

Anno accademico 2009/2010

**UNIVERSITA' DI BOLOGNA SEDE DI CESENA**



**CORSO DI LAUREA:ING. INFORMATICA LS**

**PROGETTO DI SISTEMI MULTI-AGENTE**

**Prof. ANDREA OMICINI**

**OGGETTO: MODELLO BDI (Belief,  
Desire, Intention)**

Autore: Ivan Campolieti

Matricola: 280478

# Indice

- 0. Introduzione**
- 1. Modello BDI**
  - 1.1 Alberi decisionali**
  - 1.2 Architettura e Interprete BDI**
  - 1.3 Esempio AIRCRAFT**
  - 1.4 Pregi e difetti**
- 2. SOAR**
- 3. IRMA**
- 4. PRS**
- 5. Conclusioni**
- 6. Bibliografia**

## 0. Introduzione

Lo sviluppo di sistemi che richiedono gestione e controllo di compiti in contesti complessi e dinamici acquisisce continuamente un'importanza commerciale rilevante. Si pensi ad esempio a sistemi concepiti per la gestione e il controllo del traffico aereo oppure alle reti di telecomunicazioni, ai veicoli spaziali ecc. I sistemi computazionali tradizionali hanno limiti pratici quali l'accesso all'informazione e limiti computazionali legati alle caratteristiche delle risorse impiegate. [4] I sistemi software convenzionali sono stati sviluppati prevalentemente per mondi statici in cui il contenuto informativo è certo e completo. Mentre i sistemi moderni devono fronteggiare con risorse limitate un contesto variabile secondo modalità non prevedibili e per il quale si dispone di conoscenza locale, dunque parziale. Questi vincoli influenzano lo sviluppo di architetture computazionali idonee ai domini applicativi attuali. Dall'impiego di tecniche software convenzionali per lo sviluppo di sistemi come quelli sopra riportati si evince la grande difficoltà nell'implementazione, nel controllo e nella manutenibilità degli stessi.

I sistemi Agent-Oriented offrono approcci radicalmente diversi e innovativi rispetto alle tecniche convenzionali offrendosi come soluzioni alle difficoltà emerse dall'impiego di queste ultime. Uno tra gli approcci proposti immagina che il sistema sia un agente razionale dotato di tre atteggiamenti mentali: Belief, Desire, Intention (BDI) corrispondenti rispettivamente agli stati informativi, motivazionali, intenzionali dell'agente.

## 1. Modello BDI

Il modello BDI (Belief Desire Intention) è stato proposto da Michael Bratman, professore di filosofia presso la Stanford University. BDI affonda le radici nella tradizione filosofica con l'obiettivo di comprendere il ragionamento pratico ( practical reasoning ) umano.

[6] *Ragionamento pratico vuol dire valutare considerazioni contrastanti, a favore o contro opzioni alternative, ove le considerazioni sono date da ciò che l'agente desidera/valuta/preferisce e cosa crede.*

E' importante distinguere il ragionamento teorico da quello pratico. Il primo è diretto verso le credenze (Beliefs). Ad esempio: se io credo che tutti gli uomini siano mortali e io credo che Socrate sia un uomo, allora concluderò che Socrate è mortale. Il processo che ha condotto alla conclusione è un ragionamento teorico poiché coinvolge esclusivamente le mie credenze circa il mondo. Il processo di prendere un autobus piuttosto che un treno è invece un ragionamento pratico ossia diretto verso azioni.

Il ragionamento pratico umano consiste almeno di due attività distinte: la prima riguarda la decisione di *quale* stato di fatto si vuole raggiungere e viene detta *Deliberation*; la seconda riguarda la scelta di *come* conseguire gli obiettivi e si chiama *means-end reasoning*. Il ragionamento pratico

sembra essere un processo semplice ma esistono numerose complicazioni. Le due attività descritte sono processi computazionali che, in tutti gli agenti reali, sono eseguiti avendo a disposizione risorse limitate come memoria, velocità del processore ecc. Tali limiti vincolano la dimensione delle computazioni che possono essere eseguite in un certo lasso di tempo. La limitatezza delle risorse computazionali ha due implicazioni importanti:

- La computazione è una risorsa costosa per agenti collocati in ambienti real-time. La capacità di conseguire prestazioni elevate è legata all'abilità nell'impiegare nel modo più efficiente le risorse computazionali. In altre parole un agente deve controllare la propria attività di ragionamento se vuole avere un comportamento ottimale;
- Gli agenti non possono deliberare indefinitamente, ad un certo punto devono fermarsi e, dopo aver scelto uno stato di fatto, procedere per raggiungere lo stato di fatto desiderato. Può accadere che lo stato prescelto non sia ottimale, in tal caso si deve adottare un altro stato da conseguire;

BDI è un modello che si ispira ai processi mentali che coinvolgono il ragionamento umano e individua in Belief, Desire, Intention le tre componenti dello stato mentale. Più specificamente:

- Belief: insieme delle informazioni riguardanti l'ambiente, il contesto. Esso può contenere regole di inferenza utili per la deduzione di nuove informazioni. Può essere parziale, incompleto ed errato;
- Desire: rappresenta lo stato motivazionale dell'agente ossia l'obiettivo che vuole conseguire;
- Intention: rappresenta lo stato deliberativo dell'agente ossia i Desires per i quali ha pianificato delle azioni da eseguire.

[4] Beliefs sono essenziali poiché il mondo è dinamico e il sistema ha solo una vista parziale del mondo. Tuttavia, data la limitatezza delle risorse del sistema, è importante memorizzare alcune informazioni piuttosto che ricalcolarle a partire dai dati a disposizione.

