

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SECONDA FACOLTÀ DI INGEGNERIA CON SEDE A CESENA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

**ARTEFATTI DI COORDINAZIONE
PER AGENTI IN SMART
ENVIRONMENT**

Elaborato in: Sistemi distribuiti LA

Relatore:

Prof. Andrea Omicini

Correlatore:

Ing. Elena Nardini

Presentata da:

Andrea Mordenti

Matricola:

0000284797

ANNO ACCADEMICO 2009-2010

SESSIONE I

PAROLE CHIAVE

Sistemi pervasivi & context-aware

Sistemi multi agente

Smart Environments

Tuple Centre

Artefatti di coordinazione

TuCSon

A mio nonno

INDICE

Introduzione	pag.	1
1 Sistemi Pervasivi		5
1.1 Verso il concetto di calcolo pervasivo		7
1.2 Il contesto		7
1.3 Context-aware computing systems		9
2 Smart Environment		11
2.1 Cos'è uno Smart Environment		11
2.2 Smart Devices		12
2.2.1 Sensori e attuatori		14
2.3 Automazione negli Smart Environments		15
2.4 L'approccio ad agenti		19
3 Agenti e Sistemi Multi Agente		21
3.1 Gli agenti		21
3.1.1 Proprietà		23
3.1.2 Tipologie		24
3.1.3 Apprendimento		26
3.1.4 Dagli oggetti agli agenti		27
3.2 Sistemi multi agente		29
3.2.1 Società di agenti		29
3.2.2 Comunicazione nella società		31

3.2.3	Ambiente	32
3.2.4	Progettazione di un MAS	33
4	Infrastruttura dell'ambiente MAS	35
4.1	Modello di coordinazione	37
4.1.1	Modello basato su tuple	39
4.1.2	Dagli spazi di tuple ai centri di tuple	40
4.2	Linguaggi di coordinazione e comunicazione	42
4.2.1	LINDA	42
4.2.2	ReSpecT	43
4.3	Verso la definizione di un nuovo meta modello	46
4.3.1	Tuple centre ReSpecT e MAS	46
4.3.2	Situatedness nei tuple centre	47
4.4	Il meta modello A&A (Agents & Artifacts)	48
4.4.1	Gli agenti	48
4.4.2	I workspace	49
4.4.3	Gli artefatti	50
4.4.3.1	Tipi	53
4.5	MAS nel meta modello A&A	55
4.6	Artefatti di coordinazione	56
5	Infrastruttura per gli artefatti di coordinazione	59
5.1	L'infrastruttura TuCSon	60
5.1.1	Coordination space	62
5.1.2	Coordination media	63
5.2	Artefatti di coordinazione in TuCSon	65

5.3	TOTA middleware	69
5.3.1	Approccio TOTA	69
5.3.2	Architettura	70
5.3.3	Perché TOTA?	74
6	Progettazione dello Smart Environment	77
7	Case study	87
	Conclusioni	95
	Ringraziamenti	107

Introduzione

Il costante progresso tecnologico nel mondo informatico sta seguendo un andamento esponenziale, caratterizzato da uno sviluppo rapido e costante.

La crescente evoluzione tecnologica, la miniaturizzazione dei dispositivi elettronici parallelamente alla loro esponenziale capacità di calcolo, alle sempre maggiori performance dei processori, alle interfacce reattive e adattabili a differenti situazioni, i minori costi, la disponibilità di infrastrutture di comunicazione wireless in aumento che permettono il collegamento di questi dispositivi in rete portano a considerare tali devices parte della nostra vita quotidiana ottenendo in questo modo una distribuzione della intelligenza su scala maggiormente ampia.

Attualmente il modello di computazione (detto *Paradigma desktop*) prevede una interazione diretta, in cui un utente individuale aziona *consapevolmente* una singola apparecchiatura per uno scopo specifico, ma in un futuro che non è poi così prossimo avremo a che fare con un grande numero di dispositivi di calcolo immersi nell'ambiente che ci circonda con i quali non saremo in grado di interagire direttamente, ma saranno essi a svolgere certe azioni al posto nostro senza che noi potremmo accorgercene.

Questa tendenza alla computazione proattiva ci consentirà di introdurre il concetto di *Pervasive computing* (calcolo pervasivo).

I sistemi caratterizzati da questo tipo di calcolo saranno trattati nella prima sezione dell'elaborato, nel quale verrà poi data una nozione di contesto che sarà l'elemento da cui dipenderanno le attività dei dispositivi, il quale è rappresentato in diversi tipi a seconda del tipo di ambiente che si intende controllare ed alle attività che vi si svolgono.

Questi piccoli dispositivi e le applicazioni che risiedono in essi dovranno essere capaci di sfruttare opportunamente la crescente disponibilità di interagire con l'ambiente che li circonda offrendo certi servizi come ad esempio la memorizzazione di informazioni fungendo da ponte verso altre risorse hardware/software; le attività svolte in modo parallelo dai diversi dispositivi saranno quindi autonome ma interdipendenti.

Oggetto di questa tesi sarà appunto analizzare l'ambiente composto da questi dispositivi che svolgono operazioni in background utili al raggiungimento di obiettivi sociali, studiando come avvengono le interazioni all'interno di questo.

Tale ambiente si prenderà carico di gran parte delle attività che prima d'ora erano svolte dai suoi abitanti migliorandone così la permanenza, viene così introdotto l'idea di un ambiente consapevole di ciò che bisogna fare in un qualsiasi tipo di contesto, un ambiente intelligente, uno *Smart Environment*.

I passi avanti fatti nel campo di aree informatiche come la robotica, i software agent-based, le reti di sensori ed il calcolo pervasivo hanno portato ad un conseguente aumento della richiesta sul mercato di progetti basati su Smart Environment: è per questi motivi che vengono analizzati nel seguito dell'elaborato tali sistemi.

La definizione di questi ambienti sarà oggetto del secondo capitolo della tesi nel quale verrà approfondito il comportamento di questi in risposta alle attività degli abitanti di tale sistema e quello che è il cosiddetto "ciclo di vita" all'interno dell'ambiente; successivamente si studieranno le caratteristiche degli elementi costitutivi, gli Smart Devices che svolgono attività computazionali indipendenti possedendo ottime capacità di I/O per poter interagire con altre entità o l'ambiente stesso.

Il terzo capitolo passa a esaminare le caratteristiche degli agenti –visti come entità immerse nell'ambiente costantemente in comunicazione tra loro, che agiscono in maniera coerente al contesto [1] – dove ne viene data una definizione e dove vengono illustrati i diversi tipi, successivamente vengono presentati i sistemi formati da questi, ovvero i MAS (Multi Agent Systems) che rappresentano la naturale astrazione per concepire e realizzare sistemi pervasivi.

Il capitolo centrale dell'elaborato tratta l'importanza che ricopre l'ambiente in un sistema multi agente, portandoci quindi alla questione fondamentale della gestione dell'interazione tra agenti di un MAS a carico appunto dell'ambiente per poter liberare gli agenti dal carico che comportano questi compiti.

Una delle questioni maggiormente critiche nella relazione tra agenti ed ambiente in un MAS è la cosiddetta *situatedness* che possiamo definire come: [2] *la proprietà degli agenti di essere strettamente accoppiati con l'ambiente esterno, tipicamente espressa in termini di abilità nel reagire in maniera appropriata ai cambiamenti ambientali.*

Sezione presente nel medesimo capitolo, e di fondamentale importanza, riguarda la presentazione del meta modello A&A (Agents & Artifacts) per i sistemi multi agente, il quale regola l'interazione tra agenti ed ambiente delegando la situatedness agli artefatti. Di queste entità, gli artefatti, verrà data una definizione mostrando anche le loro caratteristiche ed i vari tipi esistenti.

Si andrà poi a definire quello che è il *modello di coordinazione* più opportuno per poter regolare l'interazione tra gli agenti che cooperano nel sistema: analizzando le proprietà dei vari tipi di modelli conosciuti, risulterà che il modo più spontaneo per modellare la comunicazione tra agenti di un MAS all'interno di uno Smart Environment è quello di uno spazio di comunicazione condiviso, e quindi tramite i cosiddetti *Tuple Centre* – Tuple Space il cui comportamento non è definito a priori e può essere esteso in accordo con le necessità applicative [3] – scritti in linguaggio ReSpecT, anch'esso appositamente esteso per gestire gli eventi ambientali.

Si vede come, combinando il meta modello A&A con i centri di tuple ReSpecT, nel sistema che sta alla base di un ambiente intelligente un adeguato artefatto di coordinazione possa essere assunto come il medium coordinativo delle interazioni tra le entità del sistema.

Nella sezione successiva si definisce l'infrastruttura che supporterà la comunicazione e la coordinazione nella società di agenti ed artefatti distribuiti tramite tuple centre, ovvero TuCSon [4] sviluppata dal DEIS (Dipartimento di Elettronica Informatica e Statistica); ne verranno presentate le caratteristiche, le utilità e gli aspetti che maggiormente interessano il campo di discussione dell'elaborato.

Il capitolo riguardante le tecnologie software alla base di uno Smart Environment si chiuderà poi con la presentazione del middleware TOTA, adottato per poter supportare la scalabilità, la dinamicità della rete e la mobilità degli elementi che compongono il sistema.

Una volta assemblate tutte le infrastrutture, tecnologie e modelli verrà fatta una panoramica generale di come questa tesi è arrivata a definire strutturalmente uno Smart Environment partendo dalla descrizione logica del proprio comportamento, fino ad arrivare al vero e proprio framework applicativo.

Concludendo si andrà ad concretizzare brevemente il sistema ottenuto applicandolo ad un semplice caso di studio

Capitolo 1

Sistemi pervasivi

1.1 Verso il concetto di calcolo pervasivo

I notevoli passi avanti nella miniaturizzazione dei componenti elettronici permette ai dispositivi di calcolo con varie capacità ed interfacce di diventare parte della vita di tutti i giorni grazie anche ai prezzi abbordabili che hanno sul mercato.

I processori sempre più piccoli, leggeri, economici e mobili sono presenti ovunque nell'ambiente, negli oggetti di tutti i giorni e spesso anche nel nostro corpo.

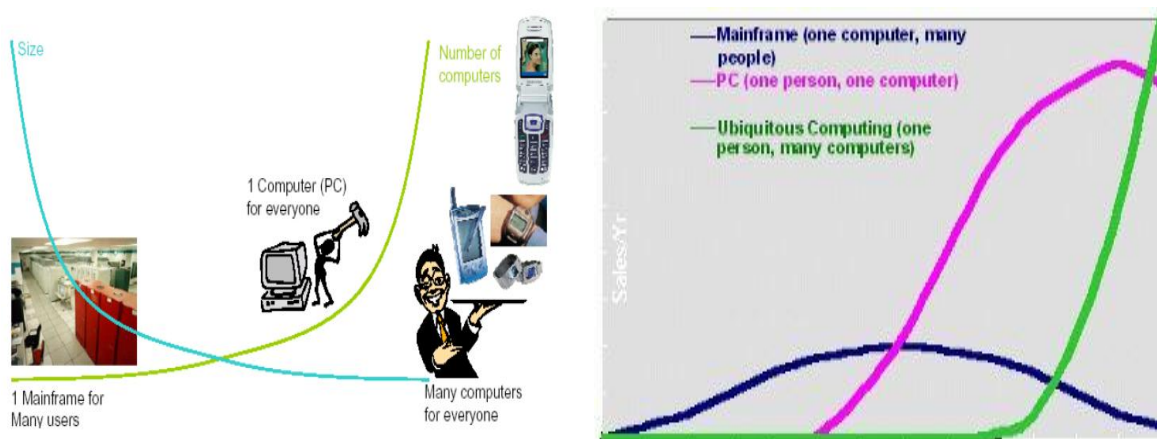


Figura 1.1 : Si mostra come, col tempo, i dispositivi di calcolo diventano con il tempo, più piccoli e più disponibili agli utenti, dunque maggiormente diffusi.

Esistono una miriade di esempi per quanto riguarda i dispositivi tecnologici ottenuti grazie alle nuove nanotecnologie: dai PDA (Personal Digital Assistants) alle polveri intelligenti (Smart Dust), ovvero microscopici sistemi elettronici tra loro in comunicazione capaci di rilevare luce, temperatura oppure vibrazioni nel luogo in cui vengono “sparsi”, apparecchi acustici direttamente inseriti all’interno dell’orecchio che catturano le vibrazioni e le riproducono amplificate, passando per gli ultimi eBook Reader (iPad).

Si riesce quindi a dare un’idea di ciò che vuol dire una unità di calcolo che si fonde con un qualsiasi oggetto di utilizzo quotidiano, estendendo e migliorando quelle che erano le sue precedenti funzionalità.

Il primo a fornire una nozione di calcolo pervasivo od *ubiquitous* fu Mark Weiser, che già agli inizi degli anni ’90 con uno dei suoi articoli esprime la sua intuizione riguardo della esponenziale crescita tecnologica.

“Le tecnologie più profonde sono quelle che scompaiono. Esse sono cucite nel tessuto della vita quotidiana fino a quando non diventano indistinguibili da essa” [5].

Un esempio banale di una tecnologia che è per così dire “scomparsa” si può fare anche in campo non strettamente legato all’informatica, prendiamo ad esempio gli occhiali da vista: chi è abituato a portarli ne trae ovviamente un grande vantaggio per le attività che svolge tutti i giorni, ma arriva al punto che non si accorge nemmeno di portarli. Stesso discorso può essere fatto per il volante della nostra auto, è lo strumento che ci permette di compiere il percorso desiderato, ma nessuno di noi mentre sta per svoltare ad una curva pensa “bene, ora girerò il volante a destra per far sì che le ruote facciano sì che l’auto vada verso destra ...”. La vera utilità della tecnologia anche per quanto riguarda l’ambito informatico è dunque questa, la possibilità che un utente azioni diversi sistemi e apparecchiature di calcolo in modo simultaneo mentre svolge la normali attività senza che esso se ne renda effettivamente conto di queste azioni simultanee che avvengono in background.

Questo paradigma è descritto come *pervasive computing* (noto anche come *ubiquitous computing*).

La *pervasive computing* descrive il concetto di calcolo integrato nell’ambiente invece che nei computer (intese come unità computazionali distinte), dalla integrazione di diverse unità di calcolo nell’ambiente circostante, la speranza è quella di avere utenti che

interagiscono con questi componenti in maniera più naturale di come già fanno [6].

Naturalmente l'idea di calcolo pervasivo non può essere riassunta in una unica definizione, piuttosto si possono classificare quelle che sono le proprietà fondamentali in modo da potere definire differenti tipi di sistemi ed applicazioni.

Ora ci troviamo a che fare con un modello *post-desktop* dove l'elaborazione delle informazioni è stata interamente integrata all'interno degli oggetti di tutti i giorni:

lo scenario della computazione pervasiva prevede la diffusa presenza di applicazioni tecnologiche distribuite nell'ambiente ed interagenti mediante reti (cablate o wireless).

L'intero sistema dovrà così essere disposto ad attivare e maneggiare un insieme di attività autonome ma interdipendenti e correlate che sono eseguite da entità distinte sia sulla base del *contesto* in cui ci si trova sia in base a quali sono le attività e gli obiettivi dell'entità stessa.

1.2 Il contesto

In più di una occasione durante le sezioni precedenti è stato fatto riferimento al termine *contesto*, considerandolo come uno degli aspetti che influenzano le azioni compiute dai componenti del sistema.

Dopo decenni di ricerca nel campo della gestione del *contesto* uno svariato numero di definizioni sono state formulate, tuttavia è possibile riportare le due definizioni classiche, che più frequentemente sono riportate negli articoli dedicati all'argomento.

In [7] vengono identificati tre aspetti importanti del *contesto* che sono:

- *Dove ti trovi*
- *Con chi sei*
- *Quali risorse si trovano nei dintorni*

in questo modo è possibile classificare il sistema a seconda di dove si trova un utente, dell'insieme di altri utenti che si trovano nelle vicinanze

come sorta di contesto sociale e della collezione di apparecchiature e reti di calcolo.

Le modifiche che possono verificarsi nel tempo per questi tre aspetti ricoprono un ruolo di importanza fondamentale nella capacità di adattamento per i sistemi context aware.

In [8] Dey definisce il contesto come una qualsiasi raccolta di informazioni che sia in grado di caratterizzare la situazione di una entità in un dato momento, dove per entità intenderemo qualsiasi soggetto appartenente al sistema in esame, una persona, un edificio, un dispositivo di calcolo, etc. che sia considerato di importanza rilevante nell'interazione tra utente ed applicazione, dove la rilevanza di una certa entità è determinata dalle attività che devono essere svolte dall'utente.

Come si vede nella figura sotto le applicazioni attingono le informazioni utili per conoscere lo stato del contesto in cui ci si trova per poter ragionare e compiere così le azioni maggiormente utili.

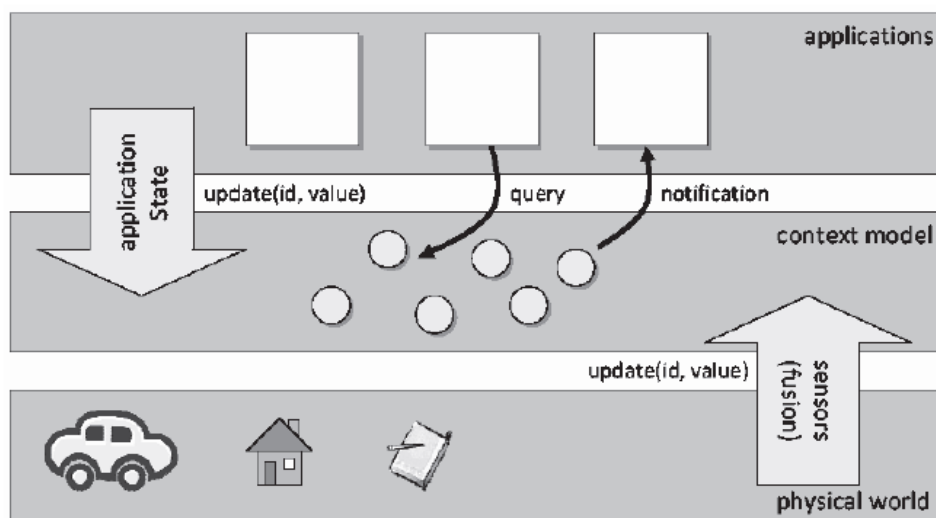


Figura 1.2: rappresentazione grossolana di quello che è un generico modello di gestione del contesto

Il contesto dunque è rappresentato da un insieme di frammenti di informazioni ambientali che vengono recuperate da una serie di dispositivi (sensori) immersi in tale ambiente.

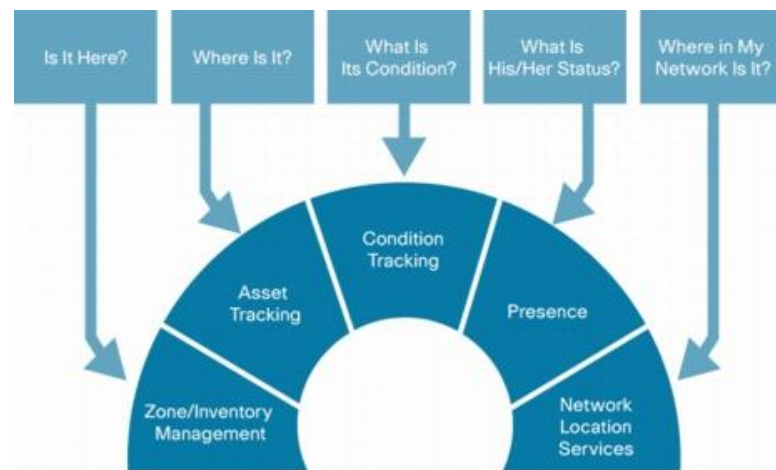


Figura 1.3: domande che ci si pone per avere informazioni sul contesto.

1.3 Context-aware computing systems

Il calcolo sensibile al contesto si riferisce al paradigma nel quale il modo di agire di ogni componente è determinato dalle circostanze in cui esso si trova.

I sistemi di questo tipo usano il contesto per migliorare la capacità delle applicazioni e dei dispositivi: essi adattano automaticamente sé stessi al contesto percepito, cambiando il loro comportamento.

Anche in questo caso sono presenti numerose definizioni per un sistema context aware, distinguibili però in due categorie: sistemi che *usano il contesto* e sistemi che *si adattano al contesto*.

Il primo definisce l'abilità di un sistema di entità di percepire, interpretare il contesto per poter così rispondere agli aspetti riguardanti la localizzazione di un utente nell'ambiente e a quelli riguardanti i dispositivi [9], il secondo ci dice che il sistema –sempre considerato come l'insieme delle entità che lo compongono– modifica le proprie caratteristiche in linea con quelle che sono le caratteristiche del contesto percepito, in modo da massimizzare la flessibilità dei servizi offerti [10]; dunque, i context-aware system hanno come obiettivo quello di fornire servizi sensibili al contesto in cui ci si trova.

In [11] vengono studiati diversi scenari di tipici sistemi context-aware e i diversi tipi di contesto con cui questi sistemi si trovano ad avere a che

fare; nella tabella sotto riportata a diversi tipi di sistemi si associa il tipo di contesto che dovrà essere osservato dalle entità.

System Name	System Description	Context Type			
		A	I	L	T
Classroom 2000	Capture of a classroom lecture			X	X
Cyberguide	Tour guide		X	X	
Teleport	Teleporting	X	X	X	
Stick-e Documents	Tour guide		X	X	X
	Paging and reminders	X	X		
Reactive Room	Intelligent control of audiovisuals	X	X	X	
GUIDE	Tour guide			X	
CyberDesk	Automatic integration of user services	X			
Conference Assistant	Conference capture and tour guide	X	X	X	X
Responsive Office	Office environment control			X	X
NETMAN	Network maintenance			X	
Fieldwork	Fieldwork data collection			X	X
Augment-able Reality	Virtual post-it notes			X	
Context Toolkit	In/Out Board		X	X	X
	Capture of serendipitous meetings		X	X	X
Active Badge	Call forwarding		X	X	

Figura 1.4: tipi di sistemi e relativi tipi di contesti osservabili

I vari tipi di contesto sono: Attività (per le quali un'entità è stata progettata), Identità (caratteristiche dell'entità, suo stato interno), Locazione (dove si trova, quali risorse le sono vicine), Tempo (l'importanza del periodo temporale nel contesto è di notevole importanza per certi sistemi).

Si potranno quindi avere diversi tipi di contesto –e quindi diverse proprietà che lo caratterizzano– a seconda dello scenario nel quale viene applicato uno Smart Environment.

Capitolo 2

Smart Environment

2.1 Cos'è uno Smart Environment

Con l'introduzione della nozione di calcolo pervasivo e sensibile al contesto sono state gettate le basi per poter definire ed esprimere le caratteristiche di quelli che sono chiamati *Smart Environment* ovvero ambienti “intelligenti”, che saranno il tema centrale dell'elaborato.

La progettazione degli Smart Environment è un obiettivo che fa appello ai ricercatori di una varietà di discipline tra cui: l'intelligenza artificiale, la pervasive & mobile computing, la robotica, middleware, software basati sugli agenti, il calcolo multimediale e le reti di sensori.

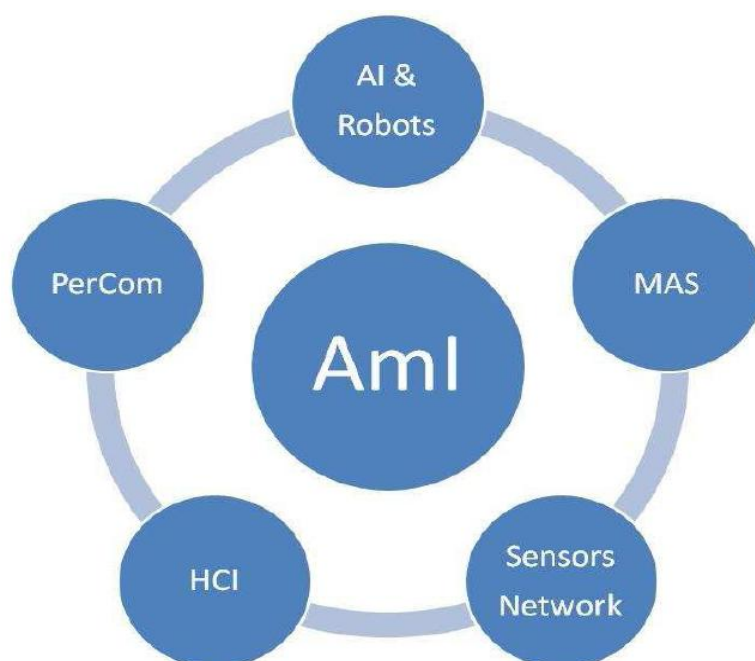


Figura 2.1: l'intelligenza ambientale e le aree scientifiche a cui è relazionata

Non esiste una enunciazione condivisa universalmente per poter definire questi tipi di ambienti, ne saranno citate alcune per poterne dare una visione del tutto generale.

Una prima interpretazione fu data da Mark Weiser, definendoli come *un mondo fisico il quale è abbondantemente ed invisibilmente integrato con sensori, attuatori, visori, display ed altre entità computazionali tessute alla perfezione in oggetti della vita quotidiana e connessi tra loro*, in [12] un ambiente è smart quando ha *l'abilità di acquisire ed applicare le conoscenze che possiede dell'ambiente esterno e dei suoi abitanti per poter così migliorarne la loro esperienza all'interno di esso*, in [13] un ambiente digitale che *proattivamente ma sensibilmente, supporta le persone nella loro vita di tutti i giorni*.

Quindi l'interazione tra utenti umani ed un sistema pervasivo può essere idealizzato come uno Smart Environment, il quale fornisce ai suoi abitanti servizi computazionali adeguati per le loro attività, in relazione al contesto. Tali servizi devono quindi essere adatti ad ogni attività degli utenti e devono essere forniti da un gruppo di dispositivi che lavorano cooperativamente, le cui attività devono essere coordinate in modo intelligente, queste unità computazionali sono chiamate *Smart Devices*.

2.2 Smart Devices

Essendo un paradigma costruito sulla computazione pervasiva e l'interazione "human-centric" sarà progettato e caratterizzato da sistemi e tecnologie che sono:

- *Integrate (embeddeed)*: più dispositivi, connessi tra loro, sono integrati nell'ambiente come se fossero una parte di esso
- *Sensibili al contesto (context-aware)*: usano il contesto in ogni istante per fornire informazioni e servizi agli utenti in dipendenza delle attività svolte
- *Personalizzabili*: devono poter essere progettati su misura in base alle esigenze

- *Adattative*: devono poter cambiare in risposta alle variazioni che possono avvenire all'interno dell'ambiente
- *Anticipatrici*: possono anticipare i (probabili) futuri desideri degli utenti senza che vi sia una conscia interazione con essi

Tra queste tecnologie vengono chiamati Smart Devices quei dispositivi che possiedono la capacità di sfruttare l'interazione con gli altri dispositivi dell'ambiente al fine di fornire servizi sensibili al contesto in cui ci si trova e per supportare forme avanzate di comunicazione tra gli utenti.

Questi dispositivi intelligenti devono essere tra loro collegati, in una rete (wireless o meno) in modo distribuito e devono essere trasparentemente accessibili in modo che l'interazione utente-dispositivo sia nascosta: è infatti uno degli obiettivi degli Smart Environment ridurre l'esplicita interazione utente-macchina (HCI).

Sono progettati per supportare un insieme di proprietà relative al calcolo pervasivo, oltre che diverse caratteristiche strutturali –data l'eterogeneità tipica di questi sistemi– per poter essere usati in qualsiasi tipo di ambiente intelligente, con la possibilità di poter accedere ed eseguire servizi remoti così come a specifici ambienti esterni.

Devono essere sensibili al contesto dell'ambiente in cui ci si trova per poter quindi ottimizzare le proprie attività, in più operano in maniera autonoma, in modo autogovernato senza che sia necessario l'intervento dell'utente.

