Un approccio basato ad agenti per sistemi di Autonomic Computing

Davide Fabbri

27 novembre 2009

1. Introduzione
2. Autonomic Computing
   - Definizione e obiettivi
   - Architettura
   - Aspetti avanzati
3. Approccio ad agenti
   - Artefatti CARTAGO come Manageability Endpoints
   - Agenti Jason come Autonomic Manager
   - MAS per Autonomic Computing
4. Conclusioni e sviluppi futuri
   - Sviluppi Futuri
   - Conclusioni

1 Introduzione

2 Autonomic Computing
- Definizione e obiettivi
- Architettura
- Aspetti avanzati

3 Approccio ad agenti
- Artefatti CARTAGO come Manageability Endpoints
- Agenti Jason come Autonomic Manager
- MAS per Autonomic Computing

4 Conclusioni e sviluppi futuri
- Sviluppi Futuri
- Conclusioni

La "Crisi della Complessità" I

Rapido Progresso Tecnologico

- Aumento della capacità computazionale dei sistemi hardware utilizzati.
- Aumento delle possibilità di comunicazione fra sotto-sistemi in termini di banda e modalità.

Sviluppo di sistemi informatici complessi

Gli sviluppatori realizzano sistemi caratterizzati da applicazioni:

- Multi-Threaded
- Distribuite
- Fortemente correlate ad altri sotto-sistemi, attraverso una fitta rete di relazioni

La "Crisi della Complessità" II

TCO: Total Cost of Ownership

Il costo totale di proprietà (TCO , definito da Gartner) per un sistema informatico è composto principalmente da:

Costi legati all'Hardware costo fisico, installazione, smantellamento; diminuzione dovuta allo sviluppo tecnologico.

Costi legati al Software principalmente dovuti da licenze, sviluppo e gestione applicativi; aumento dovuto a crescente complessità applicazioni, in particolare quelli relativi alla gestione.

Una possibile soluzione

Sfruttare la crescente capacità computazionale dei sistemi, per renderli in grado di auto gestirsi (*Self-Managing Systems*).

Autonomic Computing

Termine coniato da IBM[2] per definire sistemi informatici in grado di autogestire le risorse di cui essi sono composti.

"Autonomic" deriva dal comportamento del corpo umano, il quale è in grado per esempio di regolare battito cardiaco e respirazione senza il controllo diretto da parte dell'individuo.

Realizzando un sistema informatico dotato di capacità autonomiche, si permette ai professionisti IT dell'azienda di evitare di compiere operazioni di routine sul sistema, quali controllo dei parametri, analisi dei log, configurazione delle applicazioni, e di fare in modo che essi si possano dedicare a compiti di più alto livello.

Obiettivi

Vantaggi

Automazione selezione processi

Riduzione dei tempi

Riduzione delle capacità tecniche necessarie

Principali proprietà del sistema desiderate

Self-Protecting Protezione da attacchi esterni.

Self-Optimizing Ottimizzazione prestazioni e consumi.

Self-Healing Recupero di errori e ripristino.

Self-Configuring Configurazione del sistema.

Solitamente questo insieme di proprietà viene denominato attraverso l'appellativo *Self-**.