Desires, più comunemente pensati come Goals, costituiscono un altro elemento fondamentale dello stato del sistema. In termini computazionali un Goal può essere il valore di una variabile, un record, un'espressione simbolica. L'aspetto importante è che esso rappresenta uno stato finale desiderato. La computer-science convenzionale è Task-oriented ossia ogni task viene eseguito senza che si ricordi il perché della sua esecuzione. Ciò implica l'incapacità per tali sistemi di poter recuperare un fallimento.

Definiti i componenti Belief e Desire è completo lo stato del sistema? Supponiamo di aver deciso un piano di azioni e che cambi il contesto, cosa si dovrebbe fare? Procedere o ripianificare? La teoria classica sulle decisioni consiglierebbe di ripianificare il corso di azioni sulla base della nuova configurazione del mondo; la teoria dei software convenzionali opterebbe, essendo task-oriented, per proseguire lungo il piano di azioni preventivato. Ma quale approccio è giusto? Nessuno dei due. Il sistema dovrebbe completare i piani specificati e i sotto-goals ma dovrebbe essere in grado di riconsiderarli in momenti appropriati.

I piani completati sono detti Intentions e rappresentano il terzo componente dello stato di un sistema progettato secondo il modello BDI. Per Intentions si intende un insieme di threads eseguiti in un processo che può essere interrotto appropriatamente a seguito della ricezione di un cambiamento del contesto. Dal punto di vista semantico i piani di azione potrebbero essere visti come un tipo particolare di Belief, ma a causa dell'importanza computazionale sono distinti dal componente Belief e considerati a parte.

Il modello BDI riconosce l'importanza delle intenzioni come elemento fondamentale nel guidare il ragionamento pratico. La caratteristica più importante delle intenzioni è la pro-attività ossia la tendenza a compiere azioni. Tuttavia avere un'intenzione non porta necessariamente alla produzione di azioni. Una seconda caratteristica non trascurabile è la persistenza delle intenzioni. Se fallisce un tentativo di conseguire un obiettivo precisato in un'intenzione si esegue un nuovo tentativo, ma se le ragioni di un'intenzione perdono valore, allora l'intenzione può essere abbandonata. Una terza proprietà è data dai vincoli che l'aver adottato un'intenzione pone sul ragionamento pratico futuro. Questa caratteristica è desiderabile dal punto di vista dell'implementazione di agenti razionali poiché consente l'impiego di un filtro di ammissibilità sulle intenzioni future di un agente.

## 1.1 Alberi decisionali

[9] Una grande classe di domini applicativi real-time gode di una serie di caratteristiche rilevanti ai fini dello sviluppo di un sistema in grado di gestire e controllare funzioni e scopi per i quali è stato concepito.

1. In ogni istante temporale l'ambiente ha diverse modalità di evoluzione essendo non deterministico;
2. In ogni istante di tempo esistono potenzialmente molte azioni o procedure che il sistema può compiere. Anche il sistema è non deterministico;
3. In ogni istante di tempo si possono avere più obiettivi differenti che il sistema deve poter conseguire;
4. Le azioni che consentono il raggiungimento degli obiettivi dipendono dallo stato del contesto e sono indipendenti dallo stato interno del sistema;
5. Il contesto può essere percepito solamente a livello locale;
6. La velocità alla quale azioni e computazioni possono essere eseguite deve rispettare i limiti di velocità del sistema.

Considerate le caratteristiche 1 e 2, è possibile pensare di formulare per il sistema in questione, un modello di comportamento con struttura ad albero in cui:

- ogni ramo rappresenta un possibile percorso di esecuzione;
- ogni nodo rappresenta un certo stato di fatto;
- ogni transizione rappresenta una azione primitiva compiuta dal sistema, oppure un evento primitivo verificatosi nel contesto, oppure entrambi.

Differenziando le azioni eseguite dal sistema dagli eventi avvenuti nel contesto, è possibile immaginare che il non determinismo che governa entrambi i fenomeni si manifesti in due diversi tipi di nodo:

- Nodo scelta (decisione) che rappresenta opzioni disponibili al sistema;
- Nodo possibilità che rappresenta l'incertezza del contesto.

[9] Nel modello formale presentato si possono individuare gli obiettivi del sistema in particolari percorsi lungo la struttura ad albero ognuno etichettato con lo specifico obiettivo che mira a conseguire, unitamente a costi e vantaggi derivanti dalla scelta del percorso stesso. Il sistema necessita di azioni e procedure appropriate per poter agire ed eseguire quando ha di fronte un ventaglio di opzioni possibili. A tal fine si rende necessaria una funzione di selezione che abilita il sistema a raggiungere gli obiettivi prefissati a partire dalle risorse computazionali disponibili e dalle caratteristiche del contesto in cui il sistema è situato. In particolare sono due le tipologie di dati necessari:

1. lo stato del contesto (caratteristica 1) che essendo percepibile solo localmente (caratteristica 5) richiede un continuo aggiornamento dell'informazione in seguito ad ogni azione locale. Tale informazione può essere richiesta al componente chiamato Beliefs. Esso può essere immaginato come una variabile, un database, un insieme di espressioni logiche o qualsiasi altra struttura dati;
2. informazioni riguardanti gli obiettivi da conseguire o meglio le priorità e i costi associati agli stessi (caratteristiche 3, 4). Si può definire tale componente Desires come rappresentante dello stato motivazionale del sistema.