Essendo gli ambienti intelligenti una classe di sistemi distribuiti sappiamo che i dispositivi che sono presenti in uno Smart Environment oltre che eterogenei sono spesso sono mobili –ovvero possono cambiare continuamente la loro locazione– essi possono essere caratterizzati come [14]:

- Dispositivi di ICT “multi-purpose” come PDA, smart phone, video camere...
- Personalizzati e configurati ad uno specifico proprietario, il quale ha la possibilità di utilizzarli in modo sicuro
- Possono essere strettamente legati all'utente come nel caso dei “wearable devices” oppure in modo meno rigoroso come i cellulari
- Hanno un supporto multi funzionale per poter facilitare l'accesso e l'interoperabilità a run-time
- Possono combinare le loro applicazioni interne con più servizi esterni, ai quali accede tipicamente via una WLAN remota

In alcune applicazioni certi tipi di Smart Device sono spesso tradizionali apparecchiature elettroniche dotate di capacità di calcolo aggiuntive in modo che esse possano “pubblicizzare” il loro stato, il loro profilo nell’ambiente, facendo così in modo che gli utenti o altri dispositivi che sono localizzati nell’ambiente in cui si trovano possano valutare se ritengono adeguata quella certa apparecchiatura per l’attività che stanno svolgendo.

2.2.1 Sensori ed attuatori

Gli Smart Device che maggiormente interessano gli argomenti trattati nello svolgimento dell’elaborato sono i sensori e gli attuatori.

I *sensori* sono quei dispositivi fisici il cui compito è quello di percepire l’ambiente, ovvero di catturare e misurare le proprietà che caratterizzano lo stato dell’ambiente in una certa situazione, ovvero che rappresentano il contesto.

Properties	Measurand
Physical properties	pressure, temperature, humidity, flow
Motion properties	position, velocity, angular velocity, acceleration
Contact properties	strain, force, torque, slip, vibration
Presence	tactile/contact, proximity, distance/range, motion
Biochemical	biochemical agents
Identification	personal features, personal ID

Figura 2.2: tipi di misurazione per le principali proprietà che vengono catturate dai sensori

Naturalmente i sensori non sono tutti uguali, essi formano una rete di dispositivi eterogenei interagenti ognuno dei quali è stato progettato per poter catturare un diverso aspetto del contesto ambientale.

Sotto vengono mostrati, per i diversi tipi di contesto, una serie di sensori progettati per poterne valutare quantitativamente i relativi valori.

Type of context	Available Sensors
Light	Photodiodes, color sensors, IR and UV-sensors etc.
Visual Context	Various cameras
Audio	Microphones
Motion, Acceleration	Mercury switches, angular sensors, accelerometers, motion detectors, magnetic fields
Location	Outdoor: Global Positioning System (GPS), Global System for Mobile Communications (GSM); Indoor: Active Badge system, etc.
Touch	Touch sensors implemented in mobile devices
Temperature	Thermometers
Physical attributes	Biosensors to measure skin resistance, blood pressure

Figura 2.3: tipi di contesto e sensori per la misurazione ad esso associati.

Gli *attuatori* invece sono quelle apparecchiature fisiche che permettono allo Smart Environment di poter attuare le proprie azioni andando a modificare lo stato (della totalità oppure di una parte) dell'ambiente esterno, sulla base di quelli che sono state le informazioni ambientali rilevate in precedenza dai sensori.

2.3 Automazione negli Smart Environment

Ogni Smart Environment possiede un certo grado di autonomia, è in grado di modificare i propri attributi in base ai cambiamenti dell'ambiente e comunica con gli utenti umani in maniera naturale, per poter fare questo esso deve favorire il controllo remoto dei dispositivi e la comunicazione tra loro per l'acquisizione/diffusione di informazioni attraverso questi; in più

deve poter prendere decisioni in maniera predittiva basandosi su esperienze passate e/o sui risultati finali da raggiungere.

Tutto questo è necessario quindi che avvenga in maniera continua ed ininterrotta.

L'automazione in questi tipi di ambienti può essere vista come un ciclo continuo caratterizzato da una iniziale percezione dello stato ad opera del livello più basso del sistema, ovvero il livello fisico composto dalla totalità dei dispositivi che circondano l'ambiente.

La percezione dell'ambiente è visto come un procedimento *bottom up* ovvero ogni singolo dispositivo conosce il valore di una certa proprietà ambientale per il quale è stato programmato, riferito ad una certa porzione ambientale; tutte queste conoscenze raccolte dalla rete di sensori vengono assemblate per formare quello che sarà il contesto dall'ambiente (un certo contesto per esempio può contenere informazioni riguardo a temperatura, umidità, tempo, movimento, etc.).

La rete di sensori che raccoglie le informazioni ambientali in modo continuo ed instrada tali informazioni attraverso una certa infrastruttura di comunicazione e di rete verso gli strati superiori del sistema i quali si preoccupano di elaborare i dati e prendere le conseguenti decisioni in base anche alle attività che si stanno svolgendo all'interno dell'ambiente.

Al contrario della percezione dell'ambiente, l'applicazione delle azioni prese è un procedimento *top down* dove il flusso di esecuzione scende di nuovo verso i livelli più bassi dove l'azione viene poi suddivisa (se necessario) fra i diversi attuatori o controllori presenti, che sono in grado di cambiare lo stato dell'ambiente.

Una vecchia visione di un sistema intelligente è quella riportata nella figura sotto [15]:

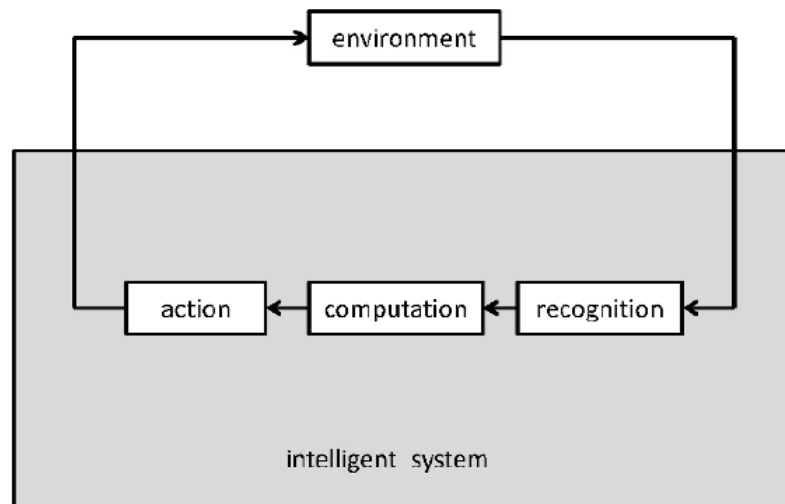


Figura 2.4: vecchia visione di un sistema intelligente

Con il passare del tempo, l'esperienza ha portato a considerare l'ambiente come la sola chiave del comportamento intelligente e quindi lo si considera parte integrante del sistema.

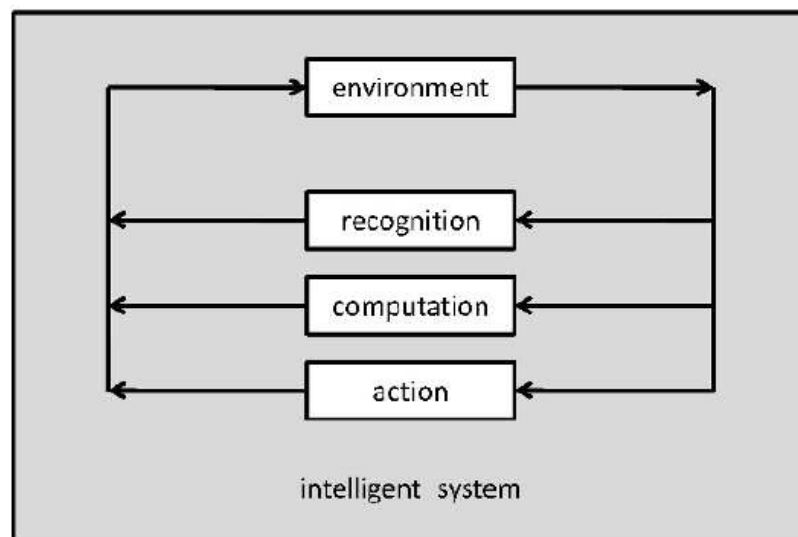


Figura 2.5: nuova visione di un sistema intelligente

Questa nuova visione dell'intelligenza confersisce una maggiore importanza all'ambiente in un ambito più ampio: non ci si limita ad un agente per area, ma si fa riferimento ad una squadra di agenti all'interno dell'ambiente così che le operazioni di percezione ed elaborazione possano essere eseguite in modo parallelo.

Le figure sotto riportate esplicitano quello che è il ciclo che compie uno Smart Environment per potersi adattare alle informazioni ambientali acquisite dai propri dispositivi.

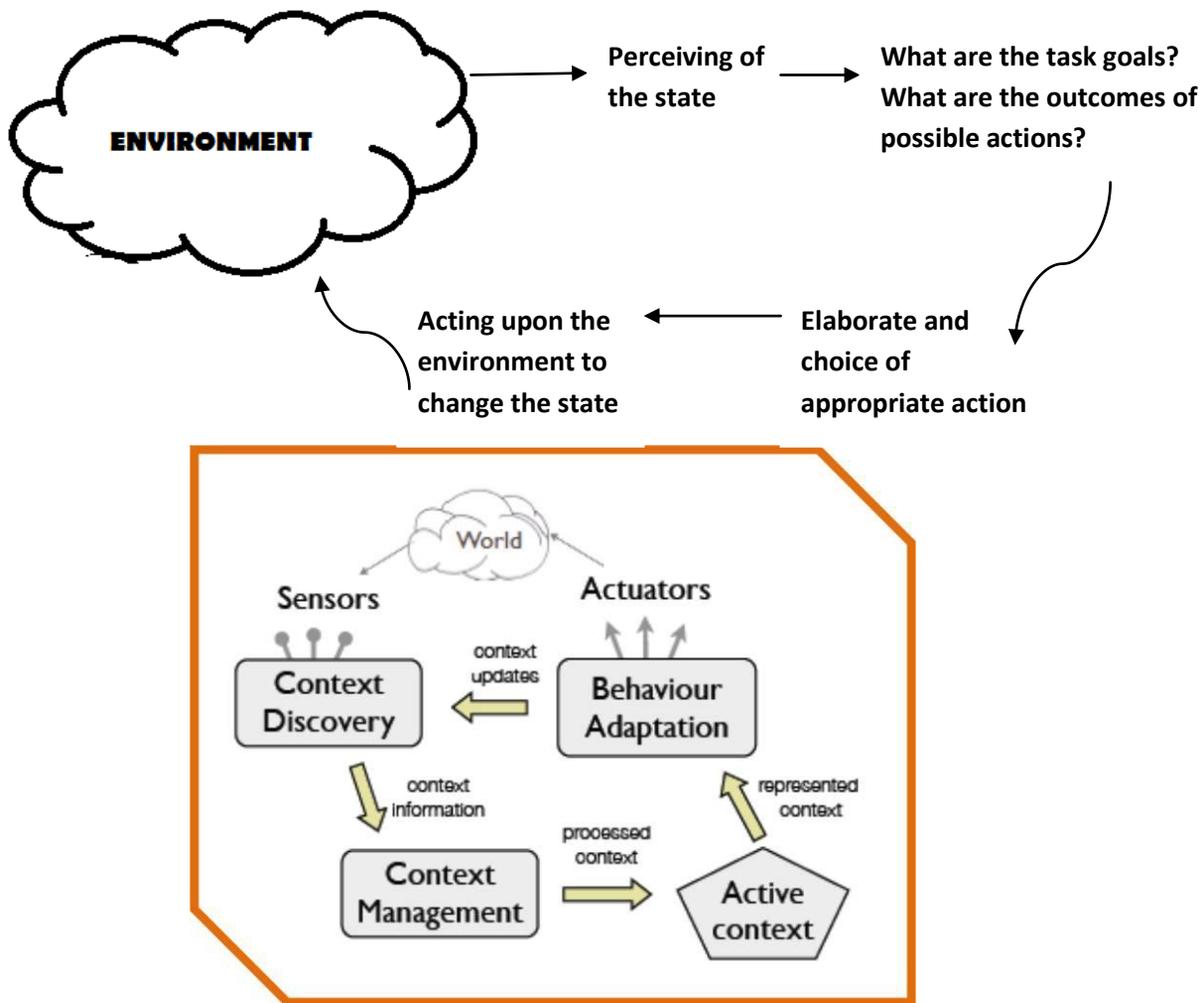


Figure 2.6-2.7: automazione ciclica in uno Smart Environment. Dalla acquisizione del contesto all'adattamento dell'ambiente

2.4 L'approccio ad agenti

Il progresso tecnologico ha portato alla nascita di una nuova classe di sistemi detti *pervasive computing system*, dove gli Smart Environment ne sono una importante caso e sono diretti a rimodellare l'ambiente esterno nel quale gli utenti con nuovi servizi che vengono forniti da software intelligenti integrati in una rete di Smart (mobile) Devices.

In questo modo alcune delle attività solitamente svolte dall'utente possono essere automatizzate con possibili miglioramenti in termini di qualità del servizio ed efficienza.

Il calcolo pervasivo comporta due tipi di intelligenza:

- *Intelligenza della parte individuale*, la quale esegue in modo autonomo la propria attività
- *Intelligenza sociale* la quale fornisce servizi che richiedono il coinvolgimento delle parti autonome che interagiscono in modo coordinato

Dal momento che i tradizionali approcci all'ingegneria del software risultano inadeguati per poter interfacciarsi con la notevole complessità di questi sistemi, è necessario introdurre nuovi modelli e paradigmi.

Il paradigma che riesce a fornire il giusto supporto nella progettazione di questi sistemi, partendo dall'analisi fino allo sviluppo, è il *paradigma ad agenti* e quindi l'approccio basato sui sistemi multi agente.

Infatti gli agenti autonomi rappresentano un modo naturale per concepire il sistema fornito dallo scenario del calcolo pervasivo, fatto –almeno in parte– da entità autonome situate in un ambiente il quale influenza la loro possibilità di agire ed interagire.

Come un agente intelligente infatti, lo Smart Environment si affida ai componenti fisici che percepiscono le proprietà del sistema per poi ragionare sulla base di ciò che è stato percepito per poter agire di conseguenza sull'ambiente e cambiarne lo stato [16].

Il capitolo seguente mira a presentare come sono definiti gli agenti ed i MAS concentrandosi poi sulle loro proprietà e caratteristiche che ne definiscono le diverse tipologie.

Capitolo 3

Agenti e Sistemi Multi Agente

3.1 Gli agenti

I computer –intesi come unità di calcolo e non semplicemente come i personal computer– non sono totalmente consci di ciò che fanno: ogni operazione che un computer esegue deve essere esplicitamente anticipata, pianificata e codificata da un programmatore.

Se si dovessero incontrare situazioni che non siano già state anticipatamente previste e trattate da un programmatore si verrebbe incontro a situazioni spiacevoli (crash di sistema, perdita di informazioni ...).

Per certi aspetti risulta comodo che per alcune applicazioni i computer si comportino come servi obbedienti, però con il costante incremento delle applicazioni ci si muove verso la ricerca di sistemi i quali sono in grado di poter decidere per sé stessi cosa necessitano di fare, al fine di soddisfare i propri obiettivi; questi sistemi di computazioni sono conosciuti come *agenti*.

Anche se non esiste una definizione universalmente accettata di agente si può comunque descrivere un agente come [17] *una entità di calcolo immersa in un ambiente, che possiede capacità di compiere azioni autonome in questo ambiente al fine di poter raggiungere i propri obiettivi* oppure anche [18] *un agente è qualsiasi cosa che sia in grado di vedere e percepire l'ambiente in cui si trova tramite sensori e su cui vi può operare tramite attuatori*.

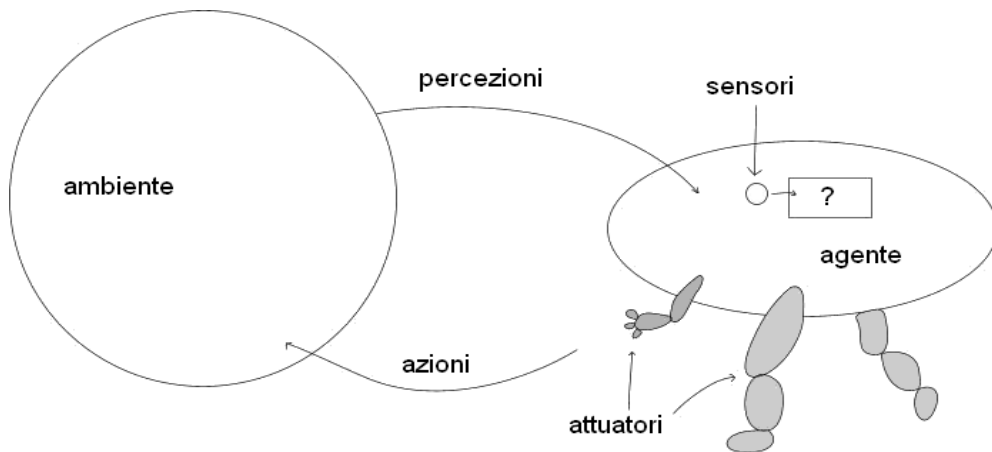


Figura 3.1: gente visto in termini di sensori-attuatori

Nel caso in cui l'agente sia immerso in un ambiente che cambia in modo continuo, in maniera imprevedibile e sia aperto, allora si ha a che fare con *agenti intelligenti* o *agenti autonomi*.

Questi agenti intelligenti sono entità software in grado di agire in modo flessibile ed autonomo prelevando informazioni dall'ambiente in cui si trova scambiando informazioni con altri agenti o con esseri umani.

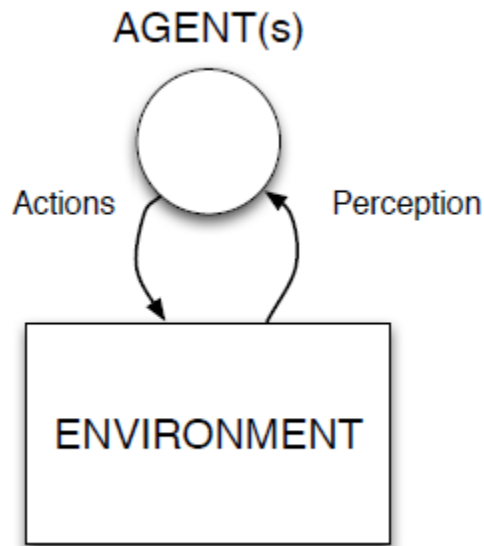


Figura 3.2: Vista di alto livello di un agente e dell'ambiente che influenza.

È stato quindi scelto l'approccio ad agenti per il caso di studio di questa tesi poiché la nozione di agente fornisce le giuste caratteristiche in termini di

autonomia e comportamento proattivo per poter progettare e sviluppare concretamente le parti individuali del sistema.

3.1.1 Proprietà

Definendo gli agenti si fa riferimento ad un certo insieme di caratteristiche, dove quelle fondamentali sono:

- *Reattività*: capacità degli agenti di percepire in cambiamenti nell'ambiente e di rispondere in modo tempestivo agendo su esso, al fine di raggiungere i propri obiettivi
- *Autonomia*: gli agenti incapsulano il proprio flusso di controllo e le regole per governarlo. Un agente non dispone di interfaccia, non può essere controllato e nemmeno invocato –come si fa per gli oggetti– il controllo non passa attraverso il confine agente-ambiente ma solo le informazioni di quest'ultimo lo possono passare

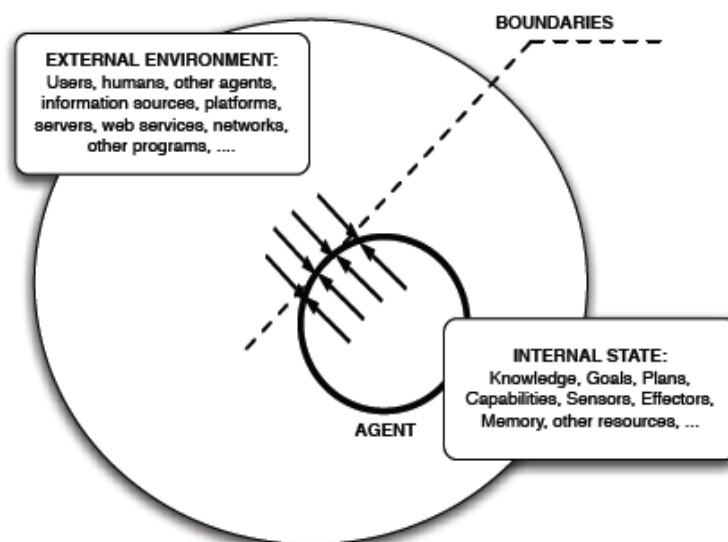


Figura 3.3: confine tra agente ed ambiente esterno

- *Socialità*: un agente è in grado di interagire con altri agenti o con esseri umani per poter soddisfare i propri obiettivi (questa caratteristica sta alla base della nascita dei sistemi multi agente)

In più un agente può essere:

- *Proattivo*: gli agenti sono stati progettati con lo scopo di realizzare autonomamente qualche obiettivo “prendendo l’iniziativa” e non limitandosi ad attendere che accada qualcosa: sono loro che questa cosa la fanno accadere
- *Situato*: ogni azione base di un agente è strettamente connessa con il contesto in cui l’azione viene eseguita, per questo l’agente sarà collocato nell’ambiente col quale interagisce
- *Mobile*: il fatto che un agente sia situato in un ambiente non significa che questo non possa muoversi all’interno di esso, il quale può essere molto vasto ed eterogeneo (Mobile Agents)
- *Adattativo (o di apprendimento)*: attraverso l’interazione con l’ambiente oppure da una grande quantità di dati gli agenti possono imparare e migliorare, grazie ad una memoria di immagazzinamento e capacità di recupero analizzando in sé stesso errori e successi passati in termini di comportamento (Learning Agents)

3.1.2 Tipologie

La tipologia con cui un agente può essere classificato non dipende unicamente dalla combinazione delle proprietà che possiede o meno un certo tipo di agente, ma anche tramite il loro ruolo nel sistema (agenti di informazione, agenti web, agenti di presentazione, etc.) e anche in dipendenza dalle caratteristiche dell’ambiente che influenzano il suo comportamento e le proprie attitudini.

Come enfatizzato dalla figura sottostante, gli agenti possono essere classificati in base alla reattività e pro attività che li caratterizza e, in secondo luogo, attraverso i diversi attributi che gli agenti devono esibire [19]:

- *Autonomia (autonomous)*: gli agenti possono operare da soli senza il bisogno di una guida umana

- *Cooperazione (social)*: l'abilità sociale di interagire con altri agenti o con gli umani
- *Apprendimento (learn)*: imparare tramite l'esperienza come reagire ed interagire con il loro ambiente esterno

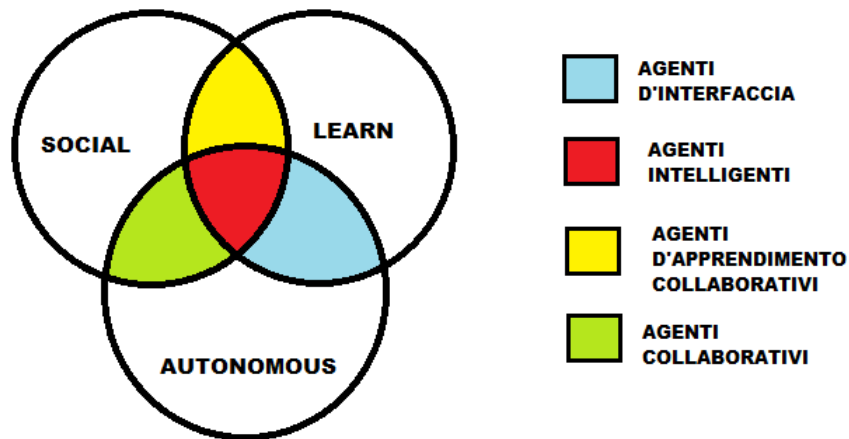


Figura 3.4: si mostrano i vari tipi di agenti esistenti, a seconda della combinazione delle tre proprietà fondamentali che li caratterizzano

È possibile stilare l'insieme delle tipologie di agente attualmente adottati e di maggiore importanza per gli sviluppi futuri analizzando le proprietà/ attributi che lo caratterizzano [19]:

- *Agenti collaborativi*: progettati per collaborare con altri agenti, hanno la facoltà di rifiutarsi di eseguire oppure modificare una richiesta che gli è pervenuta
- *Agenti d'interfaccia*: mediano il dialogo con l'utente rivestendo il ruolo di fornitori di servizi. Sono autonomi ed hanno capacità di apprendimento, ma non comunicano con altri agenti
- *Agenti mobili*: sono in grado di spostarsi da una macchina all'altra all'interno dell'ambiente mantenendo il proprio stato e hanno una fondamentale importanza per quanto riguarda i sistemi distribuiti moderni
- *Agenti d'informazione*: hanno accesso a diverse forme di informazioni; sono in grado di controllare e manipolare informazioni per rispondere a richieste di un altro agente (o un'altra entità in generale)

- *Agenti reattivi*: agiscono unicamente secondo il principio azione-reazione, percependo l'ambiente in cui si trovano ed adeguandosi il proprio comportamento ai cambiamenti di questo
- *Agenti intelligenti*: sono autonomi, dotati di capacità di apprendimento e di collaborazione sociale; possiedono al loro interno un modello simbolico dell'ambiente in cui si trovano e prendono le loro decisioni sulla base di questo modello, in modo simbolico. Esso può incrementare la propria conoscenza a tempo di esecuzione per poter meglio prendere le decisioni rivolte a raggiungere gli obiettivi
- *Agenti ibridi*: combinano le due caratteristiche sopra

3.1.3 Apprendimento

La proprietà di apprendimento di un agente è definita come “l'abilità di imparare con l'esperienza come agire e reagire con l'ambiente esterno”.

Ci chiediamo dunque quali siano i metodi con cui un agente possa acquisire le conoscenze necessarie, in [18] ne vengono classificati alcuni:

- *User looking*: l'agente apprende osservando l'utente, tenendo traccia delle sue scelte e modificando il suo profilo
- *User indirect feedback*: l'agente fornisce una serie di risultati all'utente e prende nota quando questo trascura il procedimento o addirittura compie azioni opposte
- *User direct feedback*: l'agente impara chiedendo esplicitamente all'utente il perché delle scelte fatte
- *Learning by example*: in questo caso è l'utente che fornisce all'agente una serie di esempi sui quali basarsi
- *Agent asking*: quando un agente non ha le conoscenze specifiche chiede ad altri agenti di fornirgli informazioni utili per risolvere un certo problema

3.1.4 Dagli oggetti agli agenti

Comparando le proprietà che caratterizzano gli oggetti con quelle che sono le peculiarità degli agenti, salta subito all'attenzione che entrambe queste entità sono reattive poiché agiscono a stimoli esterni (per gli agenti i cambiamenti dell'ambiente, per gli oggetti le chiamate di un proprio metodo ad esempio) ma, al contrario di quanto visto per i secondi, gli oggetti *non sono autonomi*.

Il meccanismo fondamentale della programmazione ad oggetti (OOP) prevede che sia il chiamante che decide quando trasferire il flusso di controllo all'oggetto chiamando un suo metodo.

In caso contrario ogni oggetto è fermo, è morto, non compie nessuna azione di sua spontanea volontà e potrebbe non farlo mai.

La caratteristica fondamentale di un agente è la sua *autonomia*: essi sono in grado di iniziare una propria attività indipendentemente da ogni altra entità, così un agente può operare senza una diretta invocazione esterna.

