (RETSINA)
- Ogni toolkit fornisce almeno un meccanismo di iscrizione/information discovery
- In genere implementati da particolari agenti previsti dal toolkit e non progettati dal progettista del MAS

Servizi di alto livello: meccanismi di comunicazione

- Linguaggi di comunicazione per agenti (ACL) basati sul paradigma degli atti comunicativi:
  - KQML
  - FIPA ACL
- Ogni toolkit ne mette a disposizione uno:
  - Quasi tutti FIPA ACL (JADE è del tutto FIPA compliant)
  - RETSINA usa KQML
- Un toolkit permette di definire i campi sender, receiver, performative, content,…
- L'ACL predefinito è l'UNICO meccanismo di comunicazione permesso
- Talora c'è supporto per i protocolli di comunicazione (es. protocolli FIPA in JADE) intesi come sequenze strutturate di messaggi

Servizi di alto livello: ontologie

- Strumenti per la definizione di un significato comune dei termini usati nei contenuti dei messaggi.
- Varie modalità:
  - Ontologie domain dependent (es. JADE, Zeus)
    - Il toolkit mette a disposizione strumenti per definirle
    - La definizione spetta al progettista di un MAS
  - Ontologie domain independent (es. RETSINA)
    - Sono messe a disposizione dal toolkit
    - Talora adatte a uno specifico MAS, talora no
  - Ontologie statiche: definite una volta per tutte col progetto del MAS
  - Ontologie dinamiche: estendibili man mano che nuovi agenti entrano nel sistema (es. IMPACT)

Servizi di alto livello: altri servizi

- Coordinamento e negoziazione:
  - Contract net (es. allocazione dei task in Zeus)
  - Offerte alternate
  - Vari protocolli d’asta
  - …
- Servizi per la mobilità “volontaria”
- Servizi per la creazione di team

Altre possibilità offerte dai toolkit

- Creazione di GUI
- Azioni concorrenti di un agente
- Integrazione con legacy systems
- Agentizzazione (wrappers)

Pesantezza di un sistema

- Le soluzioni adottate dai vari toolkit influenzano la “pesantezza” e, dunque, le prestazioni di un sistema
- Es.:
  - TCP/IP vs RMI
  - Crittografia e sicurezza
  - Planning
  - Ontologie complesse
  - Load balancing

Ambiti applicativi

- Un particolare toolkit, inglobando un particolare modello di MAS, è adatto a certi ambiti applicativi piuttosto che ad altri
- Modelli “deboli” (es. JADE): sono generici
- I modelli “forti” no. Es: Zeus è adatto a task oriented domains con agenti collaborativi
- Sono tutti adatti a MAS eterogenei (=con agenti diversi fra loro)
- Nessuno è veramente adatto a MAS eterogenei e aperti (= non si può prevedere quali agenti entreranno a tempo di esecuzione nel sistema). Ciò a causa della scarsa flessibilità, ad esempio nel riconoscimento dei servizi offerti e richiesti.

Strumenti per il progetto e la costruzione di un MAS

- Ogni toolkit fornisce strumenti per la costruzione di MAS che seguono l'architettura prevista e gireranno sull'infrastruttura fornita.
- Sono presenti tutte le sfumature:
  - Da semplici API
  - Ad ambienti di sviluppo grafici
- Quindi:
  - Dalla necessità di programmare quasi tutto
  - Alla possibilità di scegliere da menu le caratteristiche del proprio MAS
- In generale: maggiore è la flessibilità del modello di agente adottato, minore è la possibilità di un semplice sviluppo grafico.

Strumenti per il progetto e la costruzione di MAS

- Bisogna:
  - Definire l'ontologia usata nel MAS
  - Definire ciascun agente. Cosa esattamente definire dipende dall'architettura prevista dal toolkit; per esempio:
    - Azioni possibili
    - Risorse a disposizione
    - Protocolli di interazione conosciuti
    - Behavior
    - Goal
    - Funzioni d'utilità
    - …
  - Tenere presenti le interazioni che si avranno nel MAS
  - Implementare ciascun agente.
- Il MAS in sé non va implementato: esso è l'insieme degli agenti + i servizi per l'interazione forniti dal toolkit

Strumenti per il progetto e la costruzione di MAS

- Due casi estremi: JADE vs Zeus
- JADE:
  - Nessun supporto grafico
  - Semplici API messe a disposizione del progettista
  - Il comportamento di ciascun agente deve essere programmato in tutte le sue parti
- Zeus:
  - Supporto grafico per ogni aspetto del progetto, dalla creazione di ontologie alla creazione di agenti (da menu il progettista sceglie protocolli di coordinamento,…)
  - Generazione automatica di codice
  - Necessitano di programmazione solo i protocolli non standard e interfacce grafiche complesse

Metodologie di progetto

- Un toolkit, con il proprio modello di MAS ed i propri strumenti di progetto CASE, induce naturalmente una propria metodologia di progetto raccomandata
- Modelli deboli e strumenti di CASE non elaborati inducono metodologie deboli (o non inducono metodologie; es. JADE)
- Modelli forti e strumenti CASE elaborati inducono metodologie di progetto alquanto specifiche
- Le metodologie proposte sono spesso variazioni del paradigma di “role modelling”

Gestione e monitoraggio a tempo di esecuzione

- Tutti i toolkit prevedono strumenti per la gestione e il monitoraggio di MAS
- Scopi:
  - Debugging
  - Amministrazione di un MAS
- Si tratta generalmente di strumenti grafici, spesso implementati da agenti forniti dal toolkit.

Gestione e monitoraggio a tempo di esecuzione

- Prevede:
  - Lancio di nuovi agenti nel MAS
  - Rimozione di agenti dal MAS
  - Generazione di statistiche
  - Salvataggio della storia del MAS a scopo di documentazione
  - Uso dei dummy agent: agenti controllati direttamente dal progettista (ad esempio per l'invio di messaggi ad hoc a scopo di debugging)
  - Visualizzazione della struttura di un MAS:
    - Quali agenti sono presenti
    - Che servizi offrono
    - Quali canali comunicativi sono presenti
  - Monitoraggio dell'attività di un singolo agente:
    - Esecuzione dei piani
    - Stato di esecuzione dei behaviour
    - Raggiungimento dei goal
    - …
  - Monitoraggio delle comunicazioni fra agenti
  - …

Gestione e monitoraggio
Monitoraggio della comunicazione

- Elemento critico per capire il funzionamento del MAS nella sua globalità
- Vengono visualizzati i messaggi scambiati fra gli agenti
- Se ne occupano particolari agenti (lo Sniffer in JADE; il Society Viewer in Zeus)
- Due modalità di monitoraggio:
  - L'agente preposto legge i messaggi direttamente dalla piattaforma (es. in JADE)
  - L'agente preposto richiede agli altri agenti che gli notifichino i messaggi che inviano (es. in Zeus)

Conclusioni

- Molte offerte diverse
- Non c’è un toolkit fortemente dominante (pare ci sia una lieve prevalenza di JADE)
- Se c’è flessibilità c’è difficoltà di progetto
- Si è ancora lontani dal una ingegneria dei MAS consolidata
- I sistemi sono talora lenti
- Ci sono problemi di scalabilità: in MAS con molti agenti è difficile progettare, fare debugging, gestire
- Ci sono problemi per i sistemi aperti (è un limite della tecniche attualmente assestate, più che dei toolkits in se stessi)
- MA MOLTI SISTEMI REALI SONO STATI EFFETTIVAMENTE IMPLEMENTATI CON I TOOLKIT ESISTENTI