A partire da tale quadro si può realizzare la funzione di selezione seguendo gli approcci derivanti dalla teoria decisionale se non fosse per la caratteristica 6. Infatti l'ambiente evolve sia durante la selezione sia durante l'esecuzione dell'azione selezionata in diversi modi, non prevedibili. La prima problematica può essere superata mediante impiego di funzioni di selezione più rapide così da ridurre il rischio che si verifichino eventi durante la computazione della funzione di selezione. La seconda problematica è stata affrontata dalla teoria classica decisionale e dalla computer-science seguendo due logiche di pensiero: la prima propone la riapplicazione della funzione di selezione nel nuovo contesto; la seconda attende il risultato della computazione senza alcun riadattamento al nuovo ambiente. Riconsiderare la scelta dell'azione da eseguire ad ogni step diventa troppo oneroso, attendere la conclusione dell'azione scelta può portare a un fallimento degli obiettivi prefissati dal sistema. Per raggiungere un valido compromesso si può pensare di limitare la frequenza di ricalcolo dell'azione da intraprendere introducendo un terzo componente: Intentions. Questo rappresenta il corso di azioni correntemente scelto, ossia il risultato della più recente chiamata alla funzione di selezione.

Un albero decisionale consiste di nodi decisionali, nodi possibilità, nodi terminali e include una funzione di probabilità che mappa i nodi possibili su valori reali di probabilità e una funzione di costo che mappa i nodi terminali su numeri reali rappresentanti lo sforzo in termini di tempo o di numero di operazioni o di altri parametri possibili. La funzione di deliberazione, come una funzione di massimo o di minimo, viene definita per scegliere un'azione o una sequenza di azioni per raggiungere un determinato nodo. Si trasforma dunque un albero decisionale unitamente ad una

funzione di deliberazione in un modello equivalente BDI. Si consideri un albero decisionale completo in cui tutti i possibili percorsi sono rappresentati. Partendo dal nodo radice si attraversano gli archi e per ogni stato derivante da un nodo possibilità si crea un nuovo albero decisionale identico a quello originale fatta eccezione per il fatto che: il nodo possibilità è eliminato e l'arco incidente sul nodo eliminato è direttamente connesso con il successore. Tale procedimento si itera finché vengono rimossi tutti i nodi possibilità. Così si ottengono diversi alberi decisionali ognuno dei quali contiene solo nodi decisione e nodi terminali corrispondenti a una particolare configurazione del contesto.

Ogni albero decisionale rappresenta un possibile mondo con differenti probabilità di occorrenza. Infine si attribuisce una funzione di costo a ogni percorso dalla radice ai nodi terminali. Questi mondi possibili, ottenuti con le trasformazioni riportate, contengono due tipi di informazioni rappresentate dalla probabilità dei mondi e dai costi di ogni percorso. Suddividendo queste informazioni rispettivamente in relazioni belief-accessibility e in desire-accessibility si consente a un agente l'uso della funzione di deliberazione scelta per decidere il migliore corso di azioni. Si può formalmente rappresentare l'insieme dei percorsi selezionati dall'albero decisionale usando una terza relazione sui mondi possibili, detta intention-accessibility, che corrisponde alle intenzioni dell'agente.

Riassumendo: per ogni mondo desire-accessible esiste un corrispondente mondo intention-accessible che contiene esclusivamente i migliori piani di azione determinati dall'appropriata funzione di deliberazione.

[5] L'assiomatizzazione cattura le relazioni che intercorrono tra le tre componenti del modello BDI evidenziando tre principali tipologie:

- *Weak Realism*: è la forma più leggera di vincoli che si possono imporre su beliefs, desires, intentions e prevede che ci sia consistenza tra i vincoli di ogni componente. Ad esempio: se un agente ha una intenzione  $p$  allora l'agente non desidera la negazione di  $p$  né crede nella negazione di  $p$ ; allo stesso modo se un agente desidera  $q$ , non crede nella negazione di  $q$ . Sostanzialmente si richiede che ci sia almeno un mondo comune a intention-accessible world e a belief-accessible world. Similmente si richiede almeno un mondo che sia intention-accessible e desire-accessible ed anche un mondo che sia desire-accessible e belief-accessible;
- *Strong Realism*: gli assiomi del weak-realism possono essere insieme di assiomi per ragionare similmente all'uomo ma risultano troppo deboli per sviluppare agenti con stati mentali che possono portare a termine i propri desideri in un mondo dinamico. In particolare, se un agente ha l'intenzione che in un certo momento nel futuro possa soddisfare una proposizione  $p$ , sarebbe desiderabile che l'agente credesse nella futura esistenza di un percorso. Ossia un agente che intende eseguire un determinato cammino di azioni dovrebbe anche credere che quel cammino sia conseguibile. Il weak-realism non garantisce questa inferenza, ma richiede solamente che l'agente non creda nell'impossibilità del conseguimento del percorso. Strong-realism si basa sui seguenti assiomi: tutti i desire-accessible world sono contenuti nell'insieme di intention-accessible world; tutti i belief-accessible world sono contenuti nell'insieme di desire-accessible world.

- *Realism*: Cohen e Levesque hanno introdotto gli assiomi tra beliefs e desires. Questi richiedono che: se un agente crede ad una proposizione  $p$  allora la desidera. Estendendo alla coppia desire-intentions si può dire che: se un agente desidera una proposizione allora intende realizzarla. In altre parole: tutti i mondi intention-accessible sono contenuti nei mondi desire-accessible e questi ultimi sono contenuti nei mondi belief-accessible.

[8] Un aspetto rilevante dell'architettura BDI è la nozione di commitment (impegno) per decisioni precedenti che rappresenta l'equilibrio tra reattività e orientamento al Goal di un sistema Agent-Oriented. Un commitment generalmente consiste di due parti: una è rappresentata dalla condizione che l'agente deve conservare (commitment condition); l'altra è la condizione sotto la quale l'agente deve rinunciare al commitment (termination condition). Dato che l'agente non può controllare direttamente Beliefs e Desires, non può adottare alcuna strategia per portare a termine le precedenti decisioni. Tuttavia è in grado di scegliere cosa fare mediante le Intentions e quindi si restringe la commitment condition alle intenzioni dell'agente.