Esistono due aree chiave che differenzieranno l'approccio basato ad agenti dalla programmazione OO: autonomia ed interazione; in più altri aspetti ci portano a muoversi verso tecnologie agent-based come ad esempio [20] :

- La decentralizzazione: gli agenti sono tipicamente organizzati in modo centralizzato, dove il metodo invocato da un altro componente del sistema è internamente presente in esso, esistono però situazioni dove è richiesto un certo grado di decentralizzazione ed auto organizzazione che viene fornita dagli ambienti di agenti insieme al calcolo distribuito
- Una volta creati, gli oggetti hanno una classe tipicamente non modificabile mentre gli agenti possono modificare il proprio ruolo, i propri obiettivi e traguardi a seconda dei loro bisogni e dei continui cambiamenti della struttura dell'ambiente che li circonda
- Minore impatto: a differenza degli oggetti all'interno di un sistema, dove se uno di questi viene perduto il sistema fallirà o almeno sarà generata una eccezione, gli agenti hanno un minore impatto sul sistema; infatti se una di questi viene a mancare potrebbe verificarsi una diminuzione delle performance ma non essendo un singolo punto di fallimento la dinamicità della collettività di agenti continua ad essere dominante

- Gli oggetti potenzialmente possono interagire con l'intero spazio del sistema di oggetti ma il campo, lo *scope*, nel quale possono agire è definito una volta per tutte a design time mentre per quanto riguarda gli agenti, essi fanno affidamento sull'ambiente che li circonda, percependolo. Il campo che possono percepire può essere anche ridotto ma hanno la possibilità, se spostati, di poter percepire qualsiasi zona dell'ambiente e quindi il loro *scope* non è definito a priori
- *Emergence*: l'interazione tra più agenti individuali provoca un effetto secondario, nel quale un gruppo di agenti si comporta come una singola entità al contrario degli oggetti
- Ogni agente può modificare/aggiungere/rimuovere le sue proprietà, ha la capacità di imparare e quindi cambiare il proprio comportamento dinamicamente fornendo un alto grado di flessibilità e dinamicità che gli oggetti e la programmazione OO tipicamente non forniscono

Con il passare del tempo le cose sono cambiate muovendosi verso la computazione parallela e la distribuzione di più flussi di controllo su più oggetti confermando il bisogno da parte dei nuovi sistemi di essere caratterizzati dalle proprietà fondamentali che portano con loro gli agenti ed i sistemi che questi formano.

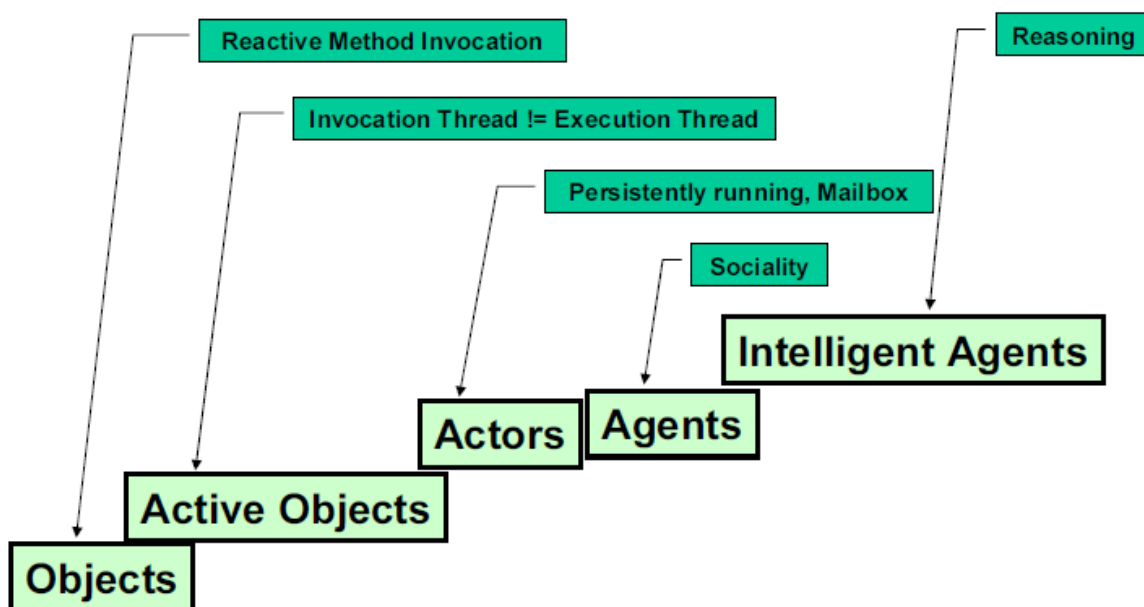


Figura 3.5: si mostra come gli sviluppi della ricerca hanno portato ad entità sempre più autonome, collaborative fino ad arrivare ai moderni agenti intelligenti.

3.2 Sistemi multi agente

I sistemi multi agente sono una delle aree di ricerca e sviluppo più importanti dell'informatica che sono emersi all'inizio degli anni '90; questi sono sistemi computazionali nei quali diversi agenti interagiscono o lavorano insieme su una rete di comunicazione al fine di svolgere un set di attività insieme oppure per soddisfare un insieme di obiettivi.

Ad oggi, i sistemi di calcolo pervasivi e complessi sono in costante crescita: concorrenza, distribuzione, apertura, eterogeneità sono divenute le tipiche caratteristiche dei sistemi moderni e, sin dalla loro nascita i sistemi multi agente o MAS (Multi Agent Systems) sono riconosciuti come il paradigma capace di poter fronteggiare la complessità di questi sistemi grazie alle sue proprietà.

Tali proprietà possono essere individuate in (i) componenti applicativi autonomi –gli agenti– che concretizzano così la natura decentralizzata dei sistemi distribuiti moderni, (ii) flessibilità con la quale si interagisce e si opera con gli agenti (tra di essi o con l'ambiente).

Oltre all'astrazione degli agenti abbiamo altri due tipi di astrazioni molto importanti che vengono introdotte dalla nozione dei sistemi multi agente, le quali sono: la *società di agenti* e *l'ambiente*.

3.2.1 Società di agenti

L'astrazione di agente da sola infatti non può bastare: un agente non vive mai da solo, esso infatti coesiste con altri in un MAS come una articolata e complessa organizzazione, una società: termine che distingue il sistema multi agente da una semplice aggregazione di più agenti.

Gli agenti comunicano –si scambiano e condividono informazioni– al fine di conseguire nel miglior modo possibile i propri scopi o quelli del sistema multi agente a cui appartengono.

La nozione di società di agenti è fondamentale per pianificare gli aspetti sociali, come compiti sociali, gestione della coordinazione e cooperazione tra agenti (intelligenza collettiva), norme organizzative e vincoli ambientali [21].

In una società gli agenti lavorano indipendentemente l'uno dall'altro per raggiungere i loro obiettivi individuali, mentre per raggiungere gli obiettivi globali comunicano e instaurano rapporti di interazione, definiti da regole sociali che guidano il comportamento globale della società verso la realizzazione degli obiettivi comuni.

Il comportamento di un agente che “convive” in una società di agenti, è spesso incomprensibile al di fuori della società in cui opera: una società di agenti esibisce comportamenti globali che non sono equivalenti alla somma dei comportamenti dei componenti individuali; i MAS sono qualcosa in più che un semplice mucchio di agenti!

Le regole sociali sfruttano l'interazione fra agenti per l'adempimento dei compiti sociali: si dovrebbe poter determinare cosa sia di competenza delle infrastrutture sociali e cosa dei singoli componenti, problema riguardante soprattutto i sistemi aperti.

Nelle società di agenti, il comportamento dei membri e le loro interazioni non sono predicibili, perciò l'evoluzione della società non è deterministica, infatti non è detto che i membri della società condividano gli stessi obiettivi, pertanto i membri potrebbero anche entrare in conflitto mentre sono impegnati a perseguire i propri scopi, inoltre l'architettura interna di ogni membro non è solitamente nota agli altri membri, quindi possono esserci architetture entità che interagiscono fra loro.

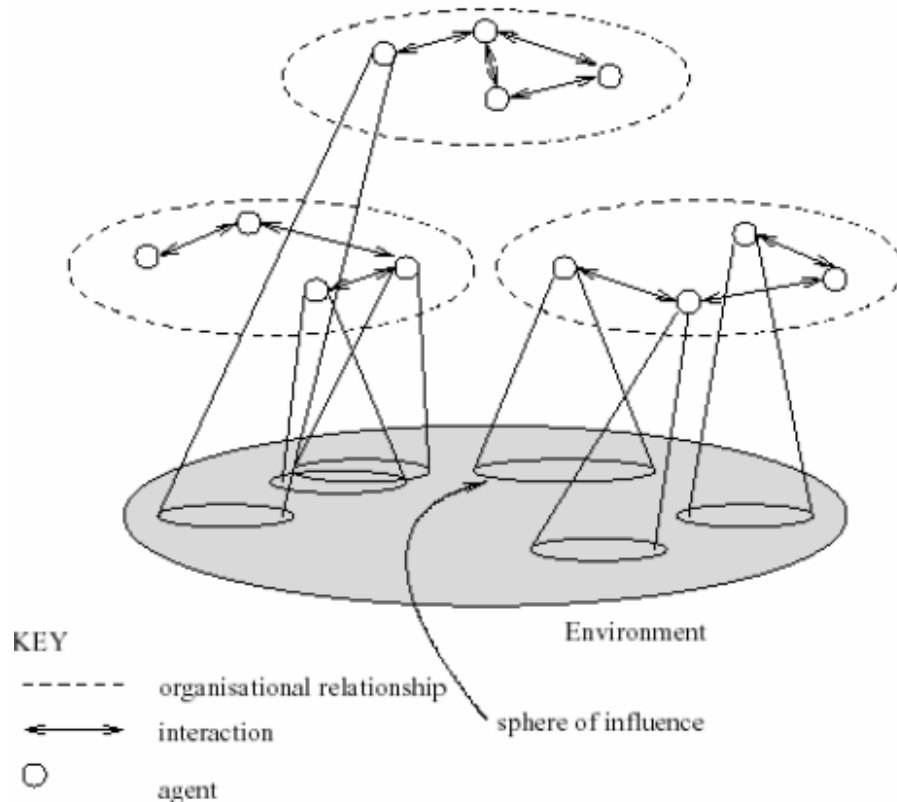


Figura 3.6: società d'agenti e sue aree di influenza.

La figura sopra riassume quanto detto fino ad ora, e in più possiamo realizzare che gli agenti sono legati da certe relazioni di organizzazione ad altre entità per poter agire su certe “aree di influenza”, le quali possono coincidere anche per agenti che non sono direttamente relazionati tra di loro.

3.2.2 Comunicazione nella società

Gli agenti di una società devono collaborare tra di loro e per poter interagire, devono altresì essere forniti di un sistema di comunicazione.

I meccanismi di comunicazione usati sono diversi: da quello dello scambio di messaggi, in cui l'invio o la ricezione di un messaggio può essere visto come una azione eseguita dall'agente, al modello a lavagna (blackboard), dove gli agenti comunicano inserendo e prelevando informazioni sulla lavagna allo spazio di informazione condiviso che favorisce il disaccoppiamento (tuple space), fino ad altre forme di comunicazione più complesse.

Un linguaggio avanzato di comunicazione che può essere adottato per i MAS è quello dello standard FIPA (Foundation for Intelligent Physical Agents), che

si propone di definire in modo rigoroso la semantica di un linguaggio di comunicazione per agenti in sistemi multi agente.

FIPA [22] si concentra sulla definizione del “comportamento esterno” –cioè sociale– degli agenti piuttosto che su quello “interno” –il modo in cui gli agenti “ragionano” ed elaborano le informazioni ricevute dipende dalla particolare implementazione– ed è riuscita a raggiungere un elevato livello di interoperabilità anche tra tecnologie differenti.

3.2.3 Ambiente

Quando si progettano gli agenti, non si possono separare dall’ambiente in cui vivono; un ambiente multi agente fornisce l’infrastruttura per l’interazione fra gli agenti, che comprende i protocolli per la comunicazione e per l’interazione.

Gli agenti e le società di agenti vivono in un ambiente eterogeneo, dinamico, aperto, imprevedibile; queste proprietà dell’ambiente influiscono ovviamente sul modo in cui gli agenti rappresentano il mondo in cui vivono e su come decidono e pianificano le loro azioni.

Le metodologie orientate agli agenti devono aiutare a modellare l’ambiente dalle prime fasi di ingegnerizzazione, ed descrivere le dipendenze tra gli agenti e l’ambiente stesso.

Le caratteristiche dell’ambiente spesso non sono note a priori, ma vengono definite secondo le esigenze del sistema; secondo le proprie caratteristiche in ambiente può essere [23]:

- Accessibile, parzialmente o totalmente, da un agente
- Deterministico o non deterministico (stocastico), anche se solitamente è deterministico
- Episodico o sequenziale: gli eventi di tipo episodico sono i più semplici da affrontare per un agente in quanto può decidere quale azione compiere basandosi unicamente sull’episodio corrente
- Statico o (tipicamente) dinamico
- Discreto (come le mosse nel gioco della dama o degli scacchi) o continuo (come la guida di un’automobile)
- Con un singolo o agente o (solitamente) multi agente

Il modo in cui i servizi sono realizzati e resi disponibili agli agenti dipende dal modo in cui il sistema e i suoi agenti sono organizzati, le metodologie quindi devono anche aiutare a realizzare, configurare e perfezionare l'ambiente.

L'ambiente, è la chiave di tutto il MAS nella progettazione di uno Smart Environment: sarà l'ambiente con i valori assunti dalle sue proprietà caratteristiche il motore di tutto il lavoro degli agenti –sia singolarmente che collettivamente, sia in modo reattivo che proattivo– a fare sì che la vita degli abitanti dell'ambiente esterno sia semplificata, facendosi carico di parte delle attività.

3.2.4 Progettazione di un MAS

Per quanto riguarda le caratteristiche degli agenti che compongono il sistema, niente sarà definito a “design time” ,di fatti le loro proprietà, il loro comportamento e le loro capacità saranno strettamente relazionate con il tipo di ambiente nel quale si troveranno.

Le questioni principali che si trovano ad affrontare i progettisti di un sistema multi agente sono suddivisibili in aspetti interni: ovvero l'analisi, la progettazione e lo sviluppo dei singoli agenti come entità fondamentali che compongono il sistema, e aspetti comunicativi: come l'interazione degli agenti con l'ambiente, con le risorse e con altri agenti.

Certamente la progettazione di un MAS sarà più complessa rispetto alla progettazione di un singolo agente, di fatti una volta progettati i singoli agenti si dovrà definire il sistema a livello di:

- *Interazione*: perché gli agenti cooperano tra loro? Non hanno capacità sufficienti per completare i loro task? Oppure non hanno le risorse sufficienti per farlo? O entrambe le cose?
- *Comunicazione*: come comunicano tra loro? Si scambiano messaggi? Tramite segnali? Usano una blackboard? Tramite StickyNotes virtuali?

- *Organizzazione*: come sono organizzati? Come modello master-slave oppure p2p? l'organizzazione è fissata in fase iniziale oppure emerge durante la risoluzione del problema?

I sistemi multi agente forniscono quindi un potenziale modello di calcolo nel quale una rete di agenti intelligenti i quali interagiscono in maniera real time integrando il lavoro delle persone e delle macchine; le applicazioni che vengono affrontate adottando un sistema multi agente piuttosto che un singolo agente portano con loro diversi vantaggi:

- Accelerano il calcolo concorrente
- Diminuiscono il flusso comunicativo, poiché l'elaborazione è localizzata nelle vicinanze della sorgente informativa
- Aumentano l'affidabilità non essendoci un unico punto nel quale possono verificarsi fallimenti
- Semplificano lo sviluppo del sistema grazie alla loro modularità (dividendo il carico applicativo in più agenti)
- Reattività real time ai cambiamenti ambientali
- Forniscono soluzioni a problemi che attingono da fonti di conoscenza distribuite, oppure a problemi inerentemente distribuiti

Gli ingeneri di un MAS dovranno poi modellare adeguatamente gli aspetti riguardanti la comunicazione e la condivisione di risorse/informazioni tra gli agenti nel MAS, nonché –visto il parallelismo delle operazioni nel sistema– aspetti relativi alla coordinazione delle operazioni che intercorrono nel tempo.

Le peculiarità dei sistemi ad agenti risultano quindi una fonte adeguata di nuove astrazioni, meta modelli, tecnologie e metodologie per i sistemi complessi di oggi come sono appunto gli Smart Environment.

Capitolo 4

Infrastruttura dell'ambiente MAS

La nozione di ambiente sta ricevendo una crescente attenzione nello sviluppo dei sistemi multi agente, ciò è testimoniato dal crescente numero di infrastrutture fornite dai progettisti, utili per lo sviluppo dell'ambiente degli agenti così da strutturare una effettiva applicazione multi agente.

Come accennato nel precedente capitolo l'approccio multi agente è basato sull'idea che le caratteristiche emergenti dei moderni e complessi sistemi software possano essere gestite opportunamente modellando il software come una composizione di entità autonome, agenti, volti ad ottenere un certo obiettivo.

Quindi per formare un sistema multi agente, gli agenti dovrebbero essere situati in un ambiente computazionale e/o fisico, nel quale le interazione degli agenti con altri agenti e con le risorse sia possibile.

Oggigiorno è riconosciuto che l'ambiente risulta una vera e propria dimensione di progettazione di un sistema multi agente: esso incapsula una buona porzione della complessità del sistema in termini di servizi e meccanismi di cui gli agenti possono essere liberati.

Esistono una buona varietà di infrastrutture per supportare l'ambiente in una applicazione multi agente, la loro struttura ed i loro obiettivi sono diversi: possono essere concepiti per una specifica applicazione oppure per catturare e gestire alcuni scenari generali; possono essere basati su reti cablate, wireless, ad hoc o possono fornire servizi relativi alla coordinazione alla organizzazione o altro ancora.

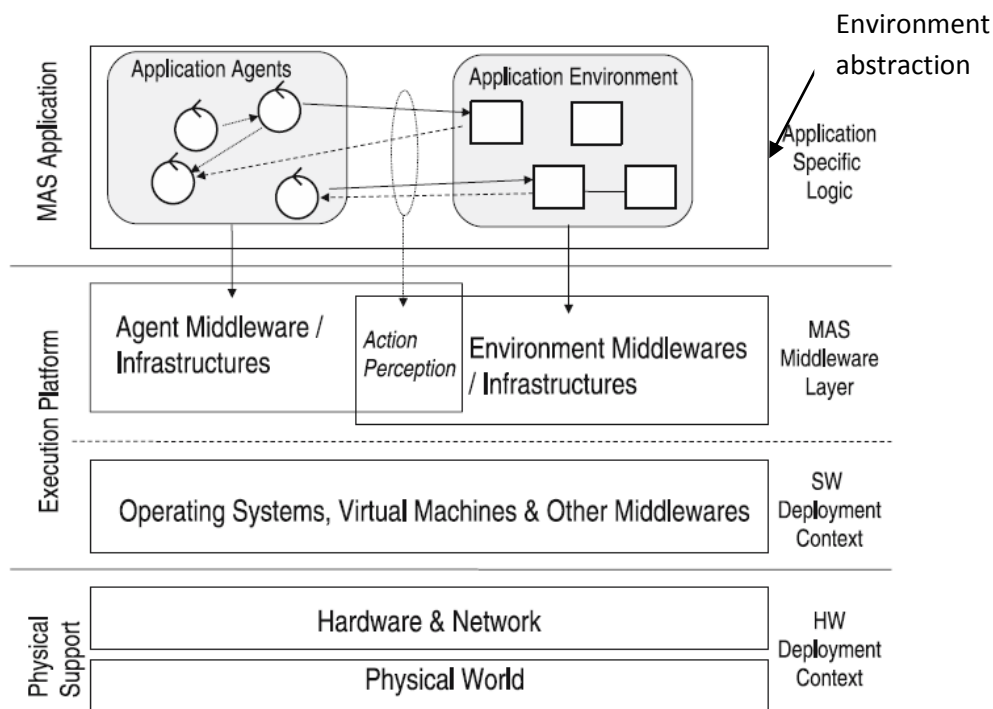


Figura 4.1: struttura a strati di un MAS con un supporto basato sull'ambiente.

Come vediamo dalla figura sopra esistono due tipi principali di infrastrutture che stanno alla base di una applicazione MAS (chiamati anche middleware): (i) una infrastruttura che fornisce il ciclo di vita degli agenti, la loro gestione e spesso altri servizi fondamentali come la diretta comunicazione (ad esempio JADE), e (ii) una o più infrastrutture d'ambiente le quali forniscono alcune classi di astrazioni ambientali (tuple spaces, stigmergic field, e altro).

In questo capitolo dell'elaborato si discuteranno le varie infrastrutture d'ambiente che formeranno poi l'infrastruttura globale del MAS che sta alla base dello Smart Environment, concentrando l'attenzione sulle infrastrutture che forniscono un supporto per l'interazione nell'ambiente: dalla semplice cooperazione/coordinazione tra agenti alla condivisione ed utilizzo delle risorse presenti.

Partiremo dunque con il definire il modello di coordinazione per il nostro sistema specificando quelle che sono le componenti ed i criteri di comunicazione tra essi.

4.1 Modello di coordinazione

Negli scenari moderni, i sistemi sono caratterizzati dal calcolo concorrente/parallelo, da attività che sono distribuite –possono girare in contesti eterogenei, su differenti dispositivi/macchine– da interazioni sociali –più componenti del sistema cooperano per raggiungere obiettivi comuni– e ambientali –interazioni con risorse esterne–, ciò porta a non fare considerazioni a priori su quello che è il comportamento, la struttura, la locazione o la permanenza nel sistema dei suoi vari componenti.

Abbiamo visto come per modellare attività indipendenti, concorrenti e cooperanti è stato scelto un approccio ad agenti caratterizzati da facoltà di calcolo indipendenti e da capacità di I/O per poter interagire con altri componenti; è quindi fondamentale la scelta di un appropriato modello di coordinazione con il quale trattare le continue e numerose interazioni tra agenti.

Nella programmazione distribuita il modello di coordinazione è visto come una “colla” che lega le varie attività indipendenti in un complesso [24], fornisce inoltre una struttura nella quale le interazioni delle entità attive e indipendenti chiamate agenti possono essere espresse.

Questo modello deve comprendere le questioni riguardanti la creazione degli agenti, la comunicazione tra essi, la loro distribuzione spaziale così come la distribuzione e la sincronizzazione delle loro attività nel tempo.

Ogni modello di coordinazione può essere suddiviso in tre elementi principali:

- *Coordinables*: sono le entità le cui interazioni sono regolate dal modello
- *Coordination Media*: sono il mezzo di coordinazione, le astrazioni che consentono le interazioni (es: spazi di tuple)
- *Coordination Laws*: sono le leggi di coordinazione che definiscono la semantica di interazione che il mezzo di coordinazione realizza. Possono essere definite in termini di:
 - *communication language* : linguaggio per lo scambio di informazioni e strutture dati, e

- *coordination language*: insieme di primitive di interazione

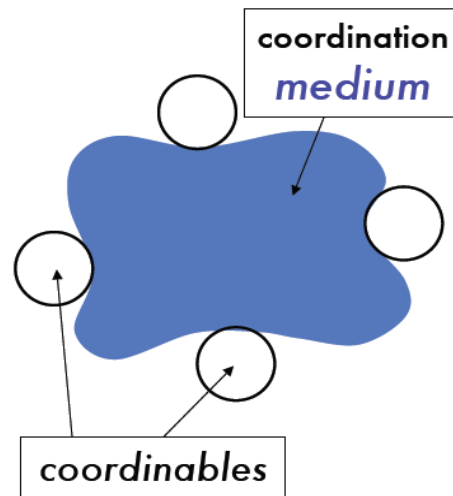


Figura 4.2: raffigurazione di un modello di coordinazione

dove l'insieme delle primitive di interazione sono utilizzate dai coordinabili per poter esprimere le loro richieste e svolgere le proprie attività, anche combinandone più tra loro in modo da poter compiere operazioni più complesse.

Si possono individuare due classi principali di modelli di coordinazione, orientati ai dati e al controllo, da cui poi nasceranno modelli ibridi:

- *Information driven*: in questo modello il mezzo di coordinazione funge da spazio condiviso di informazioni (come una shared repository) senza che venga a conoscenza della variazione di stato dei coordinabili, i quali interagiscono tra loro scambiandosi strutture dati. Le leggi di coordinazione stabiliscono come i dati devono essere rappresentati. Questo modello si concentra sulle informazioni che vengono scambiate durante la comunicazione
- *Control driven*: in questo caso le entità coordinabili interagiscono tra di loro tramite eventi. Questo modello focalizza l'attenzione sull'atto della comunicazione vero e proprio
- *Ibridi*: sono i modelli che supportano entrambe le tipologie di coordinazione

Il modello di coordinazione preso in esame nell'elaborato è quello basato su tuple per poter fornire l'astrazione richiesta per la costruzione di una società di agenti.

4.1.1 Modello basato su tuple

Nel meta modello basato su tuple gli agenti interagiscono tramite lo scambio di *tuple* le quali sono insiemi ordinati di frammenti di informazioni, possibilmente eterogenei.

Gli agenti in questo modello di coordinazione sono i coordinabili; essi cooperano, comunicano e si sincronizzano memorizzando, leggendo e consumando tuple disponibili in uno spazio condiviso chiamato spazio di tuple in maniera associativa.

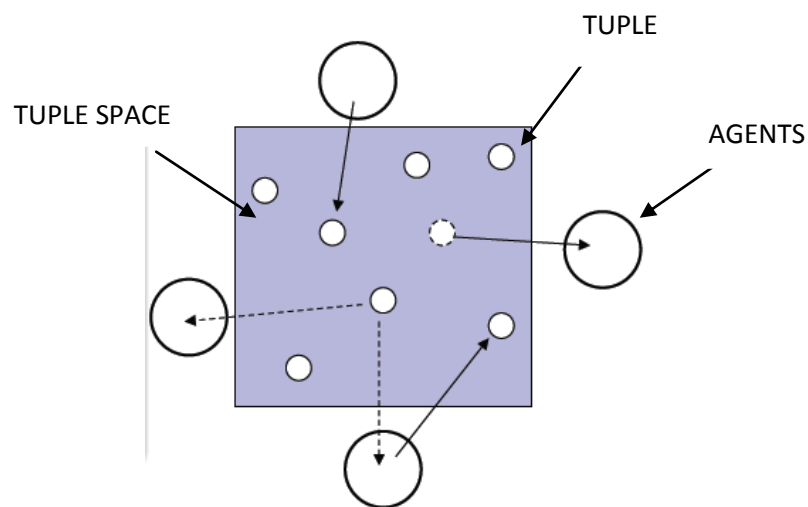


Figura 4.3: interazione tra agenti tramite uno spazio di tuple

Quindi in questo modello lo spazio di tuple assumerà il ruolo di medium coordinativo, mentre per quanto riguarda le regole di coordinazione il linguaggio di comunicazione è riconducibile all'insieme di tuple consultate e il linguaggio di coordinazione sono le primitive dello spazio di tuple attraverso il quale è possibile ricercare o inserire tuple da/sullo spazio di tuple.

In questa maniera emerge una forma di comunicazione ortogonale nel quale sia il mittente che il ricevitore interagiscono senza che abbiano una conoscenza a priori l'uno dell'altro, inoltre si ha un disaccoppiamento della comunicazione in tre dimensioni:

- **Spazio:** non è necessario che i due processi in comunicazione debbano coesistere nello stesso spazio

- **Tempo:** non è necessario che i due processi interagiscano in maniera simultanea
- **Nomi:** non è necessario che un processo che voglia interagire con un altro conosca il nome di questo. Infatti la comunicazione avviene attraverso tuple le quali non portano con loro nessuna informazione riguardante il processo che l'ha generata

4.1.2 Dagli spazi di tuple ai centri di tuple

Il principale problema del modello di coordinazione basato sullo spazio di tuple è legato al comportamento fissato a design time del medium di coordinazione.

Dato un modello che adotta gli spazi di tuple come coordination media, il comportamento dello spazio di tuple, rappresentato dal cambiamento di stato al verificarsi di eventi di comunicazione, è fissato una volta per tutte dal modello [25] e non può essere fatto su misura per le specifiche necessità di una applicazione.