Visione d'insieme

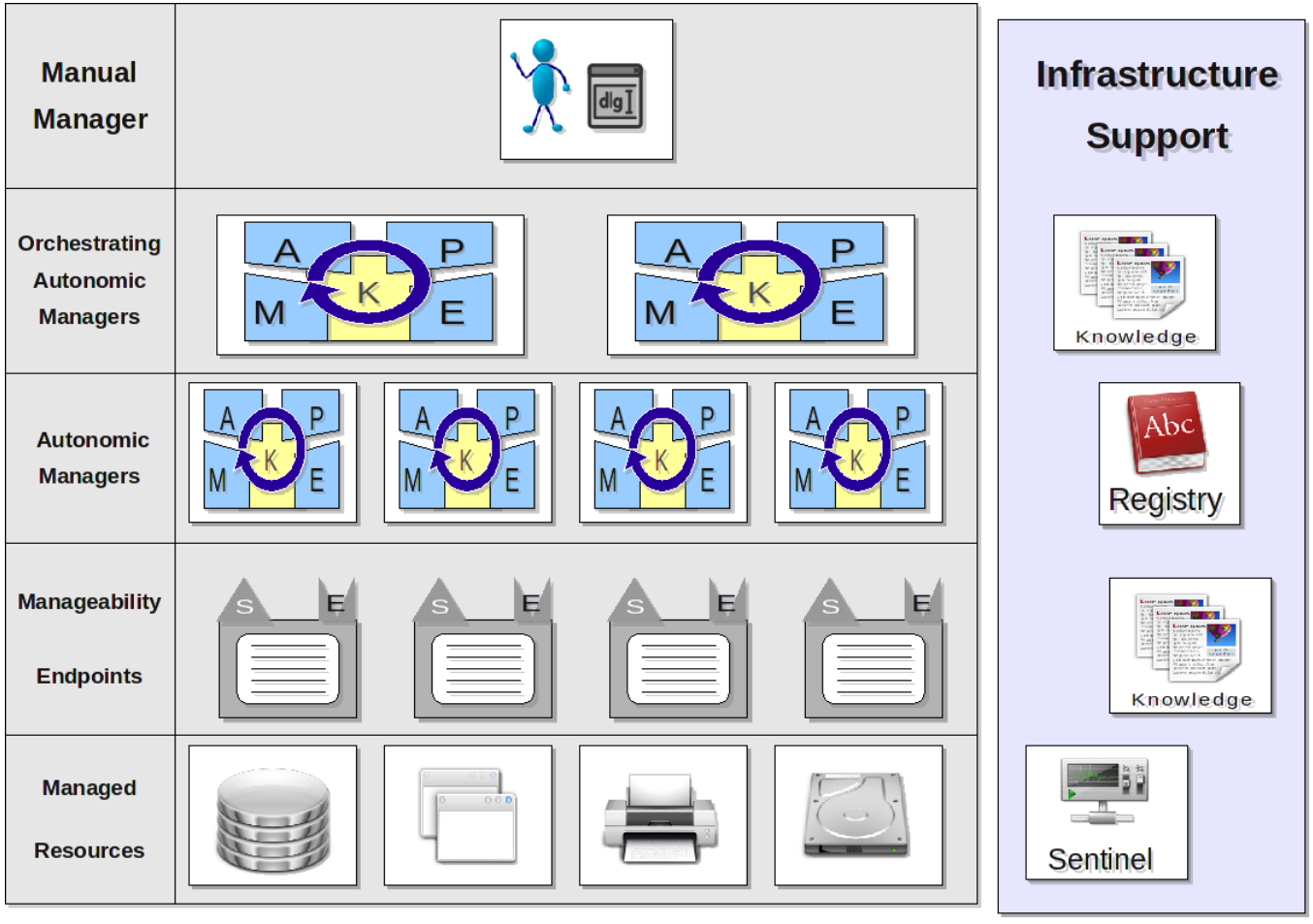


Figura: Architettura di un sistema autonomico

Componenti architetturali I

Autonomic Manager

Entità cui spetta il compito della gestione autonoma di una o più risorse del sistema.
Tale gestione viene effettuata attraverso un ciclo di controllo intelligente costituito da 4 fasi e definito MAPE-K:

M*onitor* **A***nalyze* **P***lan* **E***xecute*

Tale ciclo viene supportato dalle informazioni a disposizione dell'Autonomic Manager (**K***nowledge*).

Orchestrating Autonomic Manager

Autonomic Manager di livello più alto che non hanno il compito di gestire direttamente risorse del sistema, ma coordinano le attività autonomiche attuate nel sistema.

Componenti architetturali II

Manageability Endpoint

Entità passiva che media l'accesso alle risorse al sistema informatico da parte degli Autonomic Manager.
Essa rappresenta un'interfaccia alla risorsa che espone le operazioni utili ai fini della gestione della stessa.
Le operazioni esposte possono essere classificate in 2 categorie:

Sensors Ispezione dello stato della risorsa.

Effectors Modifica lo stato della risorsa o esecuzione di un'operazione da parte della risorsa.

Managed Resource (o IT Resource)

Risorse del sistema informatico di cui si ha intenzione di attuare una gestione autonoma.

Componenti architetturali III

Manual Manager

Interfaccia utente utilizzata dall'operatore per controllare e operare sul sistema informatico.

Supporto Infrastrutturale

In letteratura[5] vengono inserite nell'architettura di un sistema autonomico anche altre entità utili a supportare le attività dell'entità di base appena descritte.

- Knowledge Sources
- Registry
- Sentinel

Aspetti avanzati

Evoluzione sistemi di Autonomic Computing

Introdurre funzionalità autonomiche in un sistema informatico aziendale è un processo complesso per cui è necessario procedere per gradi. Ciò avviene principalmente su 2 dimensioni:

Funzionalità Complessità delle capacità autonomiche fornite ai componenti del sistema.

Scope Componenti di un sistema informatico dotati di tali capacità.