Ne derivano differenti strategie di commitment (relazioni tra intenzioni correnti e future):

1. blind-commitment: un agente che ha scelto un'intenzione la manterrà finché non l'avrà conseguita, in altre parole l'agente nega ogni cambiamento a belief e desire che contrasta con le proprie intenzioni;
2. single-minded commitment: un agente continua a perseguire l'intenzione scelta finché crede di poterla ancora conseguire, detto in altro modo l'agente consente cambiamenti di belief e in accordo a questi eventualmente abbandona l'intenzione;
3. open-minded commitment: un agente è portato ad abbandonare un'intenzione non solo se non crede più di poterla realizzare ma anche se non è più vista come un goal. L'agente permette cambiamenti di belief e desire che forzeranno l'abbandono dell'intenzione.

Così come per i vincoli statici, non si può dire quale strategia sia la migliore, perché ogni strategia è valida per una tipologia di domini applicativi. Se si conoscono i cambiamenti del contesto e i comportamenti attesi del sistema allora è pensabile l'impiego di una formalizzazione per specificare, sviluppare e verificare agenti che, una volta immersi nell'ambiente, manterranno solo comportamenti desiderati.

## 1.2 Architettura e interprete BDI

L'architettura astratta comprende tre strutture dati dinamiche che rappresentano le tre componenti Belief, Desire ed Intention, ognuna dotata di una coda di eventi in input. E' possibile aggiornare e interrogare mediante operazioni le strutture dati specificate. Gli eventi che il sistema è in grado di riconoscere possono essere esterni o interni. Si assume che gli eventi siano atomici e riconoscibili dopo la loro manifestazione.

[9] Il ciclo principale dell'interprete è riportato di seguito:

```

1  initialize-state();
2  repeat
    a) options := option-generator(event-queue);
    b) selected-options := deliberate(options);
    c) update-intentions(selected-options);
    d) execute();
    e) get-new-external-events();
    f) drop-successful-attitudes();
    g) drop-impossible-attitudes();
3  end repeat