I modelli che però si considerano nello scenario dell'elaborato sono quelli *control driven*, ovvero quelle in cui i coordinabili comunicano tra loro tramite eventi: infatti lo Smart Environment prevede che eventi generati da ambiente o altri agenti comportino una successiva elaborazione di questi per adattare l'intero sistema; questi modelli possiedono evidenti lacune nelle capacità di controllo, e tipicamente per compensare queste mancanze si carica il peso della coordinazioni sulle entità coordinabili, gli agenti nel nostro caso.

Nell'insieme però tutto questo risulta concettualmente inadeguato per come sono stati definiti gli agenti

Ma tutto questo è concettualmente inadatto per quanto abbiamo detto fino ad ora per gli agenti, i quali devono essere context aware e non possono astrarre i dettagli coordinativi.

Nasce dunque la necessità di adattare i modelli basati sul controllo a contesti *agent based* ed è quindi utile introdurre per il nostro sistema un approccio ibrido che includa proprietà da entrambi i modelli.

Ciò che sarà lasciato immutato è quanto riguarda la semantica/sintassi delle primitive Linda, ed il meccanismo di matching mentre si introducono nuove proprietà come la possibilità di definire nuovi comportamenti coordinativi oppure l'abilità di associare comportamenti coordinativi al verificarsi di certi eventi, si introducono dunque i centri di tuple.

Un centro di tuple è definito come [25] *uno spazio di tuple migliorato con un behaviour specification –specifica come lo spazio si comporta al verificarsi di un evento– il quale determina un certo comportamento in risposta ad eventi comunicativi.*

Questo behaviour specification permette di associare ad ogni evento che si può verificare in un centro di tuple, un insieme –anche vuoto– di attività chiamate *reazioni*.

Queste reazioni sono percepite dagli agenti come una singola transizione di stato del centro di tuple, perciò i processi continuano a vedere il centro di tuple come un semplice spazio di tuple; mentre prima le transizioni di stato di uno spazio di tuple consistevano in lettura/memorizzazione delle singole tuple, ora con le reazioni queste transizioni possono essere create complesse secondo le necessità.

Adottando i tuple centre come modello coordinativo per il nostro sistema multi agente, gli agenti continuano a percepirlo come un modello information driven mentre dal punto di vista del medium di coordinazione sono presenti proprietà tipiche dei modelli control driven, con il vantaggio di (i) poter osservare pienamente gli eventi, (ii) poter scegliere selettivamente a quali eventi reagire, (iii) poter implementare le regole di coordinazione manipolando lo spazio di interazione.

Nasce così la nozione di un medium di coordinazione programmabile basato sulla piena osservabilità degli eventi di comunicazione così come la programmabilità delle reazioni a questi eventi, come modo per estendere/determinare il comportamento dell'astrazione di comunicazione.

4.2 Linguaggi di coordinazione e comunicazione

4.2.1 LINDA

Di seguito viene brevemente analizzato il linguaggio di comunicazione Linda –come linguaggio costruito per supportare la comunicazione generativa che sta alla base dei nuovi sistemi distribuiti– per arrivare a quelle che sono le primitive del linguaggio di coordinazione per poter interagire con lo spazio di tuple.

Per poter accedere ad una o più tuple memorizzate nello spazio ci si basa su un metodo associativo; lo si fa specificando quello che si chiama un *template* ovvero la specifica di un insieme/classe di tuple che si desidera recuperare.

Una volta determinato il template, il recupero di queste si fonda su un meccanismo di matching il quale restituisce, come una query in un database, l'insieme di tuple specificate nel template che fanno match con quelle presenti nello spazio di tuple.

Ad esempio: $p(1)$, $\text{printer}(\text{"HP"}, \text{dpi}(300))$, $[20, 0.1]$ sono possibili tuple che possono trovarsi nello spazio, un processo che vuole recuperare tuple dello stesso dovrà anche specificare quali sono i parametri da esso ricercati per quel genere di tupla: $p(x)$, $[?int, ?int]$ dove nel primo caso si cercano le tuple di tipo “p” che facciano match con il parametro “x” da essi specificato, nel secondo si cerca una tupla composta da due numeri interi.

Le primitive del linguaggio di coordinazione sono di facile comprensione:

- **out (*T*)** mette le tuple nello spazio: ad esempio $\text{out}(p(1))$ $\text{out}(1.2, 20)$ $\text{out}(\text{printer}(\text{"HP"}, \text{dpi}(300)))$
- **in (*T*)** operazione che recupera le tuple, va a cercare dentro allo spazio di tuple se ne esiste una che fa match con il template *T*

specificato come parametro. Una volta recuperata la tupla, questa viene rimossa dallo spazio, è inoltre una primitiva bloccante, ossia il processo si blocca finché la tupla non viene trovata

- **$rd(T)$** simile alla rd , è una primitiva sospensiva che però al contrario di quella vista in precedenza non elimina la tupla dallo spazio, ma la lascia intatta.
- **$inp(T)$ / $rdp(T)$** analoghe alla $in(T)$ ed alla $rd(T)$ ma con la particolarità che non sono primitive bloccanti, i processi continuano a lavorare ad altre attività. Viene così introdotto il concetto di fallimento: se nessuna tupla dello spazio fa match con quella specificata nel template viene riportato un fallimento
- **$rd_all(T)$ / $in_all(T)$** poiché le primitive rd e in se trovano più tuple ne ritornano una sola, con questa primitiva viene ritornato l'insieme delle tuple che fanno match con il template

data la natura distribuita dei sistemi multi agente, gli spazi di tuple potrebbero anche risiedere su diversi nodi della rete e quindi esisteranno anche primitive sulla specifica dell'area della rete che interessa esaminare: **$ts @node out(T)$** in questo esempio la tupla T viene messa nello spazio di tuple che risiede nel nodo specificato "node".

4.2.2 Linguaggio ReSpecT

ReSpecT (Reaction Specification Tuples) è un linguaggio logico di coordinazione per i sistemi multi agente diretto a definire il comportamento del centro di tuple al fine di coordinare le entità computazionali autonome come, nel caso analizzato dalla tesi, gli agenti software.

ReSpecT adotta il modello di coordinazione basato su tuple logiche dove la comunicazione è appunto costituita da queste tuple ed il comportamento del medium coordinativo in risposta agli eventi non è fissato una volta per tutte dal modello di coordinazione, ma può essere adattato ai requisiti globali dell'ambiente.

Dato che un centro di tuple altro non è che uno spazio di tuple arricchito con la possibilità di programmare il proprio comportamento in risposta alle interazioni, possiamo prima di tutto dire che gli agenti possono operare con un centro di tuple ReSpecT nella stessa maniera come uno spazio di tuple Linda: ovvero con lo scambio di tuple attraverso semplici primitive di coordinazione [26].

Sulla base di questo schema concettuale, ReSpecT ha un duplice ruolo:

- come un linguaggio di specifica, ReSpecT permette di filtrare gli eventi di comunicazione per potervi associare reazioni per mezzo di tuple logiche
- come un linguaggio di reazione supporta la nozione di reazione come attività computazionale da eseguire localmente al centro di tuple in risposta ad eventi comunicativi fornendo sia una sintassi tuple-based sia un modello per la loro esecuzione

Nella tabella sotto riportata viene mostrata il cuore della sintassi ReSpecT:

```

<TCSpecification> ::= { <SpecificationTuple> . }
<SpecificationTuple> ::= reaction( <SimpleTCEvent> , [ <Guard> , ] <Reaction> )
<SimpleTCEvent> ::= <SimpleTCPredicate> ( <Tuple> ) | time( <Time> )
<Guard> ::= <GuardPredicate> | ( <GuardPredicate> { , <GuardPredicate> } )
<Reaction> ::= <ReactionGoal> | ( <ReactionGoal> { , <ReactionGoal> } )
<ReactionGoal> ::= <TCPredicate> ( <Tuple> ) | <ObservationPredicate> ( <Tuple> ) |
  <Computation> | ( <ReactionGoal> ; <ReactionGoal> )
<TCPredicate> ::= <SimpleTCPredicate> | <TCLinkPredicate>
<TCLinkPredicate> ::= <TCIdentifier> ? <SimpleTCPredicate>
<SimpleTCPredicate> ::= <TCStatePredicate> | <TCForgePredicate>
<TCStatePredicate> ::= in | inp | rd | rdp | out | no | get | set
<TCForgePredicate> ::= <TCStatePredicate> _s
<ObservationPredicate> ::= <EventView> _ ( <EventInformation> )
  <EventView> ::= current | event | start
  <EventInformation> ::= predicate | tuple | source | target | time
  <GuardPredicate> ::= request | response | success | failure | endo | exo | intra | inter |
    from_agent | to_agent | from_tc | to_tc |
    before( <Time> ) | after( <Time> )
  <Time> is a non-negative integer
  <Tuple> is Prolog term
  <Computation> is a Prolog-like goal performing arithmetic / logic computations
  <TCIdentifier> ::= <TCName> @ <NetworkLocation>
  <TCName> is a Prolog ground term
  <NetworkLocation> is a Prolog string representing either an IP name or a DNS entry

```

Figura 4.4: Core syntax of ReSpecT

in cui le reazioni sono definite attraverso tuple logiche: la tupla di specifica **reaction (Op, R)** associa ad un evento generato da una operazione di comunicazione *Op* alla reazione *R* definita come una sequenza di “Reaction Goal” con i quali possono accedere alle proprietà dell’evento (vedi sotto), eseguire una serie di semplici operazioni e manipolare le tuple di un tuple centre.

Tuple space access and modification

<code>out_r(T)</code>	succeeds and inserts tuple <code>T</code> into the tuple centre
<code>rd_r(TT)</code>	succeeds, if a tuple <code>T</code> matching template <code>TT</code> is found in the tuple centre, by unifying <code>T</code> with <code>TT</code> ; fails otherwise
<code>in_r(TT)</code>	succeeds, if a tuple <code>T</code> matching template <code>TT</code> is found in the tuple centre, by unifying <code>T</code> with <code>TT</code> and removing <code>T</code> from the tuple centre; fails otherwise
<code>no_r(TT)</code>	succeeds, if no tuple matching template <code>TT</code> is found in the tuple centre; fails otherwise

Communication event information

<code>current_tuple(T)</code>	succeeds, if <code>T</code> matches the tuple involved by the current communication event
<code>current_agent(A)</code>	succeeds, if <code>A</code> matches the identifier of the agent which triggered the current communication event
<code>current_op(Op)</code>	succeeds, if <code>Op</code> matches the operation which triggered the current communication event
<code>current_tc(N)</code>	succeeds, if <code>N</code> matches the identifier of the tuple centre where the reaction is executed
<code>pre</code>	succeeds in the <i>pre</i> phase of any operation
<code>post</code>	succeeds in the <i>post</i> phase of any operation
<code>success</code>	succeeds in the <i>pre</i> phase of any operation, and in the <i>post</i> phase of any successful operation
<code>failure</code>	succeeds in the <i>post</i> phase of any failed operation

Figura 4.5: principali predicati ReSpecT per le reazioni e proprietà d'evento osservabili.

4.3 Verso la definizione di un nuovo meta modello

4.3.1 Tuple centre ReSpecT e MAS

Coerentemente con la natura tipicamente distribuita dei sistemi multi agente, ogni centro di tuple ReSpecT memorizza le informazioni dell'area di sistema in cui si trovano in termini di tuple logiche,

ordinarie, ed il suo comportamento in termini di specifiche ReSpecT, ovvero sotto forma di tuple di specifica, in maniera locale.

In questo modo si torna a quando detto in precedenza nel capitolo riguardante i sistemi pervasivi, si rende possibile la distribuzione dell'intelligenza attraverso vari nodi del sistema –quando necessario–, infatti gli agenti sono comunemente distribuiti attraverso lo Smart Environment e possono rendere disponibile la loro conoscenza sotto forma di tuple ovunque essi si trovino.

Inoltre grazie alla loro rappresentazione uniforme, gli agenti sono in grado di manipolare le tuple ordinarie e le tuple legate al comportamento adottando lo stesso protocollo.

Ciò consente agli agenti intelligenti di modificare/ integrare il comportamento del sistema multi agente nella stessa maniera diretta, lineare, con cui comunicano con altri agenti, ovvero aggiungendo, rimuovendo o leggendo tuple di specifica (di comportamento).

4.3.2 Situatedness nei tuple centre

I moderni sistemi multi agente sono sistemi distribuiti immersi in un ambiente e dovrebbero essere reattivi agli eventi di qualsiasi tipo generati da questo, così il coordinatore dovrà mediare al meglio ogni attività verso l'ambiente per consentire una interazione redditizia.

In questo caso i centri di tuple ReSpecT non dovranno soltanto catturare gli eventi comunicativi ma devono essere anche in grado di catturare eventi generici generati dall'ambiente esterno.

Questa proprietà che caratterizza i MAS chiamata *situatedness* descrive di un sistema di essere strettamente legato all'ambiente che lo circonda reagendo nel modo migliore ai cambiamenti di questo [2].

Partendo da queste valutazioni il linguaggio ReSpecT dovrà essere opportunamente esteso per poter gestire eventi ambientali ed esprimere le interazioni di un ambiente MAS, ovvero per poter esplicitamente interagire con l'ambiente.

In accordo con questa nuova prospettiva il sistema multi agente che compone il nostro Smart Environment verrà modellato e progettato sulla base di due astrazioni fondamentali: *gli agenti e gli artefatti*.

4.4 Il meta modello A&A (Agents & Artifacts)

Il meta modello A&A (Agents & Artifacts) è un nuovo framework concettuale per modellare e progettare sistemi orientati agli oggetti, esso disciplina l'interazione tra agenti e l'ambiente MAS delegando la proprietà di *situatedness* agli artefatti.

Secondo le moderne teorie sociali e psicologiche come l'Activity Theory (AT) gli artefatti giocano un ruolo fondamentale nel contesto delle organizzazione umane per supportare gli incarichi cooperativi e le complesse attività collaborative.

Gli artefatti sono strumenti sia fisici che cognitivi che sono condivisi e sfruttati dalla collettività per poter raggiungere obiettivi sia globali che individuali.

In altre parole (come già accennato sopra) il meta modello A&A reinterpreta i MAS in termini di due astrazioni fondamentali, gli agenti e gli artefatti, ma gli ultimi lavori sul framework A&A hanno introdotto i *workspaces* come una nuova astrazione nel meta modello.

Di seguito vengono date le interpretazioni di agenti, artefatti e workspace secondo il meta modello.

4.4.1 Gli agenti

A&A riconosce negli agenti l'autonomia come principale proprietà, da un punto di vista della computazione essi incapsulano il controllo, il quale non può uscire dai confini dell'agente stesso, solo i dati possono farlo; esso infatti non lascia mai il controllo, non viene controllato da nessuno [27].

Di conseguenza è possibile accorpare all'agente la nozione di *azione* il cui tipo dipende verso quale soggetto è diretta:

- *azioni interne*: interne all'agente
- *azioni comunicative*: comprende la comunicazione con uno o più agenti con lo scopo di cambiarne lo stato
- *azioni pragmatiche*: come interazioni con l'ambiente

4.4.2 I Workspace

I workspace sono contenitori logici di artefatti che servono per descrivere la topologia dell'ambiente di lavoro fornendo un modo per definire la nozione di località.

Un agente infatti può lavorare unicamente con gli artefatti appartenenti al workspace dove esso è in esecuzione, ma può essere concettualmente situato in una molteplicità di workspace nello stesso tempo, tipicamente distribuiti su differenti nodi della rete.

Sono strutture adattabili che aggregano artefatti, i quali incapsulano le risorse oppure gli elementi di uno spazio di lavoro che si vuole rappresentare così da creare un contesto per le persone che vi lavorano fornendo informazioni su quella certa area e supportando la scoperta di risorse e servizi.

Un ambiente di lavoro, o *working environment*, è concepito come una serie di artefatti dinamici organizzati in workspace; questo ambiente può essere distribuito se un singolo workspace non viene mappato su un singolo nodo della rete ma diffuso su più nodi.

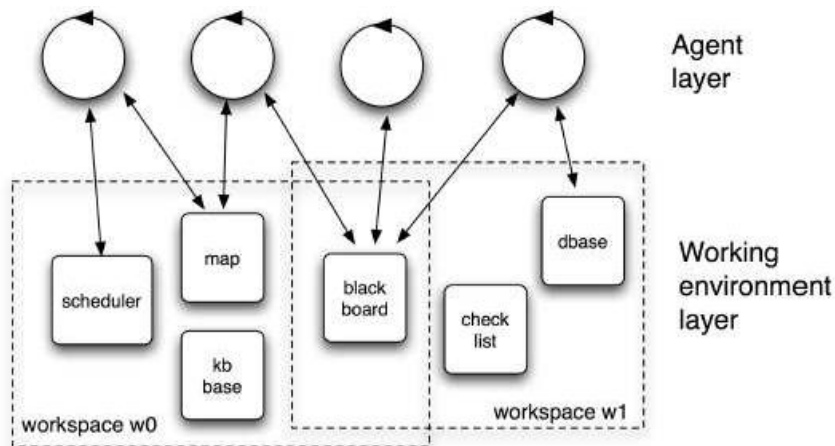


Figura 4.6: agenti che lavorano su diversi working environment

4.4.3 Gli Artefatti

La nozione di artefatto è l'astrazione chiave del modello A&A: essa è destinata a rappresentare una qualsiasi entità appartenente al *working environment* creata condivisa e usata dagli agenti per fargli prendere carico delle proprie attività, specialmente quelle importanti; sono quindi i blocchi costitutivi per comporre un complesso ambiente di lavoro.

Sono quindi entità di calcolo reattive e passive che incorporano diverse funzionalità e informazioni, le quali possono essere sfruttate dagli agenti comunicando tramite una apposita interfaccia.

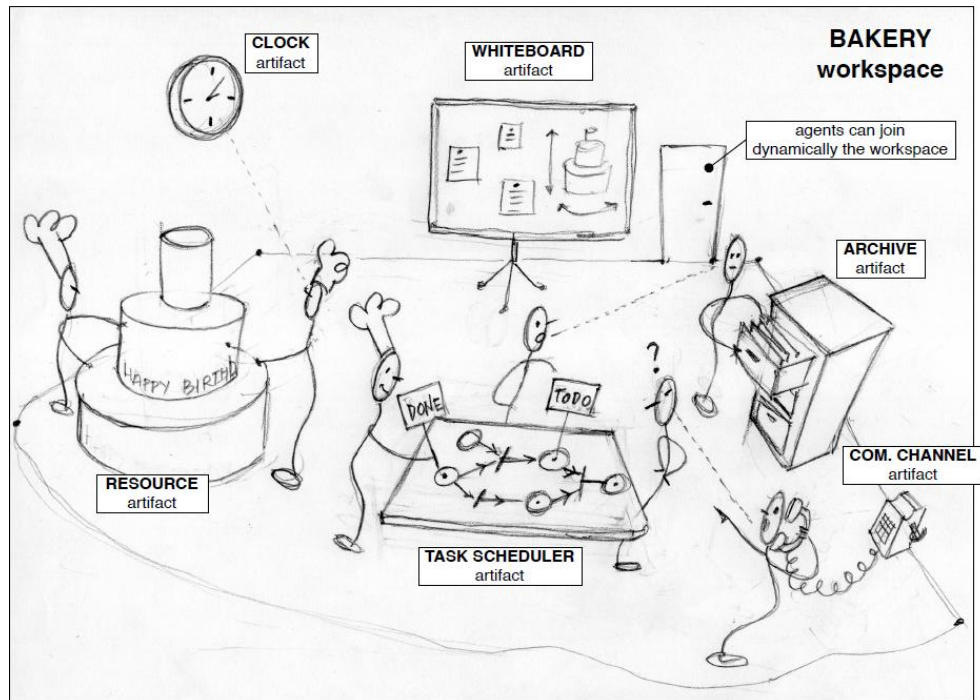


Figura 4.7: esempio di rappresentazione di agenti e artefatti in un panificio

Possono essere quindi identificati quattro elementi base per descrivere un artefatto: l'interfaccia d'uso, le istruzioni operative, la funzione e la struttura ed il comportamento [28].

- Interfaccia d'uso** La relazione chiave tra agenti ed artefatti è la nozione di *uso* oltre alla creazione ed alla distruzione di questi. Per questo la nozione di interfaccia d'uso è definita come *l'insieme base delle operazioni e degli stati ed eventi osservabili che un artefatto espone per poter essere utilizzabili dagli agenti*. Generalmente l'interazione tra agenti ed artefatti avviene come segue: un certo agente esegue delle azioni che determinano l'innesco di alcune operazioni dell'artefatto, le quali portano all'osservazione degli eventi o della evoluzione degli stati dell'artefatto. Dato che gli artefatti sono stati progettati da umani per umani un ruolo chiave lo gioca il manuale di artefatto il quale contiene tipicamente quelli che sono gli scopi dell'artefatto, le informazioni su come usarli e le azioni da prendere in caso di malfunzionamento.

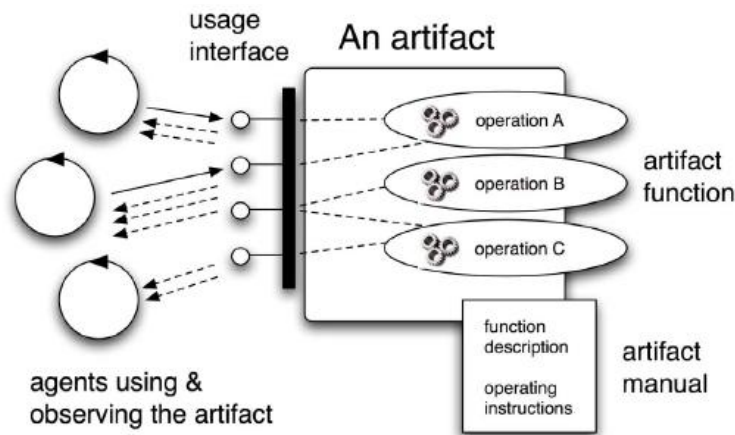


Figura 4.8: interazione agente – artefatto tramite l'interfaccia d'uso

- **Istruzioni operative** rappresentano la descrizione di *come* usare un artefatto per ottenere le proprie funzionalità, descrivono i possibili protocolli d'uso ad esempio la sequenza di operazioni che devono essere invocate su un artefatto per poter sfruttarne le funzionalità; in più possono fornire informazioni riguardanti la sintassi e la semantica per promuovere un uso cognitivo dell'artefatto.
- **Funzione** si intende lo scopo stabilito dai progettisti/programmatori dell'artefatto, ovvero *cosa* sono le funzionalità che l'artefatti intende fornire.
- **Struttura e comportamento** fanno riferimento agli aspetti interni dell'artefatto, come un artefatto è implementato per poter fornire le proprie funzioni. Tale aspetto è tipicamente nascosto agli utenti ed è appartenente al dominio dei progettisti e programmatori.

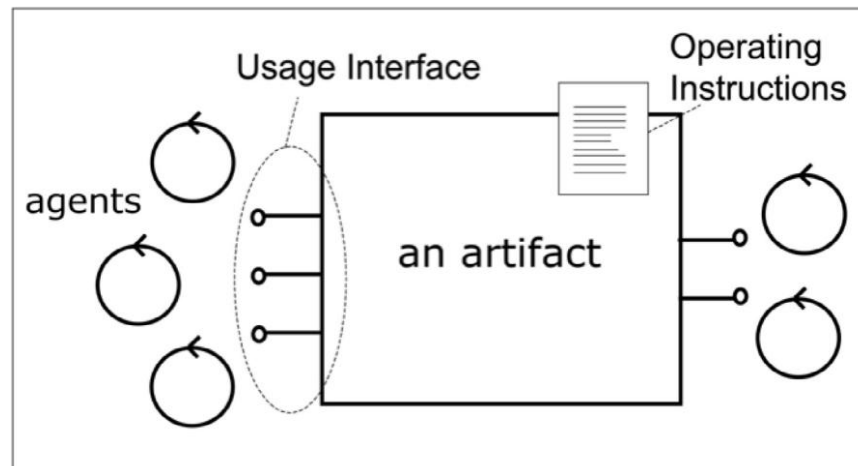


Figura 4.9: rappresentazione astratta di un artefatto

Nonostante il loro specifico modello/architettura, per poter eseguire le proprie azioni sugli artefatti e percepire gli eventi osservabili, gli agenti devono essere situati nell'ambiente di lavoro, *working environment*: per questo la nozione di *agent body* viene introdotta.

L'*agent body* funziona come un medium attraverso il quale l'*agent mind*, ovvero la parte progettata e programmata in base ad una certa architettura, può percepire ed influenzare il *working environment*.

Essi sono dunque necessari per disaccoppiare la mente dell'agente dal *working environment* nel quale l'agente è situato per fare in modo di usare il meta modello A&A con differenti tipi di modelli di programmazione per gli *agent mind*, inclusi sia i modelli di agenti reattivi, mobili e intelligenti.

4.4.3.1 Tipi

Esistono diversi tipi di artefatti che popolano un ambiente ad agenti, i quali offrono diversi insiemi di servizi, racchiudono una varietà di differenti modelli, tecnologie e strumenti ed indirizzano una vasta gamma di questioni applicative.

Per poter supportare lo sviluppo di un ambiente ad agenti da parte dei progettisti, il meta modello A&A fornisce una tassonomia di artefatti, classificando questi secondo il loro ruolo di mediazione; più precisamente sono classificati in base al tipo delle entità (non altri artefatti) che hanno lo scopo di essere legate insieme.

Sulla base di quanto detto, in [27] vengono definiti gli *artefatti individuali*, *artefatti sociali* e *artefatti risorsa*.

- **Artefatti individuali** artefatti sfruttati da un agente, mediano tra il singolo agente e l'ambiente. Essi non sono direttamente interessati alle attività di altri agenti ma possono, tramite link, interagire con altri artefatti appartenenti al MAS.
- **Artefatti sociali** sono, al contrario di quelli sopra, sfruttati da diversi agenti e mediano appunto tra questa molteplicità di agenti e l'ambiente. Generalmente essi forniscono al MAS un servizio che ha in primo luogo lo scopo di raggiungere un obiettivo sociale del sistema, piuttosto che l'obiettivo di un singolo agente.
- **Artefatti risorsa** tipicamente sono artefatti che “avvolgono” risorse esterne e mediano l'interazione tra agenti del MAS e queste risorse. Questo tipo di artefatto si preoccupa di portare al livello di astrazione degli agenti tutte le entità computazionali (e fisiche) che possono essere utili per gli agenti, da oggetti (in scenari di programmazione OOP) a servizi come i web services. Attrezzano tali risorse con interfacce d'uso, istruzioni operative e descrittori di servizio, per poter realizzare le proprie attività mappando interazioni di alto livello tra agenti su interazioni di livello più basso.

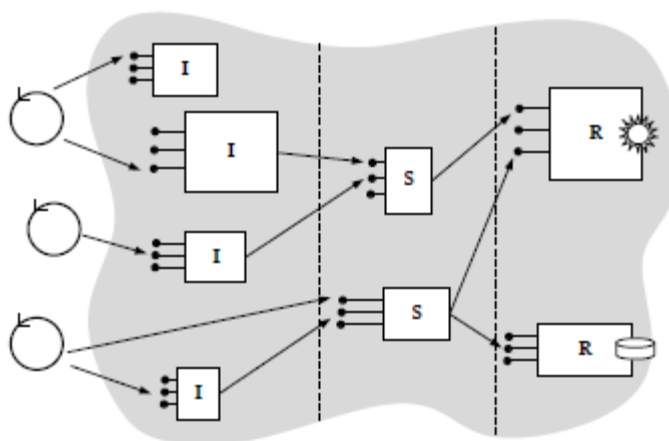


Figura 5.10: vista stratificata ad artefatti di un ambiente MAS

La figura sopra mostra come gli artefatti di più alto livello si prendono carico anche di attività che stanno svolgendo più artefatti individuali.