Standard per l'Autonomic Computing

Grande eterogeneità di risorse hardware e software $\longrightarrow$ Necessità di standard aperti per l'accesso a risorse della stessa classe.

WSDM Web Service Distributed Management

SDD Solution Deployment Descriptor

Outline

1. Introduzione
2. Autonomic Computing
   - Definizione e obiettivi
   - Architettura
   - Aspetti avanzati
3. Approccio ad agenti
   - Artefatti CARTAGO come Manageability Endpoints
   - Agenti Jason come Autonomic Manager
   - MAS per Autonomic Computing
4. Conclusioni e sviluppi futuri
   - Sviluppi Futuri
   - Conclusioni

Introduzione

- Un sistema autonomico è caratterizzato da molteplici Autonomic Manager che effettuano un ciclo di controllo intelligente sulle risorse che compongono il sistema.
- Il modello di programmazione object-oriented non prevede astrazioni di base che incapsulino un flusso di controllo al proprio interno, in grado di realizzare tale ciclo.
- Un possibile approccio per la realizzazione di sistemi autonomici è quello di utilizzare un modello di programmazione ad agenti i quali sono in grado di ovviare al suddetto problema.

Modello ad agenti adottato

- Meta-modello A&A[4].
- Agenti BDI implementati con Jason[1].
- Artefatti CARTAGO[3].

Manageability Endpoint → accesso alle risorse mediante:

- *Sensors*: accesso allo stato interno della risorsa, in CARTAGO modellata dall'insieme delle proprietà osservabili definite.
- *Effectors*: accesso alle operazioni sulla risorsa, in CARTAGO definite dall'interfaccia d'uso e dalle relative operazioni implementate.

Sensors

In CARTAGO realizzabili attraverso 2 primitive:

- `observeProperty(AId,PropName,?PropVal)`: osservazione puntuale di una proprietà (*One Shot Sensors*).
- `focus(AId,[SId],+Filt,?InitPropVal)`: osservazione continua di una o più proprietà (*Continuos Sensors*).

Effectors

Utilizzo della primitiva:

- `use(AId,OpName,UICtrlName(Params),[SId],+TOut,+Filt)`

Agenti BDI per Autonomic Computing I

- Diversi modelli di agenti sviluppati e studiati in letteratura.
- Quali motivazioni hanno spinto a scegliere un modello BDI (Jason in particolare)?

Ciclo di ragionamento Jason $\Leftrightarrow$ Ciclo MAPE-K di un Autonomic Manager

- I passi computazionali che compongono il ciclo di ragionamento di un agente Jason presentano affinità con i compiti delle singole fasi del ciclo MAPE-K di un Autonomic Manager.
- 1 Ciclo Jason $\neq$ 1 Ciclo MAPE-K.
- A ciascuna fase del ciclo di un Autonomic Manager possiamo associare i passi computazionali del ciclo di ragionamento Jason.

Agenti BDI per Autonomic Computing II

- Monitor
  - **1)** Percezione dell'ambiente.
  - **2)** Aggiornamento della base di conoscenza.
  - **3)** Ricezione di comunicazioni esterne.
- Analyze
  - **4)** Selezione dei messaggi.
  - **5)** Selezione di un evento.
  - **6)** Recupero dei piani rilevanti.
- Plan
  - **7)** Determinazione dei piani applicabili.
  - **8)** Selezione di un piano.
- Execute
  - **9)** Selezione dell'intenzione per l'esecuzione.
  - **10)** Esecuzione di un passo del piano istanziato.

Realizzazione di Autonomic Manager con Jason I

Approccio scelto

Livelli definiti in letteratura[2]: incremento graduale delle funzionalità autonomiche implementate.

Manuale

- Operazioni di gestione interamente affidate all'operatore di sistema
- Caso di studio non interessante, in quanto realizzabile con i classici modelli di programmazione object-oriented.

Realizzazione di Autonomic Manager con Jason II

Controllo

- Implementazione della fase di *Monitor* del ciclo MAPE-K.
- Esecuzione delle operazioni decise dall'operatore di sistema attraverso l'interfaccia utente.
- Necessarie 5 attività da parte dell'Autonomic Manager
  - Recupero riferimento artefatto Manageability Endpoint.
  - Recupero riferimento artefatto che consente l'interazione con l'utente.
  - Osservazione continua della risorsa mediante `focus` sull'artefatto Manageability Endpoint.
  - Rilevazione di particolari condizioni della risorsa e notifica all'interfaccia utente dell'operatore di sistema.
  - Rilevazione di istruzioni da parte dell'operatore di sistema ed esecuzione di esse sul Manageability Endpoint.