```

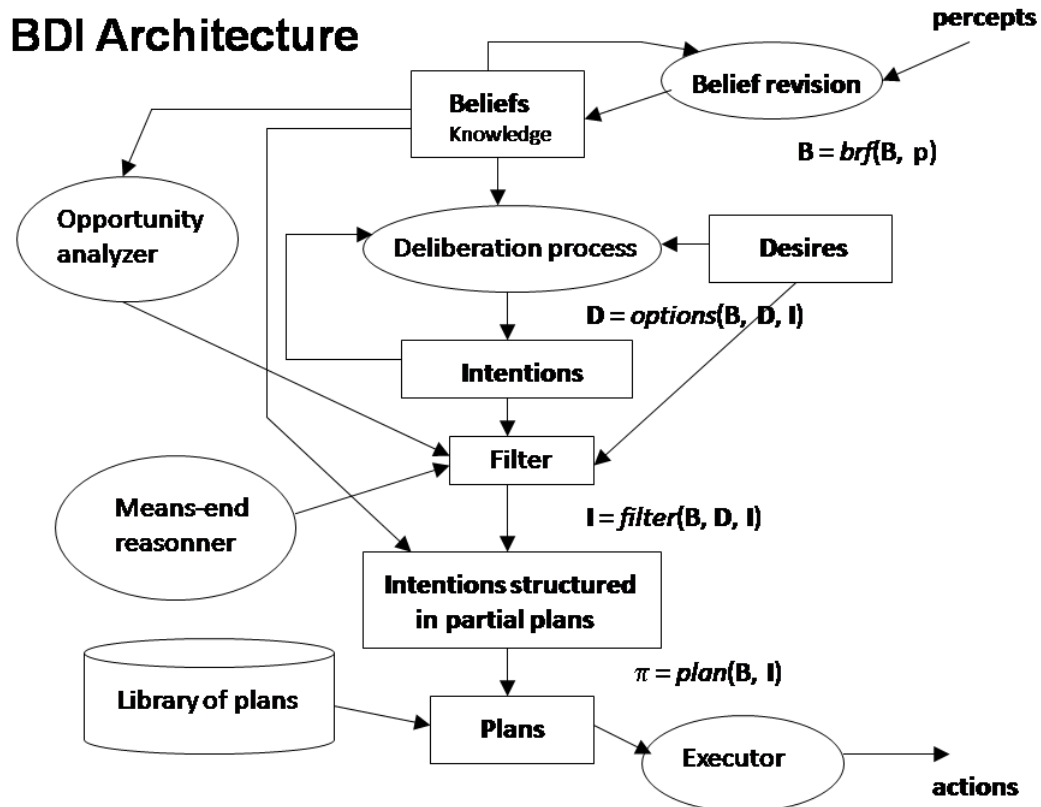


Fig. 1: Architettura BDI

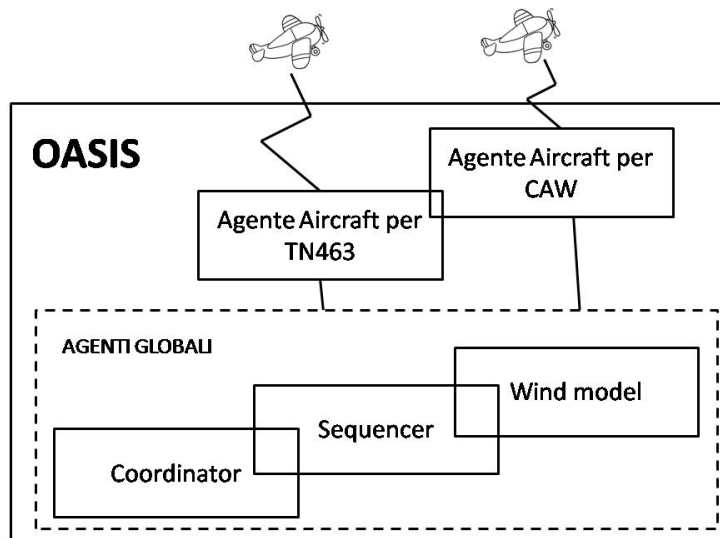
All'inizio di ogni ciclo il generatore di opzioni legge la coda degli eventi e restituisce una lista di opzioni, successivamente il deliberatore seleziona un sottoinsieme di opzioni da adottare e le aggiunge alla struttura della Intention. Se è presente un'intenzione che prevede azione atomica questa viene eseguita dall'agente. Ogni evento esterno che si verifica durante il ciclo dell'interprete viene inserita nella coda degli eventi. Gli eventi interni vengono aggiunti man mano che si verificano. Successivamente l'agente modifica la struttura di Desire e Intention cancellando desires ed intentions concluse con esito positivo e con esito negativo. Tale architettura è astratta e basata su insiemi di belief, desire, intention per cui le procedure richieste non sono provabili computazionalmente.

### **1.3 Esempio AIRCRAFT**

Il traffico aereo mondiale continua a crescere parallelamente allo sviluppo economico delle nazioni. I costi diretti e indiretti attribuibili alla congestione del traffico aereo sono enormi e talvolta insostenibili. Per risolvere il problema è stato sviluppato un sistema che sfrutta in modo più efficiente l'infrastruttura esistente per la gestione del traffico aereo e aiuta i controllori di volo nel pesante lavoro. Tale sistema si chiama [2] OASIS e gestisce il flusso di aircraft in arrivo in aeroporto garantendo enormi guadagni di capienza a costi relativamente ridotti. OASIS calcola accuratamente i tempi stimati di volo, determina la sequenza di aircraft in atterraggio producendo il minor ritardo totale e consiglia ai controllori di volo le azioni più appropriate per rispettare la sequenza calcolata, inoltre, controlla e confronta il progresso attuale degli aircraft con la sequenza stabilita e notifica ai controllori di volo le differenze significative e le azioni per rettificare la situazione corrente. Il sistema risponde adeguatamente ai cambiamenti ambientali (ad es. le condizioni meteorologiche) e ai cambiamenti negli obiettivi dell'utente (ad es. emergenze per un aircraft). OASIS lascia così più tempo a disposizione dei controllori di volo affinché possano valutare le azioni alternative di controllo. Infatti possono riservare manualmente postazioni nella sequenza e se necessario derivare istruzioni diverse da quelle consigliate da OASIS. In questo modo si migliora l'utilizzo e la gestione del traffico aereo evitando l'inflessibilità di sistemi totalmente automatizzati. Per sviluppare un sistema che possa svolgere le funzioni presentate sono necessari i seguenti requisiti: il sistema deve poter predire quando un aircraft può trovarsi in pista considerando le performance passate dello stesso, quelle future e i venti; deve sequenziare ogni aircraft secondo un ordine di atterraggio che ottimizzi l'impiego delle risorse pista. La sequenza deve essere prodotta in breve tempo e continuamente aggiornata per riflettere i cambiamenti correnti. Dopo aver assegnato agli aircraft i tempi di atterraggio calcolati si determinano i controlli e le istruzioni per rispettare i valori calcolati. La correttezza delle istruzioni dipende dalla posizione dell'aircraft. Il sistema deve poter offrire alternative multiple ai controllori di volo che sostituiscono, qualora ritenute migliori, le scelte di OASIS.

L'implementazione di OASIS prevede un agente Aircraft per ogni aircraft in arrivo e un numero di agenti globali che comprendono: serializzatore, vento, modellatore, coordinatore, traiettoria, verificatore. In ogni momento il sistema contiene fino a 70/80 agenti in esecuzione corretta, sequenzializza e controlla le direttive da consegnare ai controllori di volo in tempo reale. Gli agenti

Aircraft sono responsabili del volo mentre gli agenti globali sono responsabili della serializzazione e della coordinazione degli agenti Aircraft.



**Fig. 2: Architettura OASIS**

[2] **Modellazione:** un agente Aircraft è responsabile del volo lungo un percorso, dato dalle coordinate di una sequenza di punti. Un esempio di incertezza nel contesto è dato dal vento. Se ci fossero solo variabili ambientali per ogni valore della velocità del vento, ad un particolare punto avremmo la corrispondenza belief-accessible world. Le scelte a disposizione di un agente Aircraft includono il volo lungo differenti traiettorie percorse a velocità e ad altitudini comprese tra un valore minimo e massimo. Ciò è rappresentabile mediante rami multipli in ogni belief-accessible world. Dato che il punto finale rappresenta l'aeroporto di destinazione, i percorsi desiderati dall'agente Aircraft sono quelli per i quali i valori ETA ( tempo stimato di arrivo ), calcolati nel nodo finale, coincidono con i valori desiderati. I desire-accessible worlds si ottengono dai belief-accessible tagliando i percorsi che non soddisfano la condizione specificata. Gli Intention-accessible worlds si ricavano dai percorsi filtrati dai desire-accessible scegliendo solo i percorsi che meglio rispettano i consumi di carburante, le performance di volo, ecc.

**Teoria delle decisioni e commitment:** l'obiettivo primario dell'agente Sequenziatore è fare atterrare gli aircraft in sicurezza e secondo una sequenza ottimale. Date le caratteristiche degli aircraft, la distanza desiderata tra gli stessi, la velocità del vento, la pista assegnata e una funzione di costo, l'agente Sequenziatore impiega un numero diverso di strategie di deliberazione per computare la migliore sequenza di arrivo degli aircraft e i loro rispettivi ETA. Determinando uno scheduling specifico, l'agente Scheduling, single-mindedly termina le intenzioni.