In conclusione possiamo dire che gli artefatti si comportano come una sorta di *strumento di mediazione* il quale permette *l'interazione agente-agente ed agente-ambiente*.

4.5 MAS nel meta modello A&A

Nello scenario di un sistema multi agente per lo sviluppo di uno Smart Environment la comunicazione diretta ed esplicita non risulta sempre l'approccio migliore per poter raggiungere un comportamento coerente di tale sistema composto da una società di agenti.

In questi scenari il ruolo fondamentale viene giocato dall'ambiente e come i modelli di coordinazione vengano basati su questo (*environment-based coordination*) adottando infrastrutture che si basano sulla interazione mediata, sulla coordinazione/cooperazione senza che vi sia una interazione diretta.

In accordo con AT le attività di cui si fanno carico uno o più componenti –gli agenti– di un sistema multi agente, individualmente o in modo collettivo, non possono essere concepite o comprese senza che venga considerato il ruolo dell'artefatto che media le azioni ed interazioni tra questi componenti.

Gli artefatti di un MAS da una parte le interazioni tra i componenti individuali ed il loro ambiente, dall'altra parte incorporano parte dell'ambiente che può essere controllato e progettato per poter supportare le attività di questi componenti.

Infatti, come una parte osservabile dell'ambiente, gli artefatti possono essere monitorati durante le varie attività per valutare le performance del sistema e per poter tener traccia degli stati della “storia” del sistema [29]. In altre parole gli artefatti di mediazione diventano le entità principali per l'analisi e la sintesi sia per le attività di lavoro individuali che cooperative all'interno del sistema complesso: nel meta modello A&A sarà lecito quindi reinterpretare il medium di coordinazione, vedendolo come un artefatto sociale, un *artefatto di coordinazione* che incorpora le politiche di coordinazione all'interno del MAS.

4.6 Artefatti di coordinazione

Un artefatto di coordinazione è di fondamentale importanza per poter supportare le attività sociali (coordinazione, cooperazione, competizione) che avvengono attraverso l'ambiente abitato da un insieme di agenti dinamici ed eterogenei.

Un artefatto di coordinazione può essere concepito come una entità persistente specializzata nel fornire un servizio di coordinazione al MAS [29].

Con il termine coordinazione possiamo intenderlo in questo contesto, in modo generico, come la gestione delle dipendenze tra attività separate, formando e vincolando lo spazio di interazione tra agenti.

Gli artefatti di coordinazione sono astrazioni infrastrutturali che intendono incrementare l'automazione della coordinazione tra le attività degli agenti; possono essere considerati i blocchi che stanno alla base della creazione di working environment condivisi e collaborativi, alleggerendo così il carico coordinativo che sta sulle spalle degli agenti che operano.

Per questo l'idea di un artefatto di coordinazione risulta simile al concetto di *coordination media*, possono essere fatti dunque alcuni esempi a differenti livelli di astrazione: da artefatti con funzioni di comunicazione (black board, message boxes), a quelli con specifiche funzioni di sincronizzazione (scheduler, semafori) per arrivare a quelli con capacità di coordinazione di alto livello (motori di workflow, infrastrutture di feromoni).

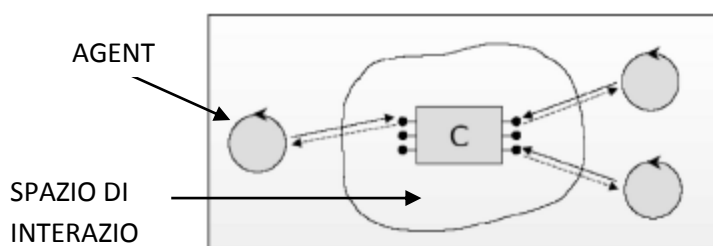


Figura 4.11: *coordination media* come artefatto sociale

Fondamentalmente un artefatto di coordinazione comporta una forma di mediazione tra gli agenti che lo utilizzano, incorpora e mette in atto

alcune politiche di coordinazione; di conseguenza esso mira ad essere una astrazione essenziale per creare e comporre le attività sociali ed a regolare queste attività sociali.

Possiamo dunque elencare le proprietà base che lo caratterizzano [30]:

- *Encapsulating coordination*: l'artefatto incapsula il servizio di coordinazione permettendo così agli agenti di astrarre dal *come* questo servizio è implementato. Esso viene percepito come una entità individuale ma che può essere distribuita su differenti nodi della infrastruttura del MAS
- *Specialisation*: gli artefatti di coordinazione sono specializzati nell'automatizzazione delle attività coordinative, tipicamente adottando un modello computazionale appropriato per una effettiva ed efficiente gestione delle interazioni
- *Malleability*: dovendo gestire la coordinazione in sistemi aperti di agenti dove l'imprevedibilità degli eventi e la dinamicità sono la caratteristica principale, il comportamento coordinativo deve poter adattarsi e cambiare altrettanto dinamicamente
- *Linkability*: si riferisce alla capacità di collegare insieme più artefatti in modo da poter così supportare la scalabilità del sistema insieme alla complessità delle attività di coordinazione
- *Spatial extension*: differentemente dagli agenti l'artefatto di coordinazione può avere una estensione topologica, ovvero lo stesso artefatto può essere presente in differenti nodi della rete, mentre un singolo agente non può essere distribuito

È necessario che forniscano anche un ottimo grado di flessibilità data la vasta eterogeneità delle attività che devono svolgere gli agenti, se viene a mancare anche solo una parte di questa flessibilità si potrebbe avere un impatto drammatico sull'intero insieme delle attività del sistema.

Per applicare l'astrazione di artefatto coordinativo a sistemi multi agente con coordinazione environment-based la nozione di artefatto di coordinazione può essere definita unendo due punti di vista differenti:

- Uno dei ruoli principali di tale artefatto è quello di strumento per la progettazione e lo sviluppo dei blocchi costitutivi specializzati per fornire funzionalità di coordinazione adatte per essere

programmate e configurate secondo lo specifico problema di coordinazione da risolvere

- L'artefatto di coordinazione può essere usato *per rappresentare qualsiasi dispositivo di primo livello che supporti l'interazione tra agenti*; per questo ragionamento quindi ogni Smart Device può essere descritto in termini di artefatto di coordinazione dotato di interfaccia di utilizzo, comportamento coordinativo e a volte, anche con un set di istruzioni operative

In breve, gli artefatti di coordinazione possono essere concepiti come astrazioni di ingegnerizzazione, usate per progettare, costruire e supportare a run time la coordinazione di una società di agenti e fornendo opportuni strumenti al loro working environment.

Concludendo gli artefatti sono attualmente analizzati come i blocchi costitutivi per la programmazione dei MAS, la progettazione dell'ambiente MAS e la rielaborazione della nozione di agente intelligente come utilizzatori di artefatti goal-oriented (progettati per un obiettivo) : come nello scenario umano, gli artefatti non solo amplificano le capacità degli agenti (umani) ma, come entità, possono cambiare la natura delle attività migliorando le prestazioni complessive.

Capitolo 5

Infrastruttura per gli artefatti di coordinazione

L'infrastruttura che supporta la rete di un artefatti coordinativi fornisce servizi per il loro accesso/uso, supportando i livelli di cooperazione e coordinazione di ogni attività sociale; questi servizi vanno dalla creazione e rilevamento di artefatti all'adattamento dinamico del loro stato e del comportamento coordinativo.

In [29] viene riportato la seguente rappresentazione di una possibile infrastruttura a tre livelli per un sistema multi agente:

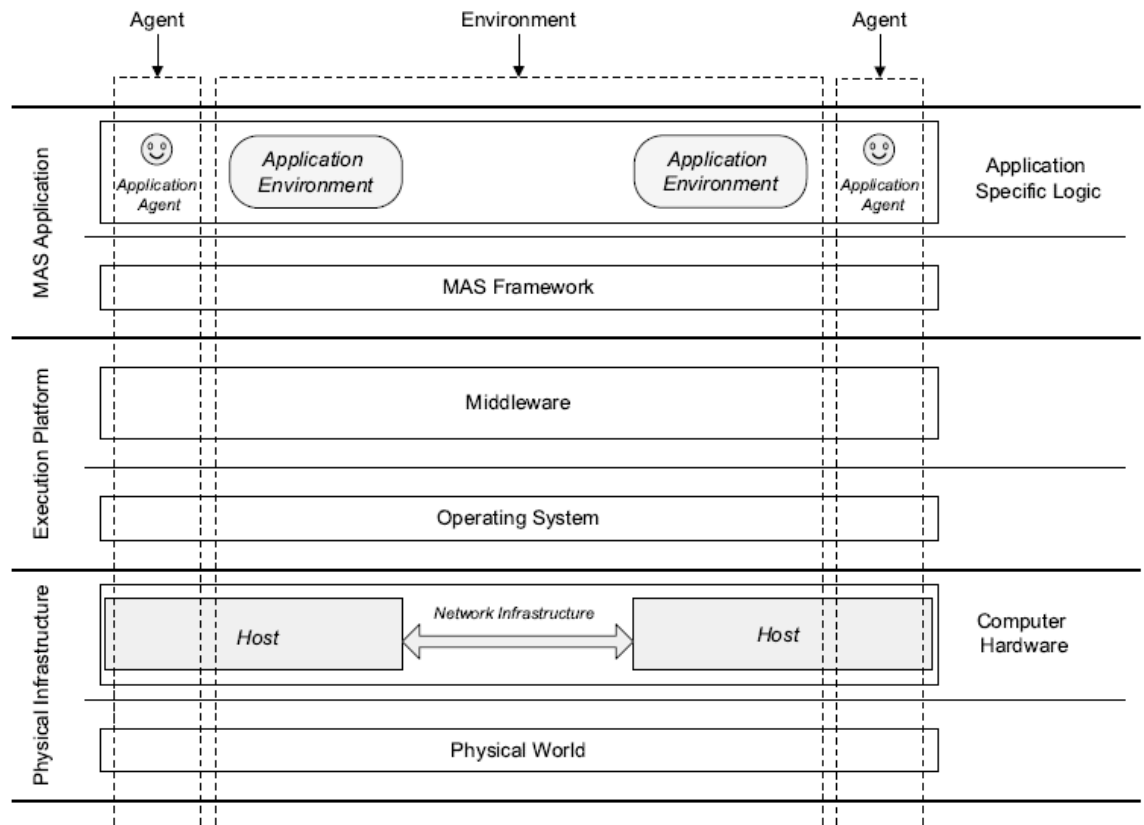


Figura 5.1: struttura a tre livelli per un MAS

ragionando sullo schema sopra l'infrastruttura degli artefatti di coordinazione è parte della piattaforma esecutiva, mentre gli artefatti stessi sono definiti ed usati al livello applicativo del MAS programmato in base alla logica della specifica applicazione.

L'infrastruttura fornisce una astrazione flessibile e robusta per poter modellare e formare lo spazio di interazione tra gli agenti in accordo con gli obiettivi sociali del sistema.

Il coordination media introdotto nel contesto di modelli e linguaggi sarà descritto a livello agente come un artefatto di coordinazione, le primitive di coordinazione definiranno l'interfaccia di utilizzo, mentre il comportamento coordinativo è definito dalle leggi di coordinazione.

In generale gli artefatti coordinativi possono essere visti come l'astrazione fondamentale per la realizzazione di infrastrutture che disciplinano la coordinazione.

Un concreto esempio di infrastruttura/modello che verrà introdotto per poter costituire il framework del sistema MAS per uno Smart Environment è l'infrastruttura di coordinazione TuCSoN che possiede le proprietà principali che caratterizzano la struttura dell'artefatto di coordinazione.

5.1 L'infrastruttura TuCSoN

TuCSoN (Tuple Centres Spread over Networks) [4] è una infrastruttura di coordinazione che supporta la comunicazione e la coordinazione tra agenti fornendo centri di tuple come servizi di coordinazione run time distribuiti attraverso i nodi della rete, dove gli agenti possono inserire, consultare o recuperare informazioni sotto forma di tuple.

La coordinazione è offerta per mezzo di tuple centre poiché sono l'esempio di un modello di coordinazione già esistente per i MAS che esibisce alcune delle proprietà principali descritte per gli artefatti di coordinazione.

In questo contesto i tuple centre possono essere infatti concepiti come degli artefatti di coordinazione general purpose che possono essere personalizzabili

in modo dinamico per descrivere uno specifico comportamento coordinativo, possono quindi essere inquadrati come artefatti di coordinazione dove:

- l'**interfaccia d'utilizzo** è composta dalle primitive di coordinazione (Linda-based) per poter inserire (out), recuperare (in), leggere (rd), ispezionare (get_spec) e settare (set_spec) il comportamento coordinativo del centro di tuple
- il **comportamento coordinativo** è espresso come una applicazione ReSpecT
- le **istruzioni operative** e la descrizione di funzione non sono espresse in modo esplicito: esse sono implicitamente descritte nell'applicazione ReSpecT che definisce il comportamento dell'artefatto

Nella infrastruttura TuCSoN i tuple centre devono essere quindi opportunamente programmati per realizzare artefatti di coordinazione con differenti scopi di coordinazione come: comunicazione flessibile, mediazione tra le conoscenze, condivisione di risorse, etc.

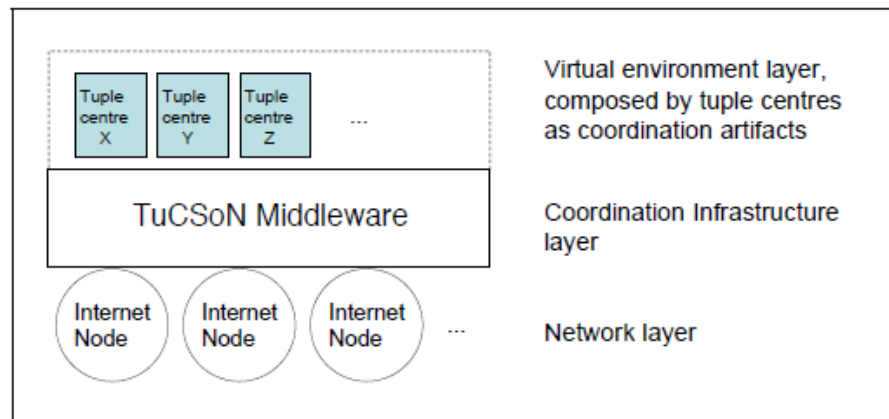


Figura 5.2: vista logica dell'infrastruttura TuCSoN, analoga al modello a tre livelli per i MAS visto in precedenza

Le proprietà rilevanti di TuCSoN comprendono:

- TuCSoN **coordination space**: ha duplice interpretazione, (i) come spazio di interazione globale costituito da astrazioni di comunicazione univoche, oppure (ii) come una collezione di spazi di interazione locali

- **TuCSon coordination media:** astrazione di comunicazione il cui comportamento è definito in modo da incorporare le leggi di coordinazione globali

5.1.1 Coordination space

Lo spazio di coordinazione TuCSon si affida ad una molteplicità di astrazioni di comunicazione indipendenti, i tuple centre, distribuiti sui nodi della rete ed usati dagli agenti per interagire con altri agenti e l'ambiente di esecuzione. Ogni tuple centre è associato ad un nodo della infrastruttura TuCSon ed è localizzato da un identificatore univoco.

Come mostrato nella figura sotto, ogni nodo fornisce la propria versione del TuCSon media space –insieme degli ammissibili identificatori di tuple centre– implementando alla propria maniera ogni tuple centre.

Come definito in ReSpecT ogni tuple centre *tc* fornito in un nodo *node* della rete sarà identificato univocamente dalla sintassi *tc@node*

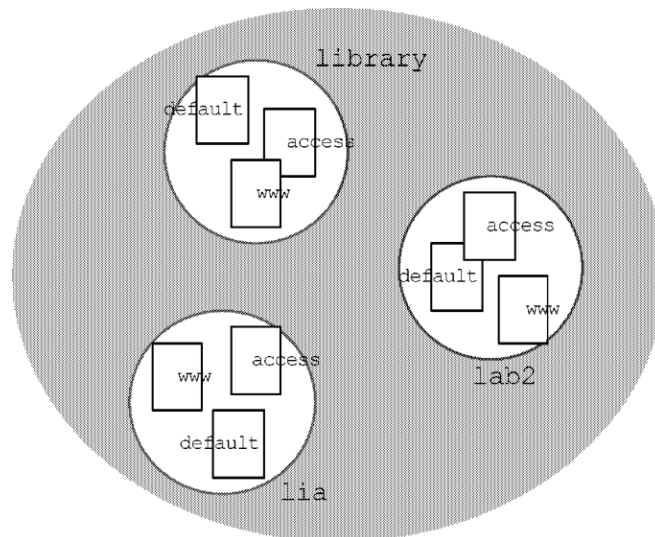


Figura 5.3: esempio di tre nodi TuCSon, ognuno dei quali implementa una propria versione (locale) dello stesso spazio {default, access, www}

Ad esempio, come visto nella figura sopra, il nome access@lia.deis.unibo.it identifica univocamente il tuple centre *access* fornito dal nodo della rete (Internet in questo caso) *lia.deis.unibo.it*.

Tale figura identifica una collezione di spazi di interazioni locali, si potrebbe fornire uno spazio globale dove ogni tuple centre di ogni singolo nodo rappresenta un tuple centre dello spazio globale.

5.1.2 Coordination media

Come detto in capitoli precedenti il tipico problema della coordinazione basata su Linda è il comportamento fissato una volta per tutte degli spazi di tuple.

Per questa ragione TuCSoN sfrutta i centri di tuple –ed anche il linguaggio ReSpecT– come il proprio coordination media.

Ogni nodo fornisce una differente molteplicità di centri di tuple ed in più ognuno di questi tuple centre –il loro comportamento– può essere definito separatamente ed indipendentemente da ogni altro tuple centre in accordo con le specifiche attività di coordinazione.

Così, differenti astrazioni di comunicazione possono incapsulare differenti regole di coordinazione fornendo ai progettisti una più fine implementazione delle politiche di coordinazione globali.

Analizzando dal punto di vista della topologia, gli artefatti di coordinazione TuCSoN sono raggruppati in nodi dell'infrastruttura, distribuiti attraverso la rete ed organizzati in domini.

Tali domini sono caratterizzati da nodi *places* e nodi *gateway*: i primi sono destinati ad ospitare i tuple centre per le specifiche applicazioni/sistemi mentre i secondi ospitano i tuple centre usati per l'amministrazione del dominio mantenendo le informazioni sui nodi *places* –i quali possono appartenere a più domini e possono anche essere gateway essi stessi di un sotto dominio.

Sotto vengono riportate due immagini molto importanti, la prima mostra una vista logica di quello che è un MAS basato sulla infrastruttura TuCSoN nel quale si nota come l'organizzazione della coordinazione in TuCSoN è composta da nodi che ospitano le entità coordinative fondamentali: i tuple centre; a questi nodi ogni agente può accedere tramite una propria interfaccia con le apposite primitive Linda-like.

La seconda immagine mostra come è possibile rappresentare, l'architettura di un sistema multi agente immerso in un ambiente, il quale controlla le proprietà dell'ambiente stesso tramite appositi Smart Device secondo il meta modello ad agenti ed artefatti (A&A) sopra descritto.

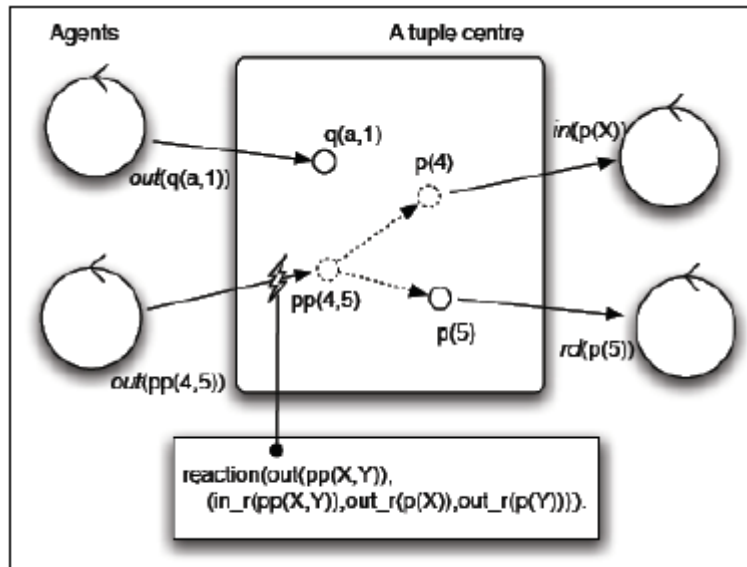


Figura 5.4: interazione tra agenti attraverso un TuCSoN tuple centre.

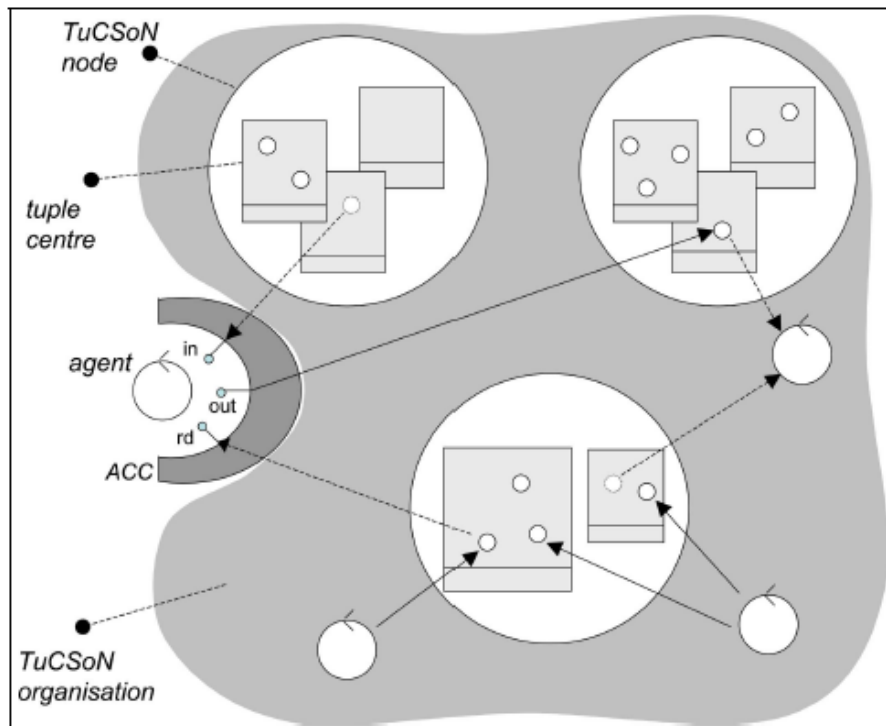


Figura 5.5: vista di un MAS che sfrutta l'infrastruttura TuCSoN

Nell'infrastruttura TuCSoN l'ACC (*Agent Coordination Context*) introdotto in [31] come il luogo concettuale dove stabilire il confine tra l'agente e l'ambiente, incapsula l'interfaccia che permette le azioni e le percezioni dell'agente all'interno dell'ambiente.

Un ACC lavora come un modello per l'ambiente dell'agente descrivendo l'ambiente con cui tale agente interagisce, inoltre abilita e regola l'interazione tra agente ed ambiente definendo lo spazio delle interazioni ammissibili.

5.2 Artefatti di coordinazione in TuCSoN

Così come gli agenti incapsulano competenze ed abilità riguardo all'esecuzione di certe attività, il raggiungimento di alcuni obiettivi o la soluzione ad alcuni problemi, gli artefatti di coordinazione incapsulano strategie, conoscenze ed esperienze per costruire e regolare le attività sociali. Esistono diversi tipi di artefatti di coordinazione usati nella progettazione dei sistemi implementati su TuCSoN:

- **Artefatti di coordinazione per la comunicazione:** forma comune di artefatto utilizzata per fornire servizi di comunicazione, permettendo lo scambio di informazioni tra agenti in un contesto aperto e dinamico il quale richiede un certo livello di disaccoppiamento tra i partecipanti alla comunicazione. In particolare possono essere adottati per supportare la comunicazione se i partecipanti non conoscono gli altri, se non prendono parte alla comunicazione simultaneamente e anche se non appartengono allo stesso contesto spaziale o se ignorano la loro mutua posizione. Ad esempio, artefatti di questo tipo possono essere le *mailbox* che possono essere adottate per supportare il disaccoppiamento temporale e spaziale tra i molteplici mittenti e ricevitori.

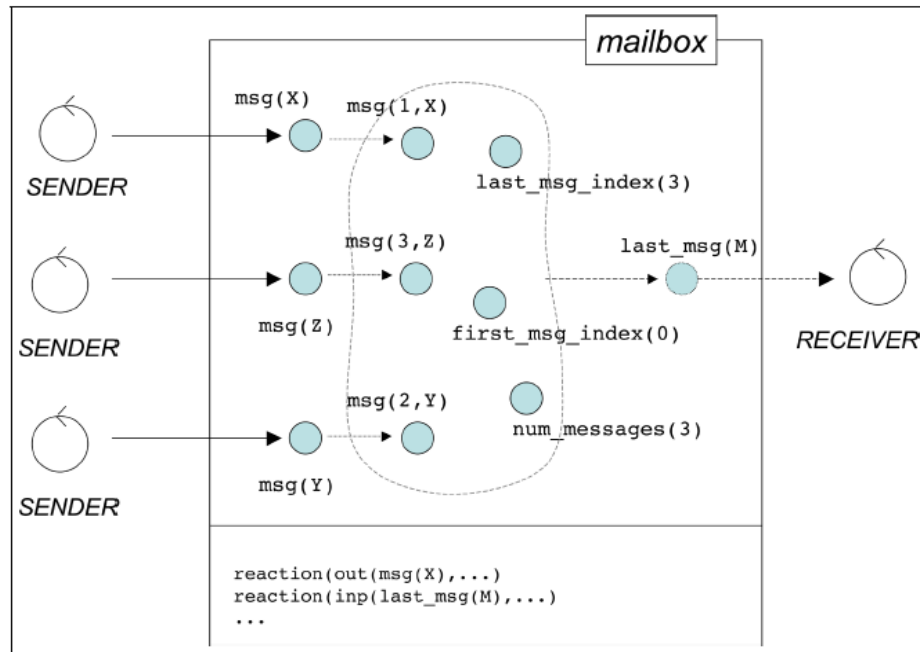


Figura 5.6: mailbox tuple centre e comportamento coordinativo in ReSpecT

- **Artefatti di coordinazione per la condivisione di risorse:** la condivisione di risorse ed attività è tra i più comuni problemi di coordinazione in sistemi distribuiti e concorrenti. Un ambiente di lavoro può essere strumentato con artefatti di coordinazione progettati e programmati per fornire certe politiche di accesso nelle attività o nell'accesso alle risorse, incorporando strategie di sincronizzazione come semafori, monitor, etc.
- **Artefatti di coordinazione per il “workflow management”:** il workflow management consiste nell'automatizzata integrazione e coordinazione di attività eterogenee ed indipendenti coinvolte nello stesso processo globale; oltre alle altre questo include lo scheduling e la sincronizzazione delle attività, la gestione del flusso del controllo e delle informazioni, la gestione delle eccezioni etc.