Realizzazione di Autonomic Manager con Jason III

Analisi

- Implementazione della fase di *Analyze* del ciclo MAPE-K.
- Ampliamento delle attività definite al livello precedente, in particolare relativamente alla modellazione dei piani dell'agente relativi alla rilevazioni di condizioni di interesse:
  - Inserimento di condizioni nel contesto del piano relative ad informazioni a disponizione dell'agente attraverso la propria Belief Base.
  - Inserimento di goal nel corpo del piano, aventi lo scopo di ottenere ulteriori informazioni dalla risorsa e dall'ambiente esterno (es. interrogazioni di basi di conoscenza condivise, interazioni con altri Autonomic Manager).

Realizzazione di Autonomic Manager con Jason IV

Ciclo di controllo

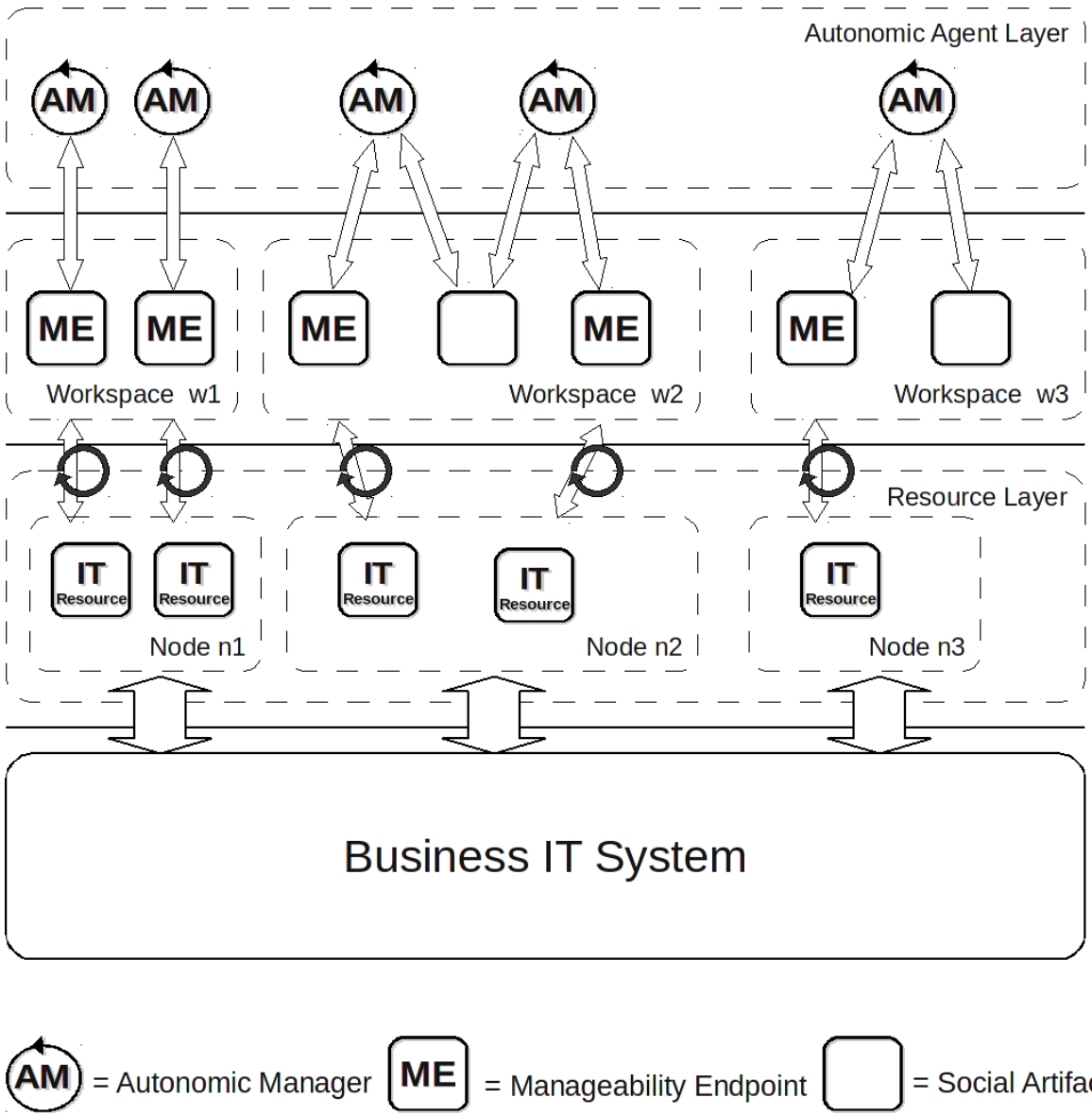
- Implementazione della fase di *Plan* del ciclo MAPE-K.
- Le operazioni emerse dal piano elaborato dovranno in ogni caso essere messe in atto dall'operatore di sistema.
- Costruzione di una libreria di piani per fare in modo che il Practical Reasoning dell'agente porti ad una possibile soluzione.
- Introduzione di politiche di business sulla base delle quali l'agente pianificherà le azioni da eseguire.
  - Definizione di un *Policy File* (XML-Based per esempio).
  - Recupero delle direttive attraverso un artefatto chiamato *Policy Artifact*.
  - Sulla base di tali direttive l'agente modifica la propria libreria di piani attraverso le azioni interne definite da Jason.