**Interprete astratto:** ogni agente tratta beliefs e desires correnti, e le opzioni disponibili per raggiungere gli obiettivi specificati dai Desires sono riportate come piani di azione. Ad esempio: sebbene ci siano differenti modi per ottenere l'ETA desiderato, i piani dell'agente Aircraft includono come opzioni solamente le traiettorie che risultano più efficienti in termini di consumo di carburante.

## 1.4 Pregi e difetti BDI

I pregi del modello BDI:

- capacità di costruire piani di azione che reagiscano a specifiche situazioni e che possano essere invocati sulla base degli scopi stabiliti. I piani di azione sono sensibili al contesto in cui sono invocati e semplificano lo sviluppo modulare e incrementale;
- Equilibrio tra componente reattiva e goal-oriented del comportamento del sistema. E' acquisito tramite l'impegno per i piani selezionati e tramite il ricalcolo dei piani stessi in presenza di cambiamenti del contesto;
- Il linguaggio di programmazione e la rappresentazione ad alto livello consentono all'utente finale di codificare la conoscenza direttamente in termini di attitudini mentali senza dover ricorrere a costrutti di programmazione a basso livello.

[1] I limiti:

- Gli agenti BDI mancano di meccanismi per imparare dagli eventi passati che gli permettano di adattarsi alle nuove situazioni;
- Gli studiosi della teoria decisionale classica si chiedono se sia necessario disporre delle tre attitudini mentali mentre i ricercatori dell'Intelligenza Artificiale distribuita si chiedono se siano sufficienti le tre attitudini;
- La logica multi-modale proposta dal modello BDI ha poca rilevanza in pratica;
- Il modello non descrive esplicitamente i meccanismi di interazione tra gli agenti e la loro integrazione in sistemi multi-agente;
- La maggioranza delle implementazioni non offrono una rappresentazione esplicita dei Goals.

## 2. SOAR

Soar (State, Operator And Result) risolve i problemi in termini di ricerca in uno spazio delle soluzioni in cui applicare determinati operatori a stati per ottenere risultati.

[3] Offre un modello cognitivo e un'implementazione del modello per sviluppare sistemi AI. L'idea discende dalle teorie di Newell sulla scienza cognitiva. Soar è una delle architetture più ampiamente adottate dalla ricerca negli aspetti del comportamento umano allo sviluppo di giochi. L'obiettivo di Soar è la costruzione di sistemi che racchiudano intelligenza. Il comportamento è incapsulato in

regole simili alla forma if-then. La risoluzione di problemi consiste in una ricerca nello spazio di soluzioni. Se il sistema fallisce si impiegano altri metodi come ad esempio la ricerca locale. Quando si individua una soluzione, tramite un metodo detto chunking, si apprende una nuova regola che viene adottata nel momento in cui un agente incontra nuovamente il problema risolto.

Soar può essere considerato in tre differenti modi:

1. Una teoria della conoscenza;
2. Un insieme di proposizioni e vincoli sul processo cognitivo che propone un framework entro il quale costruire i modelli cognitivi e in questa soluzione può essere considerato come architettura per risolvere problemi basati sulla conoscenza, problemi di apprendimento e di interazione con l'ambiente esterno;
3. Un linguaggio di programmazione per l'Intelligenza Artificiale.

Il modello Soar è totalmente compatibile con le architetture BDI. Basti pensare che è basato su stati e operatori che sono simili ai piani di azione reattivi. Gli operatori possono essere descritti da precondizioni che aiutano nella selezione degli stessi per l'esecuzione basata sullo stato corrente dell'agente.

Il mapping tra architettura BDI e Soar segue facilmente:

- Intentions: gli operatori selezionati in Soar;
- Beliefs: inclusi nello stato corrente in Soar;
- Desires: i Goals;
- Strategie di commitment: le strategie per definire le condizioni di terminazione degli operatori.

## **Codificare un task**

Il minimo insieme di funzioni deve includere:

- Proposizione del task (nella forma di operatore);
- Terminare il task quando è stato completato (terminare l'operatore che rappresenta il task);
- Proporre uno spazio dei problemi per il task;
- Proporre uno stato iniziale;
- Proporre operatori per trasformare lo stato nello spazio dei problemi;
- Applicare un operatore per passare da uno stato ad un altro dello spazio dei problemi.

Informazioni aggiuntive per il controllo della ricerca nello spazio degli stati possono includere:

- Confronto della desiderabilità tra più tasks;
- Confronto degli spazi dei problemi per i tasks;
- Confronto degli operatori candidati per differenti stati del task;

- Implementazione e controllo di sotto-task che derivano dalla selezione.

Entrambe le architetture Soar e BDI sono state impiegate in diverse applicazioni su larga scala grazie alle caratteristiche di cui godono: efficienza, real-time, scalabilità.