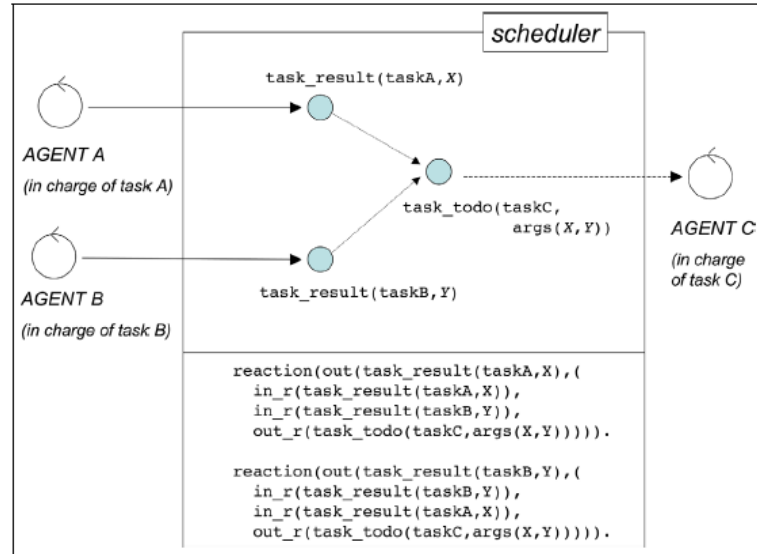


Figura 5.7: scheduler tuple centre

Nella seguente immagine viene mostrato come un motore di workflow (workflow engine) può essere ideato come un artefatto di coordinazione general purpose, il quale è programmato dinamicamente per decretare il comportamento coordinativo a seconda delle specifiche del flusso di lavoro

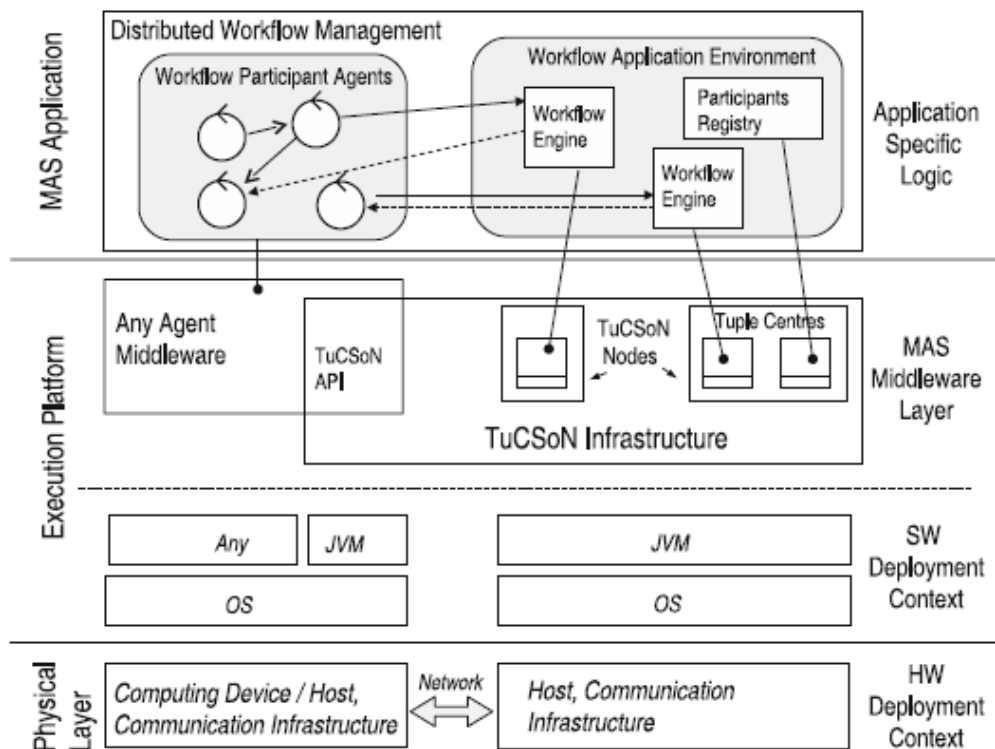


Figura 5.8: applicazione distribuita per la gestione del flusso di lavoro basata su TuCSoN.

Per quanto riguarda l'ultima immagine sotto, viene rappresentato anche il componente denominato *transducer*.

Dato che l'infrastruttura possiede la responsabilità di tradurre gli eventi d'ambiente –fisici– in eventi ReSpecT per poter quindi aggiungere informazioni ai tuple centre, per poter realizzare questa operazione viene introdotto appunto il transducer.

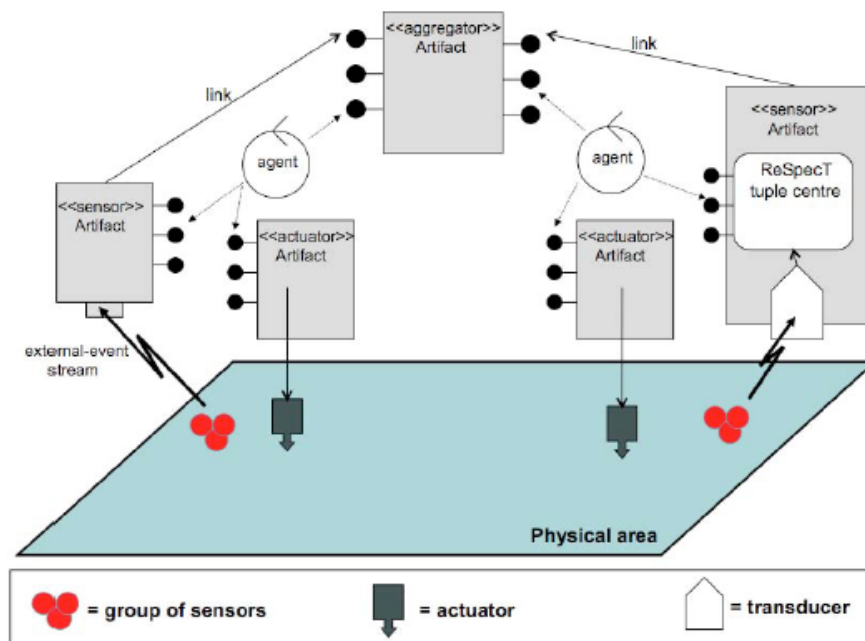


Figura 5.9: *situated MAS nel meta modello A&A*

Quest'ultima rappresentazione chiarisce come andrà strutturato un caso di Smart Environment, dove una rete di sensori immersi nell'ambiente fornisce uno stream continuo di dati che procurano informazioni sul contesto, questi verranno poi opportunamente convertiti in tuple che verranno utilizzate dalla società di agenti per poter prendere le decisioni più adatte ai tipi di traguardi che devono raggiungere applicando sull'ambiente le conseguenti azioni tramite una rete di sensori, anche loro immersi in esso.

5.3 TOTA Middleware

Il diffondersi del calcolo distribuito come ad esempio nelle reti peer-to-peer, o come la mobile/pervasive computing, porta a introdurre nuovi approcci per supportare e facilitare lo sviluppo e l'esecuzione di applicazioni distribuite.

Gran parte dei dispositivi che sono incorporati in oggetti della vita di tutti i giorni, diventeranno intrinsecamente mobili perciò i processi software distribuiti ed i componenti –il termine agente verrà usato per indicare un generico componente attivo di una applicazione distribuita– dovranno effettivamente interagire tra di essi ed orchestrare le loro attività nonostante la dinamicità della rete e dell'ambiente induca alla mobilità.

Gli scenari visti sopra introducono dei requisiti impegnativi per lo sviluppo di applicazioni distribuite, (i) le attività degli agenti software sono strettamente relazionate con l'ambiente nel quale sono situati i dispositivi che li ospitano, ciò consegue che essi devono essere eseguiti in maniera *context aware* acquisendo dinamicamente informazioni sull'ambiente che li circonda, (ii) l'intrinseca apertura e dinamicità di questi scenari prevedono che nuovi dispositivi ed agenti possono arrivare, partire (uscire dal sistema) e morire (smettere di funzionare) in qualsiasi istante, richiede alle applicazioni di essere adattative nelle proprie attività ed interazioni trattando questa dinamicità in maniera autonoma; (iii) i requisiti sopra descritti non devono scontrarsi con il bisogno di promuovere un modello di programmazione semplice che sia possibilmente supportato da una infrastruttura leggera.

5.3.1 Approccio TOTA

I tradizionali approcci per il calcolo pervasivo non possono trovare soluzioni efficienti per le questioni sopra considerate: solitamente gli agenti software sono sprovvisti di informazioni sul contesto –oppure possiedono informazioni poco espressive– le quali vengono richieste esplicitamente da dispositivi esterni, sono strettamente relazionati nelle interazioni così da rendere complesso il supporto e la realizzazione di una interoperabilità e coordinazione adattativa tra di essi; il risultato è quindi solitamente una

infrastruttura di supporto pesante con l'aumento della complessità delle applicazioni.

Il middleware TOTA (*Tuples On The Air*) è un' infrastruttura concepita esplicitamente per supportare i sistemi multi agente in scenari dinamici di rete; gli obiettivi che si pone tale infrastruttura sono [32]:

- *promuovere interazioni adattative e disaccoppiate fornendo agli agenti delle semplici, ma molto espressive informazioni di contesto locali*
- *supportare attivamente questa adattabilità togliendo agli agenti il carico di lavoro utilizzato per trattare con la rete e le applicazioni dinamiche*

basandosi su un middleware TOTA tutte le interazioni tra agenti avvengono in maniera completamente disaccoppiata tramite scambio di tuple.

In questo caso non c'è la nozione di uno spazio di tuple centralizzato condiviso: le tuple possono essere “iniettate” nella rete da qualsiasi nodo (TuCSon node) e può essere propagato e diffuso secondo un appropriato pattern.

5.3.2 Architettura

L'obiettivo nell'adottare tale approccio è quello di avere una interazione context-aware adattabile e disaccoppiata, mantenendo un supporto leggero e di facile programmazione.

Nello scenario TOTA non essendoci un unico spazio condiviso di tuple, le tuple che vengono inserite in rete dai vari nodi formano una sorta di struttura dati distribuita nello spazio che non si limitano unicamente a messaggi trasmessi tra componenti ma generalmente esprimono informazioni contestuali nell'ambiente distribuito.

Per supportare questa idea un ambiente che si basa su TOTA è composto da una rete p2p di nodi (solitamente) mobili, su ognuno dei quali gira una versione locale del middleware TOTA, ciascuno di questi nodi contiene riferimenti ad un certo insieme di nodi vicini.

Su questo spazio dinamico di nodi distribuiti, ogni componente è in grado di memorizzare tuple localmente e poterle diffondere attraverso la rete secondo meccanismo multi-hop (hop-by-hop) sempre in accordo con le proprie regole di propagazione.

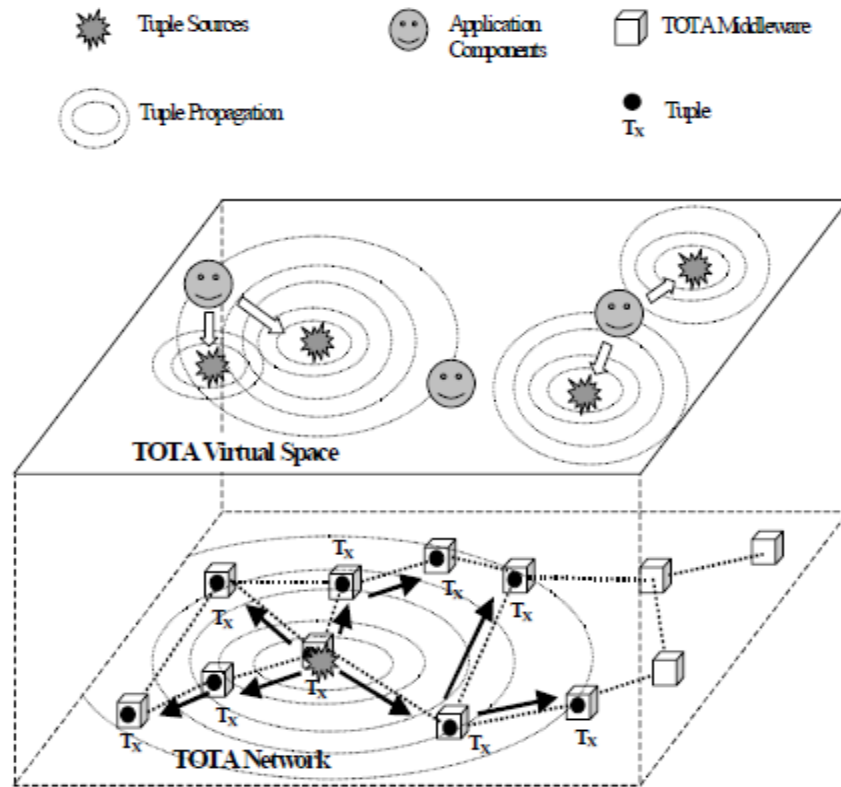


Figura 5.10: scenario generale di TOTA: gli agenti vivono in un ambiente nel quale le tuple inserite sono automaticamente propagate

In TOTA una tupla è definita in termini di (i) il suo contenuto, (ii) le regole di propagazione che determinano come tale tupla deve essere propagata attraverso la rete e (iii) le regole di mantenimento [33]; le regole di propagazione includono:

- una sorta di campo d'applicazione della tupla, ad esempio la distanza alla quale viene propagata (in numero di salti) e la possibile direzione spaziale di propagazione
- gli effetti che ha tale propagazione in conseguenza alla presenza/assenza di altre tuple nel sistema, nelle vicinanze

in più le regole di propagazione possono determinare come il contenuto di una tupla possa cambiare mentre viene propagata in effetti non è necessario che queste siano delle repliche: *assumendo diversi valori in nodi differenti le tuple possono essere effettivamente usate per costruire una struttura dati sovrapposta che esprima certi tipi di informazioni di contesto e di spazio.*

La struttura indotta dalla propagazione delle tuple deve essere mantenuta coerente nonostante la dinamicità della rete, per questo il TOTA middleware supporta la propagazione in maniera attiva ed adattabile monitorando costantemente la topologia di rete e l'entrata di nuove tuple.

Per esempio, se un nuovo nodo entra nella rete TOTA controlla le regole di propagazione delle tuple già memorizzate ed eventualmente provvede a diffonderle sul nuovo nodo; così come se la topologia di rete cambia, a causa dello spostamento di un nodo, la struttura delle tuple distribuite cambia in automatico rispettando la nuova topologia.

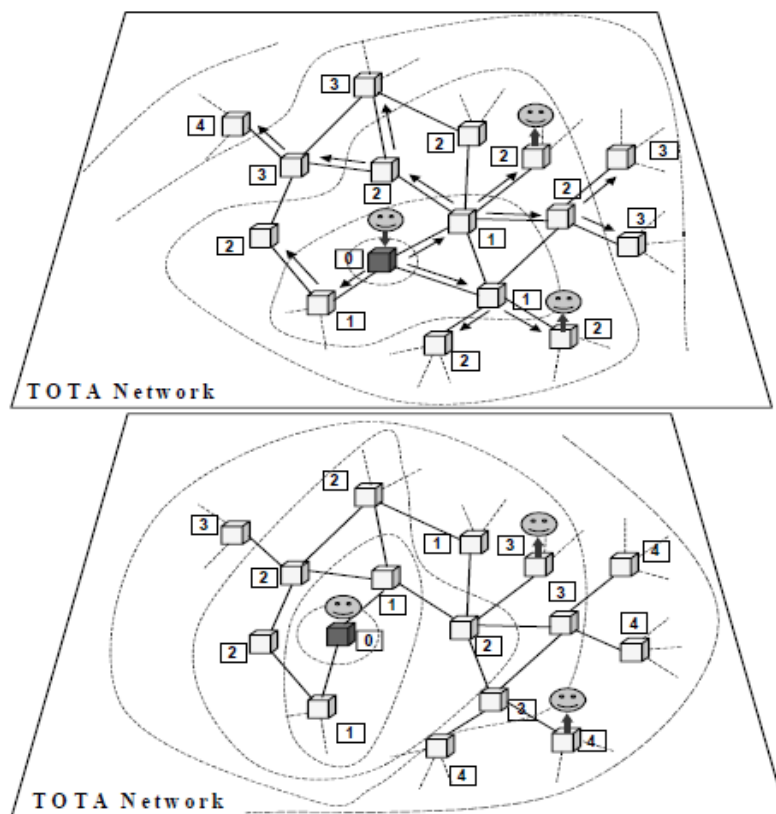


Figura 5.11: aggiornamento della topologia quando viene spostata una sorgente di tuple

I componenti software, gli agenti nel nostro caso, possono inserire nuove tuple nella rete definendo il loro contenuto e regole di propagazione, mantenendo pieno accesso al contenuto locale del middleware –ad esempio il

centro di tuple locale– e possono interrogare il centro di tuple locale oppure quello vicino per verificare la presenza di una specifica tupla.

Oltre a contenuto e regole di propagazione le tuple distribuite TOTA sono caratterizzate da regole di mantenimento che determinano come la struttura della tupla dovrebbe reagire al verificarsi di un evento nell’ambiente; gli eventi catturati da tali regole possono andare dai semplici allarmi temporali ai cambiamenti della struttura di rete (questi sono fondamentali per preservare in modo coerente la struttura delle proprietà d’ambiente, rappresentate dall’insieme di tuple).

Da un punto di vista architetturale il middleware TOTA è costituito da tre parti principali [32]:

- **TOTA API** interfaccia principale tra l’agente ed il middleware fornisce le funzionalità per permettere ad un agente di inserire nuove tuple nel sistema, per accedere a centro di tuple locale oppure
- **TOTA ENGINE** si incarica di spedire le tuple inserite dal livello applicativo (dagli agenti) e di applicare le regole di propagazione a queste per poi propagarle di conseguenza. Si occupa anche di monitorare la riconfigurazione della rete e l’arrivo di nuove tuple aggiornando o propagando di nuovo tuple già memorizzate per mantenere la coerenza della struttura
- **EVENT INTERFACE** è il componente che si incarica di notificare in maniera asincrona l’applicazione dell’arrivo di nuove tuple oppure del fatto che un (nuovo) nodo è connesso/disconnesso ad un nodo vicino

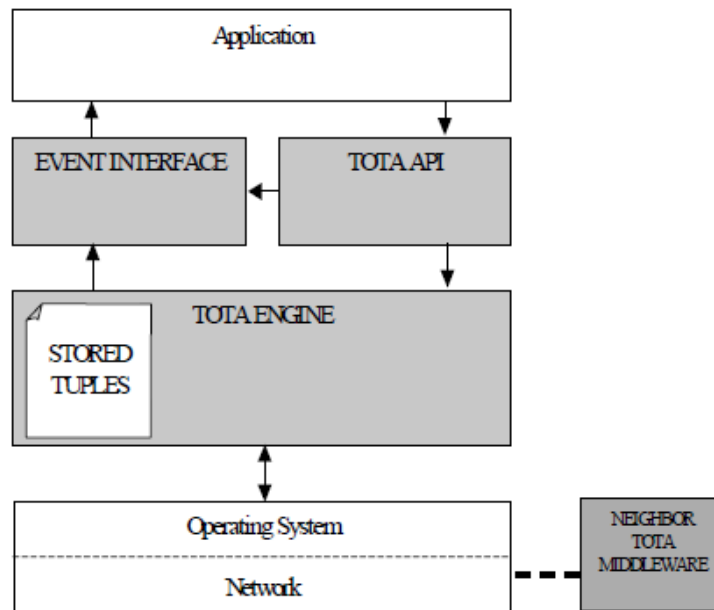


Figura 5.12: architettura del middleware TOTA

5.3.3 Perché TOTA?

La maggior parte dei modelli e dei middleware usati per lo sviluppo di applicazioni distribuite appaiono inadeguate per il supporto delle attività di coordinazione negli scenari di calcolo pervasivo.

È utile quindi mostrare i principali modelli di coordinazione esistenti analizzandone brevemente le caratteristiche deducendo il perché si è ritenuto più appropriato l'utilizzo del middleware TOTA.

Nel modello di comunicazione diretta, una applicazione distribuita è progettata per mezzo di un gruppo di agenti, quali si fanno carico di comunicare tra di loro in maniera diretta ed esplicita (come Jini¹ oppure i sistemi basati sugli agenti FIPA²).

Un problema di questo approccio è che gli agenti che devono interagire direttamente con gli altri, difficilmente possono sostenere l'apertura e la dinamicità degli scenari pervasivi: una esplicita e dispendiosa ricerca di schemi comunicativi deve essere eseguita; gli agenti possono unicamente percepire ed interagire –oppure richiedere servizi– con altri agenti senza una astrazione di contesto di alto livello.

¹ JINI, <http://www.jini.org>

² FIPA <http://www.fipa.org/>

I modelli a spazi di dati condivisi sfruttano le strutture dati localizzate in modo che gli agenti possano raccogliere le informazioni e quindi interagire e coordinare con tutti gli altri agenti.

Queste strutture dati possono essere ospitate da uno spazio dati centralizzato oppure possono essere completamente distribuiti sui nodi della rete; in questi casi gli agenti non sono più strettamente accoppiati nelle loro interazioni, perché ad esempio gli spazi di tuple mediano l'interazione promuovendo il disaccoppiamento.

Tuttavia le informazioni contestuali raccolte da queste strutture dati rappresentano unicamente una descrizione strettamente locale del contesto che può a mala pena supportare il raggiungimento dell'attività globale di coordinazione.

Infine per modelli basati sugli eventi, e modelli *publish-subscribe*, una applicazione distribuita è modellata come un insieme di agenti interagenti con altri mediante la generazione di eventi e la reazione agli eventi di interesse.

Questi modelli favoriscono sia il disaccoppiamento che una rigida *context-awareness*: gli agenti possono infatti essere considerati come immersi in un ambiente attivo, il quale è in grado di notificare ad essi di cosa sta accadendo agli agenti che ne sono interessati –sottoscrivendosi a determinati eventi–.

Tuttavia questo approccio continua a fare affidamento sugli agenti per trattare le azioni coordinate e ciò non allevia il carico che devono sostenere per compiere le loro attività.

Sono stati presentati tutti gli elementi ritenuti necessari per la struttura di un sistema a supporto di uno Smart Environment, dunque il capitolo seguente mostrerà come è possibile progettare il sistema complessivo.

Capitolo 6

Progettazione dello Smart Environment

Scopo dell'elaborato è quello di poter definire uno Smart Environment trattandolo come un caso di studio dei MAS in cui la coordinazione è basata sui centri di tuple ReSpecT, di seguito verrà presentata una possibile soluzione per la sua progettazione.

Lungo lo svolgimento della tesi sono state illustrate quelle che sono le recenti tecnologie utili per il supporto dei sistemi multi agente, ritenute adeguate per lo sviluppo di tali sistemi in modo da poter rendere l'interazione tra i componenti hardware e software il più efficiente possibile mantenendo sempre l'opportuno grado di coerenza fornendo inoltre agli costitutivi le informazioni riguardanti il contesto, lo stato dell'ambiente in cui sono immersi: ci si trova quindi nel classico scenario di pervasive computing.

Mettendo assieme tutte le tecniche esposte nei capitoli precedenti, si potrà definire quello che è la struttura finale del sistema alla base di uno Smart Environment.

Per poter rappresentare il sistema ottenuto si può fornire una rappresentazione stratificata di questo per poi ottenerne il framework complessivo.

Nel capitolo 3 gli agenti sono stati definiti come i componenti software che meglio rappresentano in modo naturale un sistema fornito dallo scenario del calcolo pervasivo.

Perciò nel livello software più alto del sistema ovviamente troveremo le applicazioni MAS e la società di agenti.

Come visto sempre nel capitolo dedicato agli agenti e MAS, la coordinazione tra le entità che interagiscono in maniera continua è un aspetto di fondamentale importanza; è stato scelto di configurare lo spazio di interazione tra gli agenti con un modello di coordinazione

basato su tuple, che favorisce il disaccoppiamento nella comunicazione, in particolare il modello sarà basato sui centri di tuple che contengono anche la possibilità di gestire gli eventi di comunicazione tramite opportune reazioni.

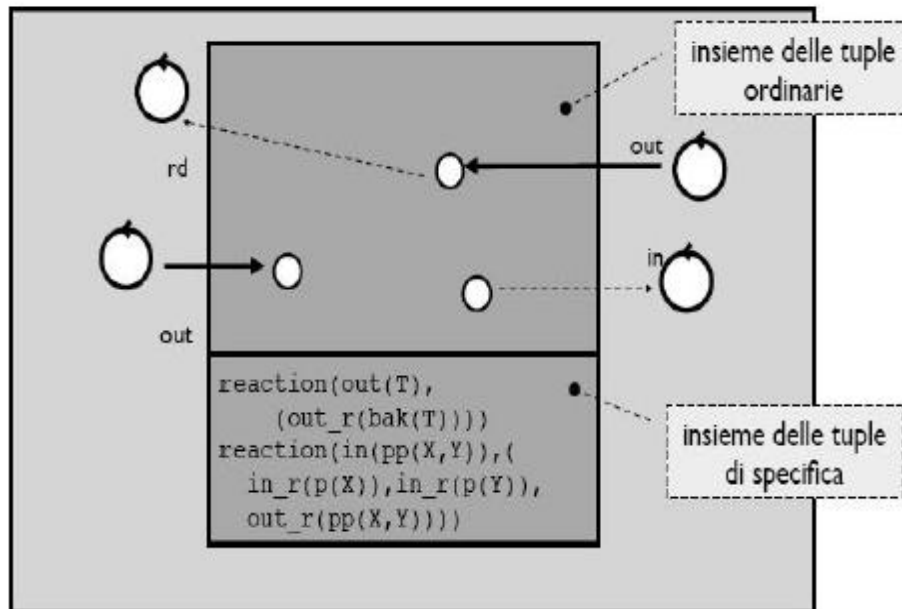


Figura 6.1: comunicazione e coordinazione tra agenti tramite tuple centre.

Oltre a questi eventi sarà opportuno gestire ogni tipo di evento generico che si verifica nell'ambiente essendo i componenti del sistema immersi nello stesso, quindi questa proprietà (situatedness) dovrà essere gestita correttamente.

Introducendo il meta modello A&A nel capitolo 4 questa proprietà del sistema di essere strettamente legato all'ambiente viene delegata agli artefatti, raggruppati in workspace che formano un ambiente di lavoro con il quale gli agenti possono interagire tramite il proprio agent body – definito sempre nello stesso capitolo.

Così si ha una rete di agenti interagenti tra loro tramite tuple centre e dove alcune attività sono delegate ad apposite entità.

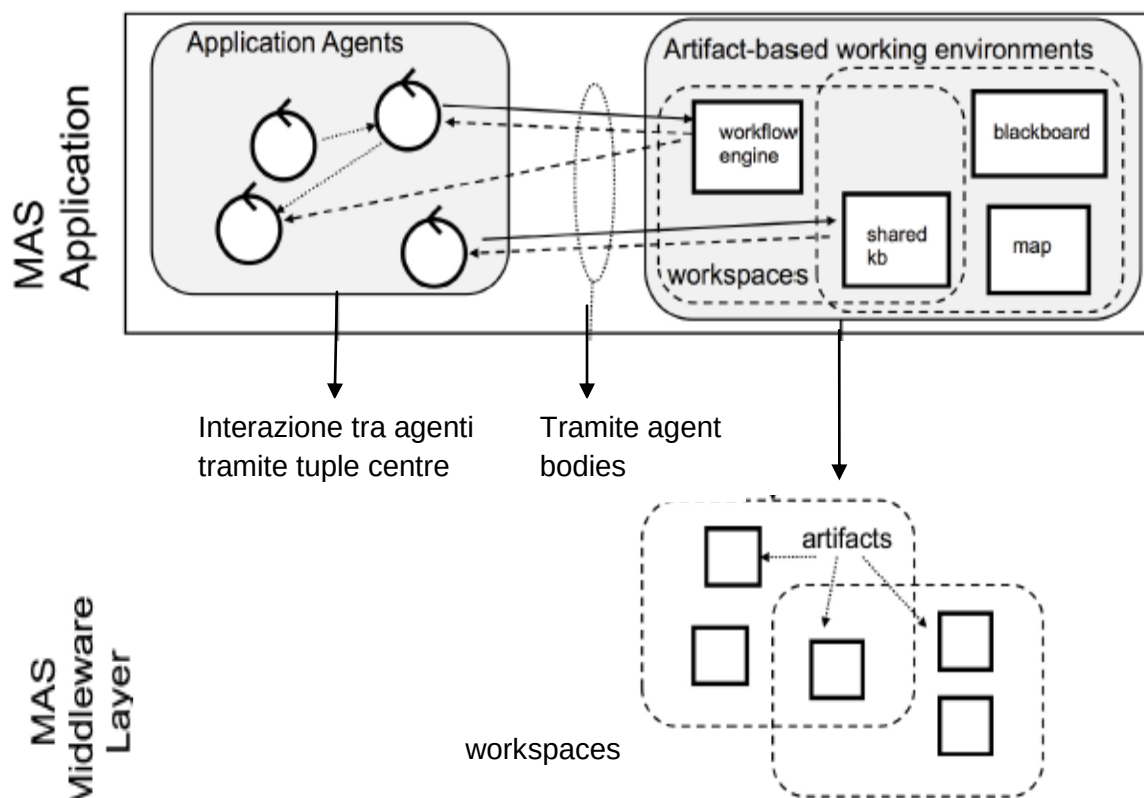


Figura 6.2: cooperazione tra agenti tramite tuple centre ed artefatti.