Realizzazione di Autonomic Manager con Jason V

Ciclo di controllo basato su politiche di business

- Integrazione di tutte le fasi in un ciclo completo MAPE-K
- Il piano elaborato viene messo in atto dall'Autonomic Manager sul Manageability Endpoint senza consenso diretto dell'operatore.
- L'operatore di sistema ha il compito di monitorare le attività svolte dagli Autonomic Manager ed eventualmente modificare le politiche di business attraverso i Policy File.
- Esecuzione di un piano elaborato dall'Autonomic Manager.
  - Memorizzazione in una lista ordinata delle operazioni da eseguire.
  - Esecuzione sequenziale (relativamente al piano istanziato) di esse fino al termine della lista.

Visione architetturale globale



1 Introduzione

2 Autonomic Computing
- Definizione e obiettivi
- Architettura
- Aspetti avanzati

3 Approccio ad agenti
- Artefatti CARTAGO come Manageability Endpoints
- Agenti Jason come Autonomic Manager
- MAS per Autonomic Computing

4 Conclusioni e sviluppi futuri
- Sviluppi Futuri
- Conclusioni

Sviluppi Futuri

- Studio delle attività di alto livello realizzabili attraverso Orchestrating Autonomic Manager
- Progettazione delle entità a supporto dell'infrastruttura, quali Registry, Sentinel e Knowledge Sources.
- Integrazione degli standard citati in letteratura, WSDM e SDD.
- Realizzazione di un layer infrastrutturale che permetta ai progettisti di astrarre alcuni concetti dell'Autonomic Computing.
- Misurazione dell'efficacia di tale approccio su sistemi reali
  - Simulazione di sistemi in scala ridotta.
  - Integrazione su sistemi reali di grandi dimensioni per testarne la scalabilità.

Conclusioni

- La soluzione proposta è ad un livello primordiale di sviluppo.
- Ampi margini di miglioramento e probabilmente vari aspetti trascurati.
- Necessaria una sperimentazione pratica su modelli simulati e reali per verificarne l'efficacia e l'efficienza.
- L'integrazione fra Jason-CARTAGO-GUI permette di effettuare tale sperimentazione in maniera agevole.
- Il modello di programmazione sembra essere in grado di fornire tutti gli strumenti per affrontare in maniera adeguata un problema rilevante come quello della "Crisi della complessità".

1. Introduzione
2. Autonomic Computing
   - Definizione e obiettivi
   - Architettura
   - Aspetti avanzati
3. Approccio ad agenti
   - Artefatti CARTAGO come Manageability Endpoints
   - Agenti Jason come Autonomic Manager
   - MAS per Autonomic Computing
4. Conclusioni e sviluppi futuri
   - Sviluppi Futuri
   - Conclusioni

Bibliografia I

Bordini, R. H., Hubner, J. F., and Wooldridge, M.
Programming Multi-Agent Systems in AgentSpeak using Jason.
Wiley, 2007.

IBM.
An architectural blueprint for autonomic computing.
IBM Research Grop on Autonomic Computing (2006), 1–36.

Omicini, A., Ricci, A., and Viroli, M.
Artifacts in the A&A meta-model for multi-agent systems.
Autonomous Agents and Multi-Agent Systems 17, 3 (Dec. 2008), 432–456.
Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Bibliografia II


Ricci, A., Viroli, M., and Cimadamore, M.
Prototyping concurrent systems with agents and artifacts: Framework and core calculus.
Electronic Notes in Theoretical Computer Science 194, 4 (Apr. 2008), 111–132.
6th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'07), CONCUR'07, Lisbon, Portugal, 8 Sept. 2007. Proceedings.


White, S. R., Hanson, J. E., Whalley, I., Chess, D. M., and Kephart, J. O.
An architectural approach to autonomic computing.
Autonomic Computing, International Conference on 0 (2004), 2–9.