### 3. IRMA

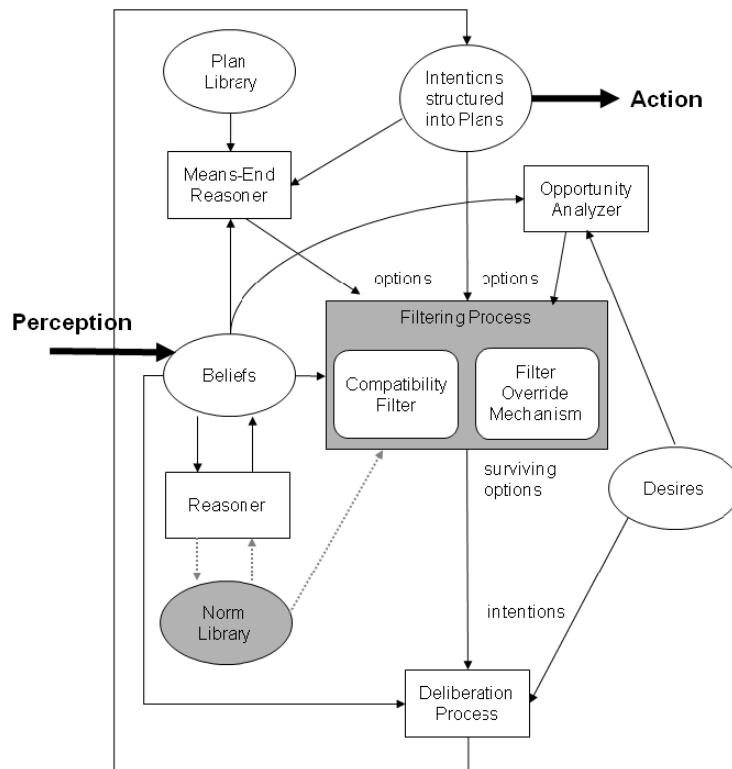
[7] L'architettura basata sul modello BDI riconosce che gli agenti saranno sempre limitati, sia computazionalmente, sia in termini di risorse. Tale vincolo impone un ragionamento mirato alla scelta ottimale delle decisioni.

IRMA consiste di quattro strutture dati:

1. Una libreria di piani;
2. Beliefs: informazioni disponibili all'agente;
3. Desires: ciò che l'agente vuole fare;
4. Intentions: desires che l'agente ha scelto e che si è prefissato di conseguire.

In aggiunta a tali strutture l'architettura IRMA consiste di:

- Un reasoner: motore di inferenza che esegue ragionamento pratico nel mondo;
- Un means-end analyzer: determina quali piani possono essere scelti per realizzare le intenzioni;
- Un opportunity-analyzer: monitora l'ambiente e in risposta ai cambiamenti genera nuove opzioni;
- Un filtering process: filtra le intenzioni e si compone di:
  1. compatibility filter;
  2. filter override mechanism.
- Un deliberator process: responsabile della scelta delle migliori intenzioni da adottare.



**Fig. 3: Architettura IRMA**

[8] L'architettura contiene una struttura di intenzioni in piani che possono essere attualmente adottati da un agente o piani memorizzati nella plan library. Le opzioni dei piani sono proposte come risultato del means-end reasoner o dall'opportunity analyzer, il quale rileva i cambiamenti del contesto e determina le nuove opportunità sulla base dei desires dell'agente. Le opzioni sono filtrate tramite due filtri che operano in parallelo: compatibility filter e filter override mechanism. Il primo filtro accerta la consistenza dei nuovi piani con le intenzioni esistenti dell'agente. Tipicamente piani incompatibili sono scartati, ma, in alcuni casi, se soddisfano determinate proprietà possono essere inviati al deliberation process. Il secondo filtro invia al deliberation process piani incompatibili che tipicamente sono scartati, in tal caso le intenzioni incompatibili verranno sospese.

Il processo di deliberazione determina la migliore opzione basandosi su Belief e Desire correnti. Si consideri un agente autonomo che vorrebbe raggiungere un certo Goal, l'agente dispone già di diversi piani di azione (nella propria plan-library) per conseguire il Goal. Prima di deliberare su un piano, l'agente attiva il processo di filtraggio che riduce i piani possibili a piani che possono essere completati a partire da quelli contenuti nella plan-library, dai Beliefs, dai Desires. L'agente sceglie da questo insieme di piani, quello che appare migliore e lo esegue.

L'architettura IRMA è stata dimostrata e testata in ambienti di simulazione Tileworld (agente robot simulato in un ambiente simulato dinamico e non prevedibile) ed è risultata una strategia utile per modellare i comportamenti degli agenti in contesti dinamici grazie soprattutto all'impiego dei filtri previsti dall'architettura.

## 4. PRS

[7] Fin dalla metà degli anni '80 molte architetture di controllo per il ragionamento pratico di agenti sono state proposte. Molte di queste sono state sviluppate solo in ambienti artificiali, poche sono state applicate ad ambienti reali, pochissime sono state implementate per applicazioni utili real-time. Unica eccezione è data da PRS (Procedural Reasoning System): Originariamente descritto nel 1987 è stato migliorato a partire da una versione LISP fino ad una completa implementazione C++ conosciuta come dMARS (distributed MultiAgent Reasoning System).

Come si può osservare dall'architettura le strutture coinvolte sono: beliefs, goals, intentions, plan library.