Il principale tipo di artefatto trattato nel corso dell'elaborato è l'artefatto di coordinazione: esso identifica un artefatto sociale che è usato nel contesto delle attività collaborative, in particolare mediando l'interazione tra gli agenti coinvolti nel medesimo contesto sociale; quindi è ritenuto necessario reinterpretare la nozione di medium di coordinazione dello spazio di comunicazione con il concetto di artefatto di coordinazione facendo così in modo che il carico coordinativo venga tolto dalla spalle degli agenti.

Successivamente ci si è posto il problema di come l'ambiente nel quale questi artefatti di coordinazione lavorano venga strutturato: nel sesto capitolo è presentata l'infrastruttura TuCSon che permette di poter combinare quanto detto prima –dove l'interazione tra agenti di uno Smart Environment è gestita dall'astrazione dei centri di tuple– con quanto introdotto dal meta modello A&A in cui il coordination media è inteso come un artefatto sociale.

Nel contesto TuCSoN i centri di tuple sono assimilabili ad artefatti di coordinazione personalizzabili dinamicamente a seconda degli obiettivi, in tale infrastruttura ogni nodo della rete è caratterizzata da un insieme di tuple centre che possono essere sfruttati dagli agenti distribuiti.

L'infrastruttura TuCSoN può essere considerato come il livello che sta alla base della società di agenti, artefatti e workspace: la struttura che sta alla base del MAS.

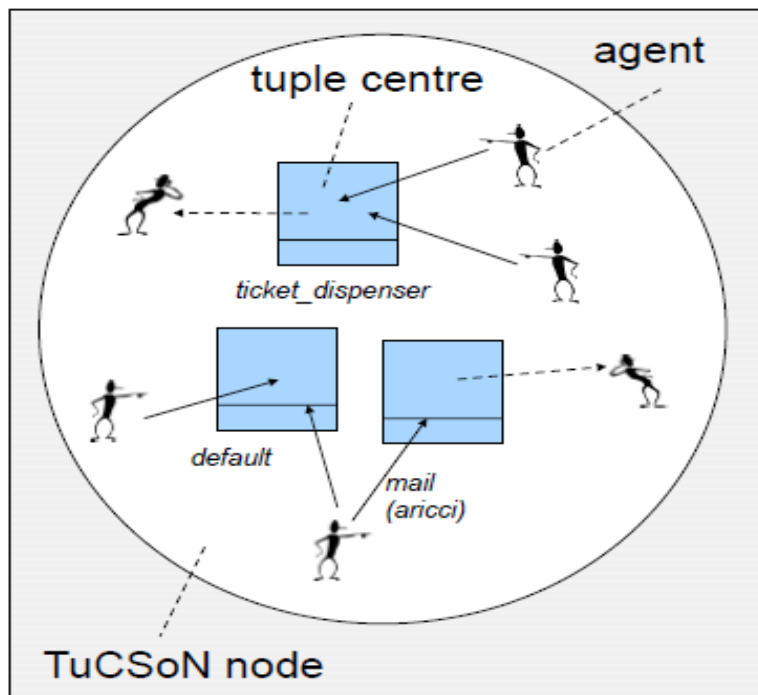


Figura 6.3: struttura di un nodo TuCSoN.

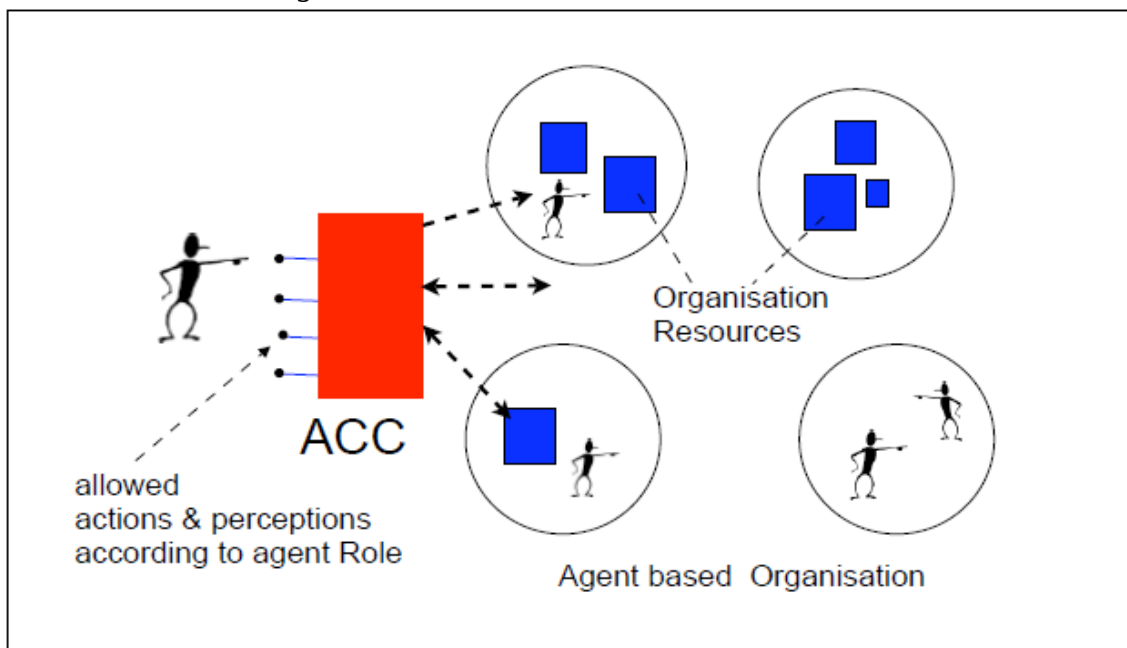
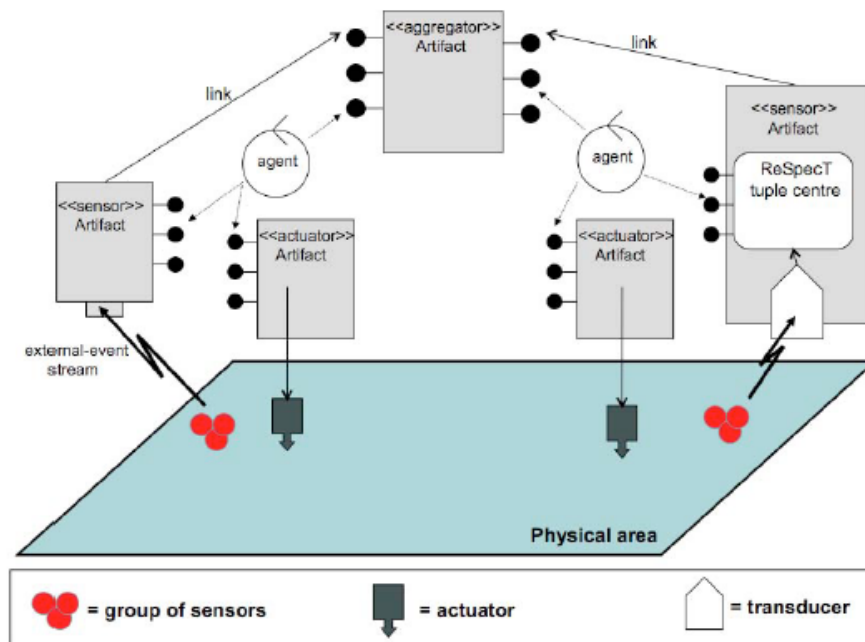


Figura 6.4: MAS framework definito da TuCSoN.



Come trattato sempre nel capitolo precedente, lo sviluppo di applicazioni

Questo middleware permette dunque di coordinare l'interazione tra

meccanismo multi-hop tra i nodi della rete –i quali possiedono riferimenti a nodi vicini.

Affrontare la questione riguardante la dinamicità di rete di uno Smart Environment è necessario dato che la topologia può cambiare continuamente ed in qualsiasi istante: se ad esempio un certo dispositivo per la rilevazione delle persone che entrano in una stanza si decide di spostarlo in un altro locale dell’edificio, l’ automatica riorganizzazione della topologia di rete è evidentemente una importante proprietà.

Si conserva dunque la struttura TuCSoN dove la rete è composta da una serie di nodi ognuno dei quali mantiene localmente un certo insieme di centri di tuple, ma questi non sono esclusivi di tale nodo, infatti possono essere visitati dai nodi vicini per verificare la presenza di una o un insieme di tuple e poter eventualmente determinare le proprie regole di propagazione.

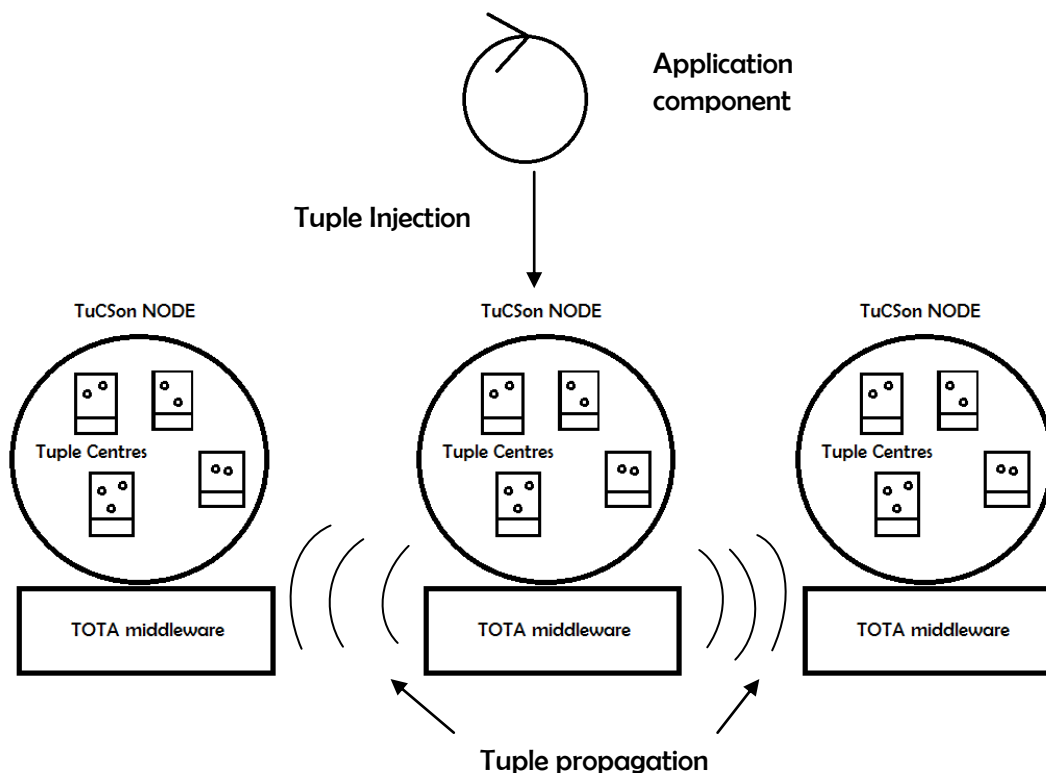


Figura 6.6: integrazione TuCSoN-TOTA, propagazione delle tuple inserite dagli agenti ai nodi vicini.

La base su cui si poggia l'intero sistema è –come detto nel capitolo 2– l'ambiente sul quale opera ed è immersa la rete di Smart Device (sensori ed attuatori) tra loro interagenti e cooperanti; tale rete rappresenta lo strato di livello più basso –quello fisico– che si occuperà di recuperare le informazioni del contesto ambientale, lo strato di rete poi fornisce servizi di connessione tra le reti di sensori smistando le informazioni da esse acquisite per poi spedirle agli strati superiori dove vengono trattati.

Per far sì che venga supportata la possibile mobilità, anche elevata, di questi dispositivi e la scalabilità di tali sistemi di dispositivi grazie alla disponibilità sempre più elevata di collegamenti senza fili, risulta maggiormente performante utilizzare connessioni wireless –in prevalenza, poi alcune apparecchiature potrebbero avere approcci ibridi di comunicazione se la loro presenza nel sistema si ritiene essenziale e stabile– che coprano un campo in prevalenza non eccessivamente esteso, quindi utilizzando tecnologie che vanno dal più semplice Bluetooth, ZigBee oppure standard 802.11a/b/g IEEE per WLAN; scelta che verrà poi presa a seconda della frequenza con cui andranno trasmessi i dati e dei consumi attesi.

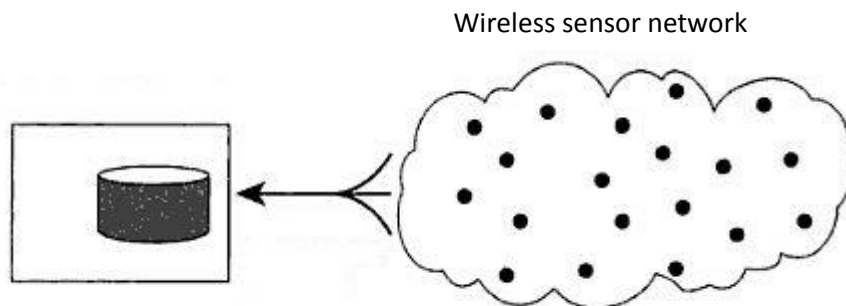


Figura 6.7: esempio di come una rete di sensori invia informazioni relative al contesto di una certa area dell'ambiente.

Dall'immagine sopra si conviene che, per una certa area dell'ambiente intelligente il gruppo di sensori relativo a tale sezione, invia le informazioni percepite ad un nodo centrale che raccoglie i dati relativi ad una certa porzione d'ambiente e quindi passarli alla postazione centrale sulla quale risiedono la totalità delle informazioni sull'ambiente.

Naturalmente, tra middleware ed i due strati fisici si troverà il livello software che ospiterà il sistema operativo –con i relativi servizi offerti–

che sarà appunto il livello che si occuperà di recuperare i dati acquisiti dalla rete di sensori e di inoltrare le operazioni di modifica dello stato ambientale che sono stati determinati dai livelli applicativi.

È quindi possibile schematizzare come il sistema finale alla base di uno Smart Environment può essere progettato, secondo una visione stratificata.

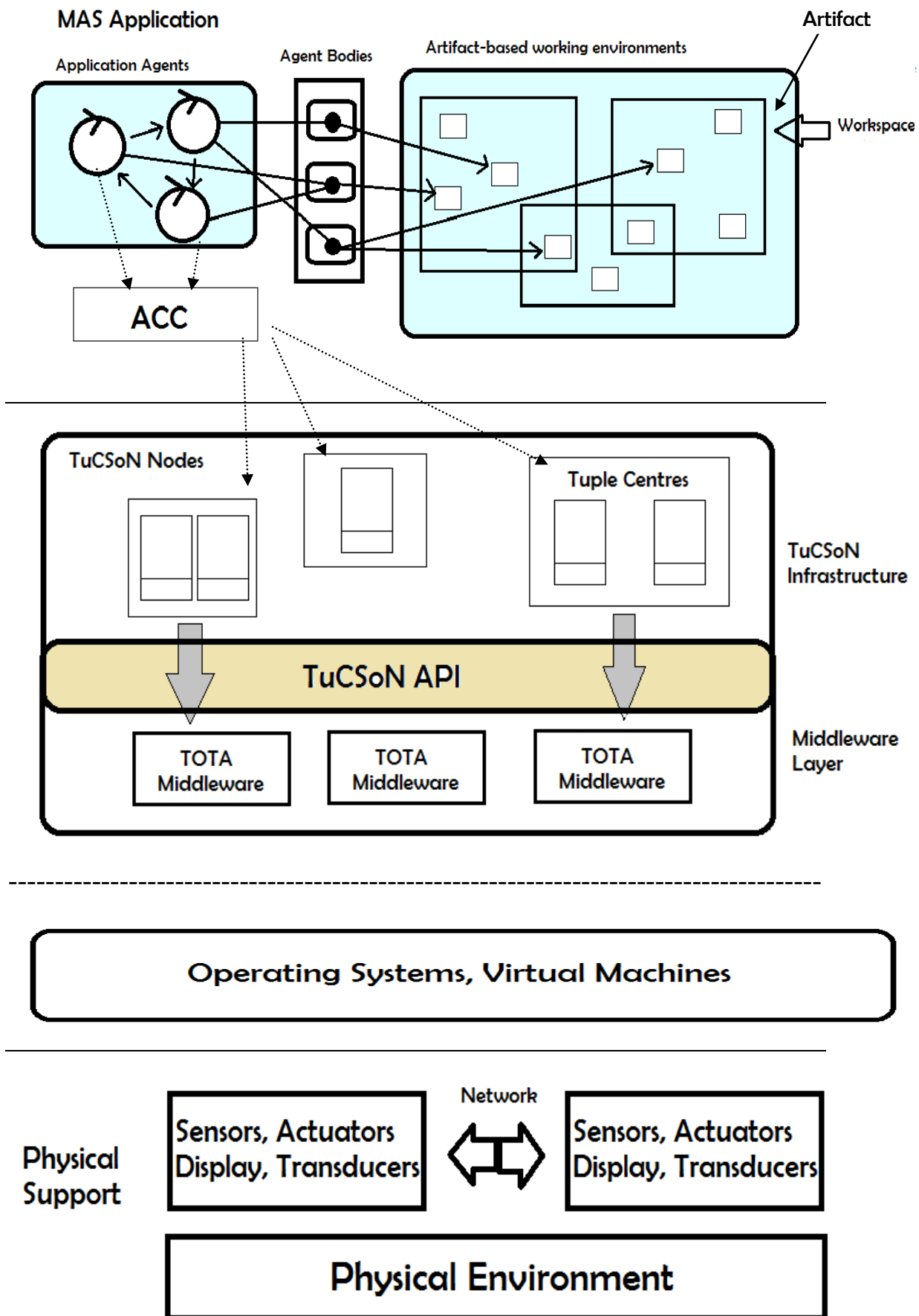


Figura 6.8: visione finale del sistema.

Capitolo 7

Case study

Dopo aver presentato un quadro generale dell'infrastruttura risultante alla fine dell'elaborato risulta utile fornire un semplice caso di studio che dia una idea pratica di come è possibile rappresentare uno scenario della vita quotidiana tramite Smart Environment.

Ad esempio, per l'uso di artefatti coordinativi in MAS basati su centri di tuple TuCSoN, possiamo considerare un' applicazione pervasiva che fornisce un insieme di servizi ai clienti che frequentano una concessionaria di automobili nuove e usate, che offre anche il servizio di autonoleggio; i clienti della concessionaria saranno dotati di dispositivi mobili come ad esempio PDA oppure smart phone, i quali permetteranno agli utenti di poter interagire con l'ambiente.

L'ambiente intelligente rappresentato dallo show room mira ad offrire servizi utili ai potenziali acquirenti aiutandoli a vedere le loro auto preferite offrendo un utile supporto mostrando le caratteristiche tecniche ed assistendoli in un possibile acquisto valutando quelle che sono le reali possibilità di investimento, anche scegliendo il tipo di finanziamento più consono alle proprie disponibilità.

Ogni volta che un cliente entra nella concessionaria un display mostra l'insieme delle macchine compatibile con le proprie preferenze insieme alla loro posizione di queste nella mappa della struttura.

Occorre dunque pensare a quelli che possono essere i possibili parametri di ricerca di un'auto che rappresentino dunque le preferenze di un certo utente. Innanzitutto la volontà del cliente di acquistare una automobile nuova oppure usata, o ancora semplicemente noleggiarne una, nel caso in cui si cerchi un'auto nuova il display potrebbe mostrare anche auto dalle caratteristiche simili ma usate, nel caso in cui sia presente un'offerta ritenuta vantaggiosa dalla concessionaria.

In più il cliente potrebbero essere direzionato a comprare un certo (i) tipo di vettura: berlina, city car, compact car, crossover, SUV, station wagon... (ii) di una certa cilindrata (non maggiore ad un 1900 cc), (iii)

che sia alimentata da un certo tipo di carburante (diesel, benzina, ibrida, con impianto GPL...) , (iv) potrebbe preferire a priori una certa casa automobilistica, (v) potrebbe essere interessato ad un'auto con disponibilità immediata o comunque disponibile in tempi ragionevolmente brevi, (vi) che rientri in una certa fascia di prezzo..

Una volta che il cliente ha composto uno schema che descriva nella maniera migliore le proprie preferenze, queste sono acquisite dall'agente software che gira sul proprio dispositivo, il quale può comunicare tali informazioni al nodo centrale del sistema che è la concessionaria che offre i servizi in modo da poter così proiettare sul display le informazioni maggiormente utili.

Sarà inoltre possibile usufruire di un servizio di segreteria che si occuperà di fissare eventuali colloqui oppure semplicemente per prenotare la prova su strada di una certa auto: tale servizio sarà modellato da un apposito agente che tiene conto degli impegni su di una agenda, rappresentabile come un artefatto di risorsa.

In questo tipo di scenario, aperto e dinamico, è importante anche l'aspetto riguardante il disaccoppiamento tra gli agenti dei clienti ed i servizi offerti oltre che specificare regole di coordinazione che gestiscano le interazioni.

In questa maniera, l'attività dell'agente del cliente si limita ad acquisire le informazioni sulle preferenze ed inviarle al servizio della concessionaria senza preoccuparsi delle regole di coordinazione coinvolte per poter ottenere sul display le automobili che meglio ricalcano le intenzioni d'acquisto del cliente.

È per le ragioni sopra riportate che possiamo sfruttare l'infrastruttura TuCSoN per poter strutturare i servizi offerti dall'ambiente intelligente rappresentato dalla concessionaria tramite centri di tuple TuCSoN, che insieme agli agenti software che risiedono sui dispositivi dei clienti ed agli artefatti dei servizi formano il sistema complessivo.

Saranno presenti numerosi tuple centre distribuiti attraverso la rete su più nodi TuCSoN, nel nostro sistema il nodo centrale contiene il centro di tuple che memorizza l'identità degli agenti dei clienti associando ad essi le preferenze, inoltre per ogni area dello show room sarà presente un nodo che descriverà le automobili presenti in tale sala.

In questo scenario, all'ingresso di un acquirente nella concessionaria l'agente che sta girando sul proprio dispositivo provvedere ad immettere nel tuple centre del nodo centrale (*central_node*) la tupla rappresentativa di un utente, la quale incapsula le proprie preferenze:

```
central_node_tc @ 'central_node_address' ? out(user(userID,  
'Car'(hasMaxSpeed(greater, 160); compare(hasMaker, {renault,  
volkswagen}); hasPrice(less, 18000))))
```

```
central_node_tc @ 'central_node_address' ? out(user(userID,  
'Car'(displacement(greater, 2000); compare(poweredBy, {GPL,  
Diesel}); booleanCompare(nowAvailable, yes))))
```

tali tuple mostrano come le preferenze di un certo utente –che viene identificato da un certo ID– dove la prima tupla indica un'auto che abbia una velocità massima di 160 km/h, che costi non più di 18000 euro e che appartenga alla casa automobilistica Ford oppure alla Volkswagen. Mentre nel secondo esempio, si indica una automobile alimentata a GPL oppure a motore diesel, che abbia un motore di 2000cc di cilindrata e che sia subito disponibile in concessionaria per l'acquisto senza dover aspettare per ordinarla.

Nella prima parte dell'espressione vista sopra, la sintassi TuCSoN specifica che tale tupla verrà inserita nel tuple centre presente nel nodo centrale dell'infrastruttura.

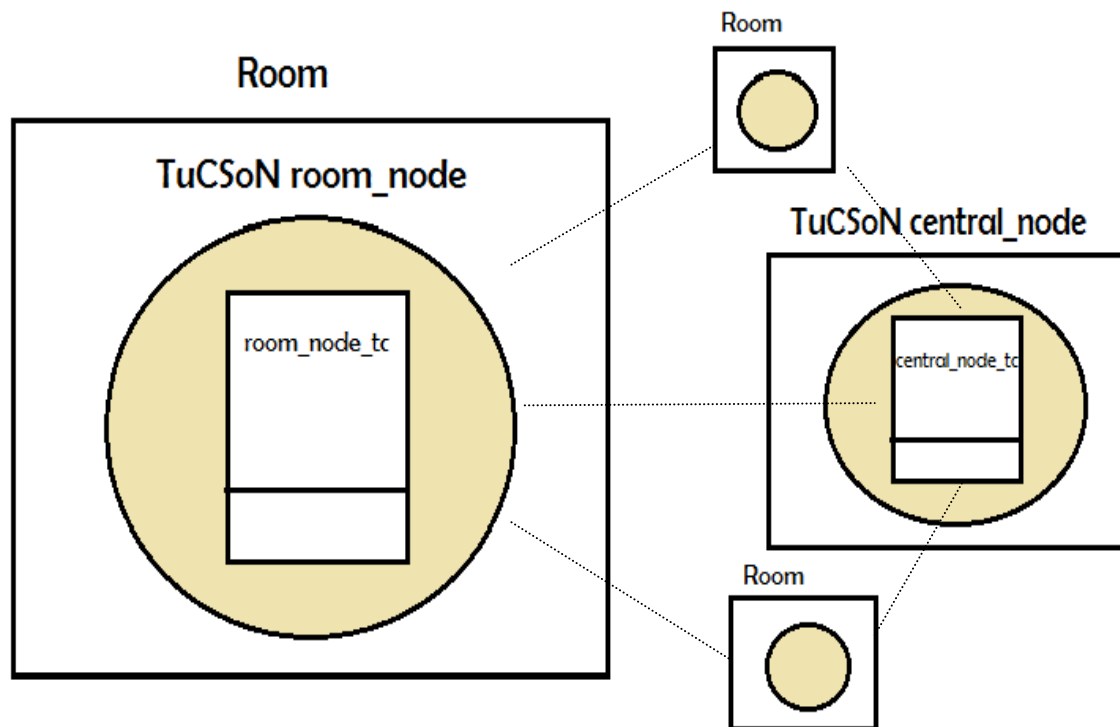


Figura C.1: organizzazione TuCSoN della concessionaria.

Quando poi un utente entra in una certa sala della concessionaria, un sensore rileverà la presenza del dispositivo –contenente un proprio tag – associato ad un determinato userID e, per mezzo di un sistema RFID (Radio Frequency Identification), provvederà a inserire nel tuple centre del nodo della sala (room_node) la seguente tupla:

entered_user(userID)

che rappresenta il cliente effettivamente presente all'interno di una certa stanza; analogamente quando il sensore rileva che l'utente è uscito dalla sala tale tupla sarà rimossa –con una operazione di **in** che faccia match con l'ID associato al dispositivo in questione.

Dunque tornando a quanto detto nel precedente capitolo riguardante la progettazione del sistema avremo lo strato fisico dell'architettura, rappresentato da una infrastruttura di lettori RFID, impostati per poter rilevare PDA o smart phone –ognuno opportunamente equipaggiato da un tag RFID– che entrano nell'area di una certa sala dello show room.