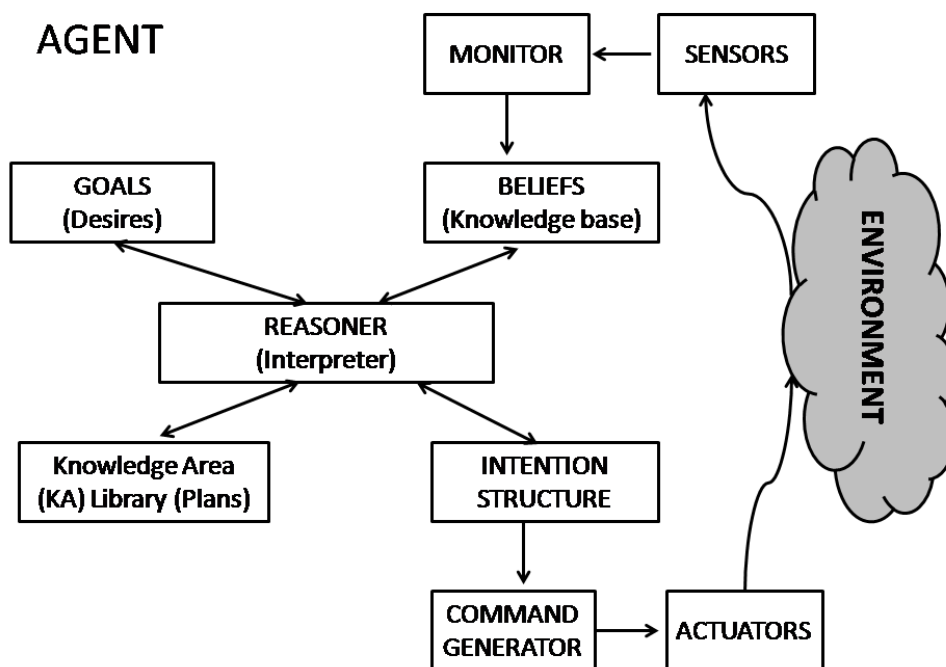


Fig. 4: Architettura PRS

Come evidenzia la figura, l'architettura permette di costruire un modello del mondo del contesto tramite interazione con i sensori. Le azioni possono essere eseguite seguendo il modulo Intentions. Al centro dell'architettura è collocato l'interprete o reasoner il quale, dato lo stato corrente di Beliefs, seleziona un goal da conseguire e produce un piano di esecuzione per raggiungere l'obiettivo. PRS verifica iterativamente le ipotesi del piano durante la sua esecuzione. Questo implica che è in grado di operare in ambienti dinamici. [3] I piani sono predefiniti per le azioni possibili nel contesto così da avere un'architettura più snella priva della necessità di generare piani. Tale caratteristica rende l'architettura ad agenti PRS ideale per costruire agenti per il controllo di robot mobili.

In PRS ogni agente è dotato di una plan library che rappresenta l'insieme delle informazioni procedurali ossia dei meccanismi che l'agente può adottare per realizzare le proprie intenzioni, il

know-how. Le opzioni disponibili all'agente sono calcolate direttamente dai piani di cui dispone l'agente stesso. Dunque un agente privo di piani risulta privo di opzioni.

[7] PRS reagisce continuamente alle percezioni dal mondo e agli eventi che in esso si susseguono selezionando piani da adottare come intenzioni.

I piani rappresentano la caratteristica principale dei sistemi PRS e comprendono diverse tipologie di componenti:

- trigger o invocazione: specifica cosa un agente debba percepire per adottare un piano che diventerà un competitore per la selezione rivolta all'esecuzione;
- plan context: il contesto dato dai Beliefs è necessario affinché l'agente possa adottare il piano per l'esecuzione;
- plan body: cattura le conoscenze procedurali rappresentandole come un OR-tree con archi etichettati con formule che possono essere: primitive di azioni esterne ( come le chiamate di funzioni C++ ), azioni interne ( aggiunte di fatti alla struttura dati Belief ), interrogazioni sui goals che sono mappati nelle credenze dell'agente o conseguimento di goals che generano nuovi sotto-goals.

All'inizio dell'esecuzione gli agenti sono inizializzati con un insieme di piani, goals, beliefs, una coda di eventi vuota e nessuna intenzione. Le operazioni dell'agente possono essere riportate come segue:

- Percepire l'environment e aggiornare l'insieme degli eventi;
- Per ogni evento generare un insieme di piani le cui trigger condition corrispondono all'evento. Questi rappresentano i piani rilevanti di un evento;
- Per ogni evento selezionare il sottoinsieme di piani rilevanti le cui condizioni di contesto sono soddisfatte dalle credenze correnti dell'agente. Tali piani sono detti piani attivi;
- Da quest'ultimo insieme selezionarne uno per l'esecuzione. Questo piano rappresenta un'intenzione;
- Includere l'intenzione nella struttura corrispondente creando un nuovo stack oppure collocandola in cima allo stack esistente;
- Selezionare uno stack intenzione, prendere l'intenzione più in alto ed eseguire le azioni previste;

Le prime applicazioni di PRS avevano come scopo il monitoraggio e la rilevazione di errori nei sistemi Reaction Control System (RCS) sugli shuttle della NASA. L'RCS forniva forze propulsive ad una serie di eliche di jet e controlli dello space shuttle.

Successivamente fu impiegato in applicazioni di gestione del controllo della rete chiamate Interactive Real-Time Telecommunications Network Management System (IRTNMS) per Telecom Australia.

## 5. Conclusioni

I vantaggi di PRS e IRMA risiedono nella loro costruzione sulla base di un modello concettuale di relazioni tra belief, desire, intention e piani. Tuttavia nessuna delle due architetture guida lo sviluppatore su come far prendere decisioni agli agenti sulla base dei propri stati mentali. Inoltre, sebbene sia possibile simulare comportamenti reattivi costruendo piani idonei a fronteggiare il susseguirsi di eventi, questo non rappresenta il modo migliore per modellare la reattività poiché le risposte reattive possono essere eseguite soltanto alla fine di ogni ciclo operativo dell'agente.

## 6. Bibliografia

- [1] [www.wikipedia.org](http://www.wikipedia.org)
- [2] Nicholas R. Jennings, Michael J. Wooldridge. *Agent technology: foundations, applications, and markets*
- [3] M. Tim Jones. *Artificial Intelligence: A Systems Approach*
- [4] Jörg P. Müller, Anand Rao, Munindar Paul Singh. *Intelligent agents V: agent theories, architectures, and languages*
- [5] Michael J. Wooldridge, Jörg P. Müller, Milind Tambe. *Intelligent agents II: agent theories, architectures, and languages. Subseries of Lecture Notes in Computer science*
- [6] Michael J. Wooldridge. *Reasoning about rational agents*
- [7] Michael Luck, Michael M. Luck, Ronald Ashri, Mark D'Inverno. *Agent-based software development*
- [8] Jörg P. Müller. *The design of intelligent agents: a layered approach*
- [9] Anand S. Rao, Michael P. Georgeff. *BDI Agents: From Theory to Practice*