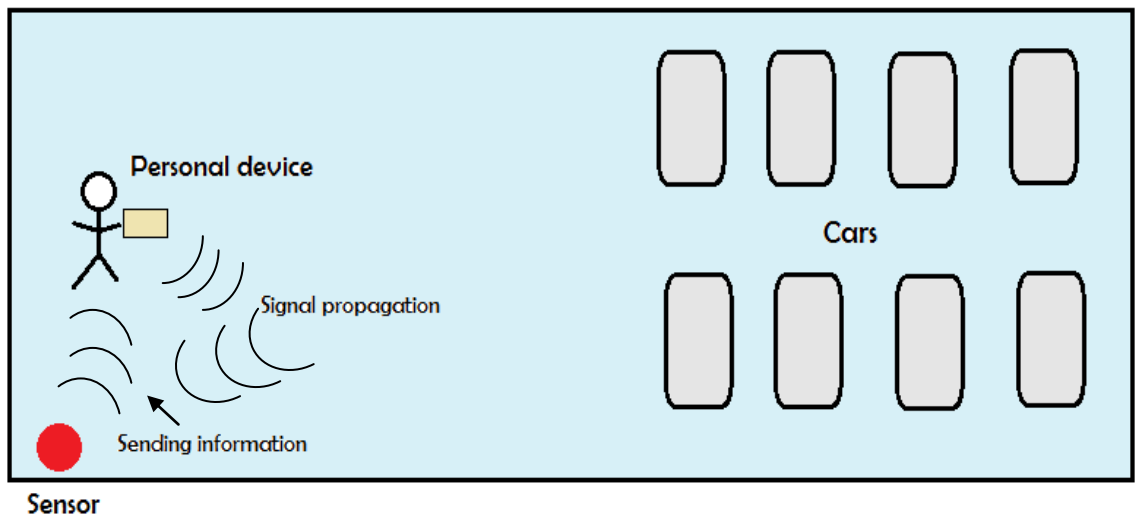


Figura C.2: enfatizzazione di cosa accade quando un utente entra in una sala della concessionaria. Il sensore rileva la sua presenza grazie ad un apparecchio di lettura RIFD, ed invia sempre a esso le informazioni sulle auto di interesse.

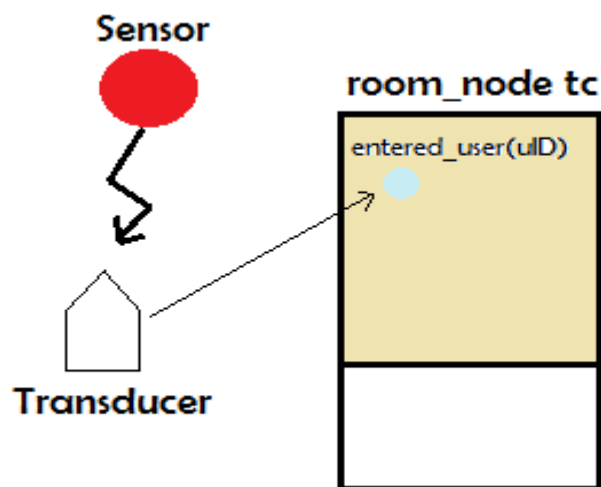


Figura C.3: come trattano nel capitolo 3, la proprietà di *situatedness* richiede anche che, a livello di infrastruttura, gli eventi d'ambiente come può essere la rilevazione di un nuovo ingresso siano tradotti in eventi del centro di tuple

Lo Smart Environment deve dunque fornire i servizi opportuni appena un utente entra in una sala: perciò si può raffigurare l'ingresso di questi come un evento ambientale e quindi gestirlo tramite una serie di reazioni.

Si suppone che il tuple centre presente in ogni room-node contenga anche l'opportuna tupla di specifica di comportamento:

```
reaction(out(entered_user(uID)),
  (operation, invocation),
  (
    central_node_tc @ central_node_address ?
    rd(user(uID, preferences)),
    in(rd-all(matching(preferences,
CarList))),
    write_display(CarList)
  )
)
```

Per mezzo di questa specification tuple, all'ingresso di un utente e quindi quando viene scritta nel *room_node_tc* la tupla *entered_user* sono eseguite una serie di operazioni: (i) la variabile *uID* assume il valore dell'identificatore dello specifico utente *userID*, (ii) la tupla *user(...)* è letta dal nodo centrale e la variabile *preferences* assume i valori specificati nella descrizione delle caratteristiche desiderate, (iii)

l'operazione **rd-all(matching(preferences,CarList))** altro non fa che leggere e restituire tutte le tuple –che rappresentano una autovettura– del'insieme *CarList* che fanno match con il template rappresentato dalle preferenze –*preferences*– dell'utente definite in *'Car'(hasMaxSpeed(greater, 160); compare(hasMaker, {renault, volkswagen}); hasPrice(less, 18000))* e infine (iv) la lista di auto identificate sarà inviata al display tramite il comando *write_display*.

Nel caso in cui il tuple centre del *room_node* contenga le seguenti tuple che descrivono la auto presenti in quella determinata sala:

```
ka: 'CityCar'(  hasMaker=ford,
                hasMaxSpeed=135,
                displacement=1200,
                hasPrice=8000,
                nowAvailable=yes,
                poweredBy=gasoline)
```

```
polo: 'CityCar' ( hasMaker=volkswagen,
                  hasMaxSpeed=145,
                  displacement=1400,
                  hasPrice=15000,
```

**nowAvailable=no,
poweredBy=diesel)**

**tuareg: 'SUV' (hasMaker=volkswagen,
hasMaxSpeed=260,
displacement=3000,
hasPrice=51000,
nowAvailable=yes,
poweredBy=diesel)**

**twingo: 'CompactCar' (hasMaker=renaut,
hasMaxSpeed=130
displacement=1100,
hasPrice=7000,
nowAvailable=yes,
poweredBy=gasoline,GPL)**

successivamente all'ingresso di un utente con certe preferenze dunque, vedrà apparire sul display del proprio dispositivo le informazioni riguardanti solamente la Volkswagen polo, non avendo tra le preferenze la Ford. Mentre il secondo tipo di utente vedrebbe apparire le informazioni riguardanti il tuareg, sempre di Volkswagen.

Se un certo cliente fosse invece interessato a prenotare una prova su strada di una certa autovettura, dopo aver ricevuto sul display la lista delle auto può scegliere quale tra questa desidera provare attraverso il proprio dispositivo tramite una apposita applicazione, in modo da fissare un appuntamento.

Il servizio di agenda, nel quale si tiene conto di tutte le prenotazioni pendenti e che quindi risulta necessaria da consultare è modellato da un apposito artefatto di risorsa che possiamo chiamare semplicemente '*timetable*' ovvero calendario.

Della coordinazione delle varie richieste da parte degli agenti software che rappresentano gli utenti della concessionaria si fa carico un tuple centre *scheduler*, il quale altro non è che un esempio di artefatto di coordinazione che fornisce servizi di scheduling; come si è visto nel capitolo 4 riguardo i tipi di artefatti di coordinazione, in questo scenario si ha il supporto di un artefatto di coordinazione per la gestione dei flussi di lavoro.

Per quanto riguarda la richiesta di prenotazione di una prova su strada, l'agente che risiede nel dispositivo dell'utente andrà a scrivere direttamente nel tuple centre di scheduling per far sì che avvenga la procedura di prenotazione:

```
scheduler_tc @ 'sched_tc_address' ?  
out(bookingRequestTask(targetCar, trialDays, userID,  
drivingLicenseNumber))
```

questo comporta una reazione nel centro di tuple:

```
reaction( out(bookingRequestTask(targetCar, trialDay, info(userID,  
drivingLicenseNumber))), from_agent,  
(  
    out_r(taskToDo(bookingRequestTask, info(targetCar,  
    trialDay, userID, drivingLicenseNumber))),  
    in_r(bookingRequestTask(targetCar, trialDay,  
    info(userID, drivingLicenseNumber)))  
)
```

Nello scenario coordinativo sarà anche presente un agente dedicato ad incaricarsi delle operazioni di prenotazione, è possibile chiamarlo *bookingAgent*: questo rimane in attesa di un compito da svolgere tramite la primitiva:

```
inp(taskToDo(taskType, info))
```

perciò, quando è pronta una nuova attività –in questo caso di prenotazione– il bookingAgent se ne fa carico:

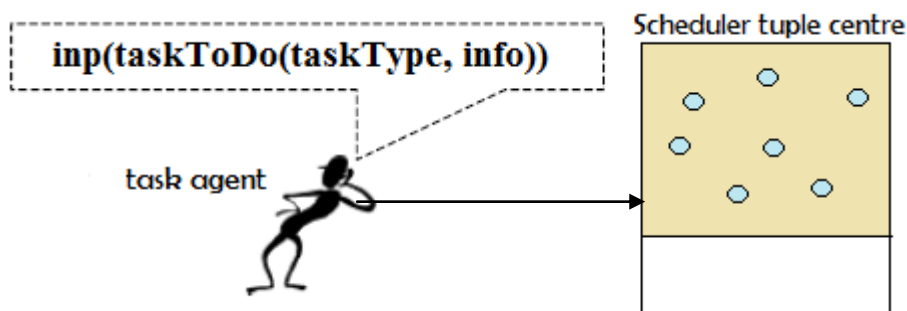


Figura C.4: agente attende che vi sia la necessità di eseguire una certa attività

```
reaction (inp(taskToDo(taskType, info)),
  (
    pre,
    current_agent(ExecutorID),
    out_r(task_started(taskType, Info, ExecutorID))
  )

reaction (out_r(task_started(taskType, info, ExecutorID)),
  from_agent,
  (
    timetable ? get (availableDays, daysList),
    timetable_tc @ timetable ?
    out_r (research (daysList, trialDay),
    in_r (task_started(taskType, info, ExecutorID))
  )
)
```

Dove *trialDay* è il giorno di prova desiderato dall'utente che è stato estrapolato dal campo *info* di *user*, inoltre la tupla che specificherà l'operazione di ricerca, farà riferimento al tuple centre presente al nodo dell'artefatto di risorsa, che memorizza tutte le date disponibili per le prenotazioni.

```
reaction (out_r (research (daysList, trialDay)),
  (
    timetable_tc @ timetable ?
    in (rd-all (matching(trialDay, daysList)
  )

reaction (rd-all (daysList matching trialDay), success,
  (
    timetable ? set (availableDays, daysList),
    out_r (task_result ("booking successful")
  )

reaction (rd-all (daysList matching trialDay), failure,
  (
    out_r (task_result ("days not available"))
```

)

Viene specificato il comportamento del centro di tuple dello scheduler nel caso in cui il ciclo per verificare se la data desiderata fa match con la lista di date a disposizione dei clienti, in un caso e nell'altro il risultato dell'attività sarà modellato come una stringa di comunicazione all'utente di avvenuta prenotazione –ciò comporterà anche l'eliminazione della data dal centro di tuple interno all'artefatto di calendario– oppure sul display verrà visualizzato un messaggio che comunica l'indisponibilità di tali date.

```

reaction (out_r (task_result(string)),
  (
    write_display(string),
    in_r (task_result(string))
  )
)

```

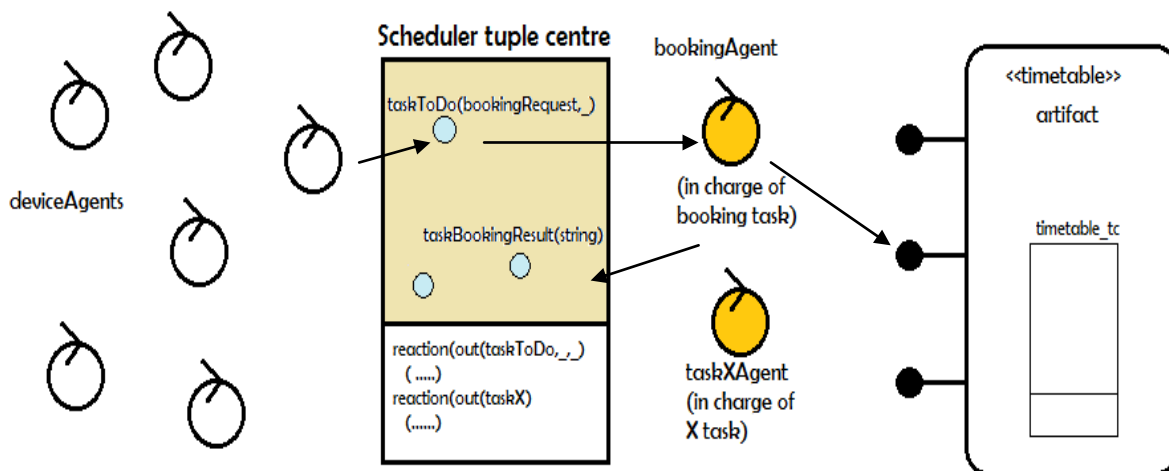


Figura C.5: interazione tra agenti per l'esecuzione di certe attività gestita tramite un artefatto coordinativo

Ipotizzando che l'utente che possiede le preferenze viste sopra
'Car' (hasMaxSpeed:<160; hasMaker in {renault, volkswagen}; hasPrice:<18000)

voglia prenotare una prova su strada il giorno Giovedì 22 Luglio: una volta entrato nella sala, gli viene presentato sul suo display l'auto che risulta conforme alle sue preferenze, azionando un apposito comando dal proprio PDA, l'agente software che risiede su questo provvederà ad immettere la tupla rappresentante la richiesta di esecuzione del *booking task* nel tuple centre che funge da artefatto di coordinazione:

out(bookingRequestTask(polo, 22/07/2010, info(284797, FO5161255T)))

questa primitiva comporta una serie di reazioni ovvero l'immissione da parte dell'agente di una tupla di richiesta di attività generica, nel cui figura il campo "tipo" di attività, e nella quale è appunto specificato che è una attività di richiesta di prenotazione:

out_r(taskToDo(bookingRequestTask, info(polo, 22/07/2010, 284797, FO5161255T))),
in_r(bookingRequestTask(polo, 22/07/2010, info(284797, FO5161255T)))

essendoci agenti dedicati per ogni attività che sono in attesa di prendersene carico, lo scheduler quindi reagisce alla tupla memorizzata, taskToDo, facendo prendere carico di tale attività l'agente adatto,

current_agent(ExecutorID),
out_r(task_started(bookingRequestTask, polo, 22/07/2010, 284797, FO5161255T, 1506))

il quale avvia una procedura di ricerca tra le date disponibili: se è presente quella a cui è interessato tramite l'artefatto rappresentante il calendario

timetable ? get (availableDays, daysList),
timetable_tc @ timetable ? out_r (research (daysList, 22/07/2010)

l'attività di ricerca verifica se la tupla rappresentante la data desiderata fa match con una di quelle presenti nel centro di tuple che mantiene le informazioni di calendario

Prima della richiesta di prenotazione:

availableDays	→	daysList
{5/07/2010,6/07/2010.....21/7/2010,22/7/2010,1/8/2010,...}		

successivamente al successo dell'operazione di ricerca:

reaction (rd-all (daysList matching trialDay), success,

```
(
    timetable      ?      set      (availableDays,
    {5/07/2010,6/07/2010.....,21/7/2010,1/8/2010...}),
    out_r (task_result (“booking successful”)
)
```

Conclusioni

Partendo dal concetto di sistema pervasivo e dall'idea di un ambiente da esso supportato, lungo il corso dell'elaborato è stato possibile dare una struttura ben definita al sistema che sta alla base di uno Smart Environment tramite le recenti tecnologie software rispettando quelle che sono le principali caratteristiche dei sistemi distribuiti moderni e concentrando l'attenzione sull'importanza degli artefatti di coordinazione in questi tipi di scenari.

Basandosi dunque sui sistemi multi agente è stata creata quella che è una infrastruttura che ricalca nella maniera più naturale possibile il comportamento autonomo e proattivo di un ambiente intelligente, il risultato di tale progettazione è stato poi schematizzato attraverso una raffigurazione stratificata che nasce dalla integrazione delle infrastrutture e tecnologie sopra citate, in maniera tale da poter essere il più coerente possibile, oltre che flessibile per poter garantire il suo impiego in un insieme di scenari abbastanza ampio.

Con la presentazione finale del caso di studio poi, si è voluto dare al lettore una semplice rappresentazione di come è concepibile, da un punto di vista di programmazione, strutturare il sistema.

È evidente anche osservando il case study come sia stata rispettata la struttura di progettazione: le varie sale della concessionaria sono state trattate come nodi TuCSoN ognuno dei quali contenente il proprio tuple centre ReSpecT con le informazioni caratteristiche del nodo ospitante, tali nodi sono visitabili poi da un insieme di agenti software, i quali scrivono sul nodo centrale dell'organizzazione TuCSoN per comunicare le proprie preferenze.

Esistono poi altri tipi di agenti caratteristici dello Smart Environment che forniscono determinati servizi e la cui attività viene coordinata da un determinato artefatto rappresentato da un centro di tuple per lo scheduling; le funzioni di supporto agli utenti dipendono in primo luogo dalla loro posizione all'interno dello show room e quindi implica una interazione anche con lo strato fisico dell'ambiente, rappresentato da una rete di sensori come lettori RFID –comunicante via onde radio– i quali, per così

dire, “wrappano” il centro di tuple caratteristico di ogni sala, andando a scrivere su di esso gli utenti effettivamente presenti.

Dopo aver discusso su come risulta opportuno strutturare il sistema risulta utile vedere alcuni scenari futuri nel quale tali ambienti potrebbero venire applicati menzionando brevemente quelle che sono le conseguenze che essi implicano nella vita di tutti i giorni per gli utenti che abitano un ambiente intelligente.

Analizzando tali scenari su vasta portata è possibile presentare tre implicazioni rilevanti, (i) il profondo cambiamento dei paradigmi economici che deriva dallo sviluppo su larga scala delle tecnologie del calcolo pervasivo, (ii) le questioni legate alla privacy che possono essere associate ad esso e (iii) la generale compatibilità ed affidabilità delle applicazioni future che si baseranno su tali tecnologie.

Sviluppi futuri

Tra le principali implicazioni di cui si deve tener conto nel vivere in un ambiente intelligente vi è quella che riguarda il lato economico; si possono menzionare alcuni scenari di esempio che possono essere il *dynamic pricing* (prezzi dinamici): dove, se la data di scadenza di un certo prodotto si avvicina il suo prezzo cala per incoraggiare l’acquisto da parte del consumatore oppure se la temperatura esterna sale, i soft drink o i gelati possono aumentare il loro prezzo in accordo con l’atteso incremento della domanda, oppure i premi delle assicurazioni real time che si basano su continue informazioni inviate dai dispositivi presenti nell’automobile riguardanti il guidatore di un certo veicolo, in modo da poterne valutare la classe di rischio, o ancora il cosiddetto *pay-per-use* che può essere generalizzato praticamente per ogni transazione: in questa maniera però, se i consumatori accettano questo schema su vasta scala, il monitoraggio continuo delle loro abitudini personali minaccia la loro privacy.

Naturalmente gli scenari in cui gli Smart Environment sarebbero applicabili sono un numero molto vasto e ricoprono estese aree della vita quotidiana della società, dai servizi pubblici, a quelli privati, dalla sanità alla

sicurezza, al risparmio energetico ed al risparmio di tempo per le varie attività.

Si possono in più considerare le Smart Classroom –che supportano ed ottimizzano lo svolgimento delle lezioni sia da parte degli studenti che da quella degli insegnanti–, gli Smart Office –dove le decisioni che solitamente richiedono il contributo di più persone vengono delegate all’ambiente, il quale riduce il ciclo di vita della decisione– o, per scenari più ampi, ambienti intelligenti su scala cittadina

È significativo pensare ai notevoli vantaggi che porterebbero ospedali intelligenti ai pazienti ed a chi ci lavora, ai dipartimenti universitari, ai comuni, alle banche, alle aree pubbliche cittadine, alle normali abitazioni; insomma praticamente in qualsiasi luogo il quale richiederebbe l’organizzazione di una serie di attività le quali, se propriamente gestite in base al contesto, permetterebbero un’ottimizzazione globale dei servizi offerti.

Vivere in uno Smart Environment

Le varie aziende che già impiegano tecnologie *ubiquitous* si muovono verso la formazione di una società di informazioni onnipresenti nel quale la disponibilità di informazioni è ovunque, in qualsiasi momento e di qualsiasi tipo, può non solo rendere il business del giorno d’oggi più affidabile, efficiente ed *user-friendly*, ma può anche stimolare lo sviluppo di nuovi modelli di commercio.

In più, basandosi sulla definizione di quelli che sono gli Smart Environment come ambienti basati su sistemi pervasivi/onnipresenti, tali sistemi hanno la potenzialità di creare reti di sorveglianza invisibili che possono coprire un’area senza precedenti della nostra vita sia pubblica che privata; queste reti diventerebbero una realtà che potrebbe turbare gli abitanti di tali ambienti.

D’altro canto, visto il ritorno economico immediato nell’utilizzo di programmi di fedeltà del consumatore o di bassi tassi di assicurazione, la basse minacce alla privacy che si potrebbero avere in futuro possono essere facilmente ignorabili.

Oggigiorno la vita senza computer –intesi come unità di calcolo e non strettamente parlando di personal computer– è inimmaginabile: i potenziali benefici economici che sono portati dalla ubiquitous computing sono di certo uno dei fattori chiave della proliferazione dell'information technology, ma tuttavia possiamo ancora decidere in maniera autonoma se partecipare e vivere in questi Smart Environment.

Tuttavia, in un futuro largamente computerizzato, non sarà più possibile evitare la dipendenza indotta da questi tipi di tecnologie.

E' importante sottolineare che l'arricchimento degli oggetti di tutti i giorni con capacità di rilevamento, di comunicazione e di calcolo non hanno a priori implicazioni negative; questi oggetti possono aiutare a creare nuovi servizi ed applicazioni utili per il beneficio degli individui e della collettività.

Esistono ovviamente un gran numero di domande riguardo alle conseguenze che caratterizzano gli Smart Environment (Le persone si potrebbero sentire sotto osservazione dai numerosi smart objects e dai sensori wireless che li circondano, oppure si dimenticheranno di questi vivendo senza accorgersene? Questi renderanno il mondo un ambiente più sicuro ed confortevole, oppure alcune persone potrebbero approfittare dei dati rilevati da tali sensori per i propri scopi? Se alcuni artefatti diventano sempre più autonomi e rendono non necessario l'intervento ed il controllo dell'uomo: chi sarà il responsabile se qualcosa dovesse andare storto?), ma speculando sulle conseguenze di queste tecnologie e valutandole nei campi della sociologia o della economia potremmo essere in grado di fornire le necessarie informazioni per poter indirizzare nella maniera appropriata lo sviluppo di sistemi pervasivi in maniera compatibile con la società.

Sarà dunque un'attenta analisi degli scenari ben conosciuti, un approccio promettente per poter aiutare gli ingegneri nello sviluppo di Smart Environment e del sottostante sistema pervasivo.

Riferimenti

- [1] Hyacinth S. Nwana. Software Agents: an overview. *Knowledge Engineering Review*, Vol. **11**, No 3, pp. 205-244, October/November 1996.
- [2] M. Casadei and A. Omicini. Situated Tuple Centre in ReSpecT. 24th *ACM Symposium on Applied Computing, Special Track on Coordination Models and Languages*, Honolulu, HI, USA 10 March 2009
- [3] A. Omicini and F. Zambonelli. Tuple Centres for the Coordination of Internet Agents. *Proc. of the 1999 ACM Symposium on Applied Computing (SAC'00)*, February 1999.
- [4] TuCSoN guide. <http://apice.unibo.it/xwiki/bin/view/TuCSoN/>
- [5] da: <http://www.ubiq.com/hypertext/weiser/UbiHome.html>
- [6] F. Zambonelli and M. Mamei. Pervasive computing and context-aware technologies.
- [7] B. Schilit, N. Adams and R. Want Context-aware computing applications. *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp 85–90, December 1994
- [8] A.K. Dey. Understanding and using context. *Personal Ubiquitous Computing* 5(1):4–7, <http://dx.doi.org/10.1007/s007790170019> 2001
- [9] Hull, R., Neaves, P. and Bedford-Roberts, J. Towards situated computing. *Proceedings of the First International Symposium on Wearable Computers (ISWC '97)*, p.146. March 1997
- [10] A.K Dey, D. Salber, M. Futakawa and G.D Abowd. An Architecture to Support Context-Aware Computing. *Submitted to UIST '99*

- [11] A.K Dey and G.D Abowd. Towards a better understanding of context and context-awareness. *Workshop on the What, Who, Where, When, and How of Context-Awareness (2000)*.
- [12] G. M. Youngblood, D. J. Cook, L. B. Holder and E. O. Heierman. Automation intelligence for the smart environment. *Proceedings of the International Joint Conference on Artificial Intelligence, 2005*.
- [13] JC Augusto. Ambient Intelligence: the confluence of Ubiquitous/Pervasive Computing and artificial intelligence. *Intelligence computing everywhere Springer London, pp 213-234*.
- [14] da Wikipedia: http://en.wikipedia.org/wiki/Smart_device
- [15] J.C. Augusto, H. Nakashima and H. Aghajan. Ambient Intelligence and Smart Environments: A state of art.
- [16] D.J Cook and S.K. Das. How Smart are our Environments? An updated look at the state of art.
- [17] M. Wooldridge and N.R Jannings. Intelligent Agents: theory and practice. *The knowledge engineering review 10(2), pp 115-152, 1995*.
- [18] S.J Russell and P. Norving. Artificial Intelligence: a modern approach. 2002
- [19] HS. Nwana. Software Agents: an overview. *The Knowledge Engineering Review, 2009*
- [20] J.Odell. Objects and agents compared. *Journal of object technology Vol.1 No.1 May-June 2002*
- [21] A. Ricci, A. Omicini and E. Denti. Engineering Agent Societies: a case study in Smart Environments. *AAMAS pp 15-19 Bologna. July 2002*.
- [22] PD. O'Brien and RC Nicol. FIPA, towards a standard for software agents. *BT Technology Journal, Springer. 1998*
- [23] Carlucci, Aiello, Nardi. *Lezioni di Intelligenza Artificiale 2005/2006*
- [24] D. Gelernter, and N. Carriero. Coordination languages and their significance. *Communications of the ACM, 35(2):97 1992*

- [25] A. Omicini and E. Denti. From Tuple Spaces to Tuple Centres. *Science of Computer Programming* (41) 277–29. 2001
- [26] ReSpecT guide. <http://apice.unibo.it/xwiki/bin/view/ReSpecT/>
- [27] A. Omicini, A. Ricci and M. Viroli. Artifacts in the A&A meta model for multi-agent system.
- [28] Viroli, M., Omicini, A., & Ricci, A. (2005). Engineering MAS environment with artifacts. In D. *Second international workshop environments for multi-agent systems (E4MAS 2005)*. AAMAS 2005, Utrecht, Netherlands.
- [29] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. New York (NY). 2004
- [30] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. *Workshop on Programming Languages for Multi-Agent Systems (PROMAS), held at AAMAS'05, Utrecht, Netherlands, 2005*.
- [31] A. Omicini. Towards a notion of agent coordination context. *Process Co-ordination and Ubiquitous Computing*, p. 187-200, 2002.
- [32] M. Mamei, F. Zambonelli and L. Leonardi. Tuples On The Air: A Middleware for Context-Aware Computing in Dynamic Networks. *ICDCS Workshops 2003*
- [33] M. Mamei, F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. *IEEE International Conference On Pervasive Computing (Percom) 2004* Orlando, FL. IEEE Computer Society Press.

Ringraziamenti

In primo luogo ringrazio la mia famiglia, che mi è sempre stata vicina e mi ha sempre sostenuto nel percorso che mi ha portato fino a qui, credendo sempre nelle mie capacità.

Ringrazio il prof. Andrea Omicini che mi ha dato la possibilità di svolgere questo elaborato e mi ha seguito nella sua evoluzione, e soprattutto la dottoressa Elena Nardini che mi ha supportato passo dopo passo nello sviluppo della tesi sopportando le mie domande, i miei continui dubbi e le mail interminabili.

Infine tutti i miei amici, dai miei tre inseparabili compagni di progetti notturni: Foia, Marco e il Franz, ad Anna che mi è sempre stata vicina, agli amici del bar e quelli di sempre che mi hanno aiutato a distrarmi, soprattutto durante il fine settimana facendomi sentir meno il peso degli esami e delle ore di studio, ai quali devo tantissimo.

