

# Agenti Intelligenti

Lorenzo D'Isidoro

lorenzo.disidoro@gmail.com

Intelligenza Artificiale e Sistemi Informatici A.A. 2022/2023

# Contents

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Introduzione agli agenti intelligenti</b>	<b>4</b>
2.1	La Computazione . . . . .	4
2.2	Gli Agenti . . . . .	4
2.3	Esempio: Curiosity . . . . .	5
2.4	Caratteristiche degli Agenti . . . . .	5
2.5	Flessibilità . . . . .	6
2.5.1	Agente Reattivo . . . . .	6
2.5.2	Agente Proattivo . . . . .	7
2.5.3	Agente Sociale . . . . .	7
2.5.4	Conclusioni . . . . .	7
<b>3</b>	<b>Il Triangolo della Computazione</b>	<b>8</b>
3.1	Decomposizione su Process o Thread . . . . .	8
3.1.1	Business Processes . . . . .	9
3.2	Decomposizione su Oggetti e Dati . . . . .	9
3.2.1	Actors e Active Object . . . . .	10
3.3	Il Paradigma Agent-Oriented . . . . .	10
<b>4</b>	<b>Agent Architectures</b>	<b>12</b>
4.1	Agenti con ragionamento deduttivo . . . . .	12
<b>5</b>	<b>Agenti Razionali</b>	<b>14</b>
5.1	Sistema Intenzionale . . . . .	14
5.2	Practical Reasoning (Ragionamento su Azioni) . . . . .	15
5.3	Proattività e Intenzioni . . . . .	15
5.4	Realizzazione di Agenti che Sfruttano il Practical Reasoning . . . . .	16

# 1 Introduzione

Il corso ha l'obiettivo di introdurre gli aspetti principali dei sistemi multiagente, ossia sistemi composti di elementi computazionali che interagiscono, noti come agenti. Gli agenti sono sistemi computazionali capaci di eseguire azioni in modo autonomo, e di interagire con altri agenti svolgendo attività sociali come cooperazione, coordinamento, negoziazione. I sistemi multi agente costituiscono una metafora naturale per modellare un ampio spettro di “artificial social systems”. Nella prima parte del corso vengono forniti gli strumenti metodologici per comprendere i sistemi multiagente discutendo le architetture di singoli agenti e le principali problematiche legate all'interazione fra agenti. La seconda parte del corso presenta alcuni linguaggi e ambienti specifici per sistemi multiagente, in modo da consentire agli studenti di implementare alcuni esempi significativi. Pagina del corso.

## 2 Introduzione agli agenti intelligenti

Parte di questo documento fa riferimento ai lucidi del corso che sono parzialmente basati sul libro di testo del corso “M. Wooldridge, An Introduction to Multi-Agent Systems, Wiley, 2009” e sul materiale fornito da Alberto Martelli dell’Università degli Studi di Torino.

### 2.1 La Computazione

Prima di introdurre gli agenti intelligenti sarà utile discutere dei cinque trend che hanno caratterizzato la storia della computazione, ovvero:

- Ubiquità: La continua riduzione dei costi di elaborazione ha permesso la diffusione di sempre più dispositivi.
- Interconnessione: I sistemi informatici oggi non esistono più da soli, ma sono collegati in rete in grandi sistemi distribuiti, quindi sono sempre più interconnessi (e.g. LAN, Internet, ...)
- Intelligenza e Delega: I computer fanno di più per noi, senza un nostro intervento (e.g. sistemi a guida autonoma, ...)
- Human Oriented: Spostarsi da una vista orientata macchina della programmazione verso concetti e metafore che sono più vicine al modo con cui noi comprendiamo e vediamo il mondo, quindi l’utilizzo di astrazioni più familiari per l’uomo (e.g. Object Oriented Programming per gli sviluppatori)

Delegazione e intelligenza implicano la necessità di **costruire sistemi informatici in grado di agire efficacemente per nostro conto.**

- La capacità di agire dei sistemi informatici in modo **indipendente**
- La capacità dei sistemi informatici di agire in un modo che **rappresentino nel migliore dei modi i nostri interessi** durante l’interazione con altri esseri umani o sistemi
  - Interconnessione e distribuzione, accoppiate con la necessità di sistemi che rappresentino nel migliore dei modi i nostri interessi, portano a sistemi che possono cooperare e raggiungere accordi o competere.

Tutte queste tendenze hanno portato alla nascita di un nuovo campo in informatica: **Sistemi multi-agente.**

### 2.2 Gli Agenti

Un **agente** è un sistema di computazione capace di agire in modo **indipendente** per conto di un utente o proprietario, capendo cosa deve essere fatto per

soddisfare gli **obiettivi** di progettazione, piuttosto che essere costantemente informato.

Un **sistema multi agente (MAS)** consiste in una serie di agenti, che interagiscono l'uno con l'altro. Per interagire con successo, richiedono la capacità di **cooperare**, **coordinarsi** e **negoziare** tra loro, in modo analogo a quanto fanno le persone. Un MAS quindi ha il compito di rispondere a queste domande:

- Come possiamo creare agenti capaci di agire in modo indipendente, autonomo, in modo che possano svolgere con successo i compiti che gli deleghiamo?
- Come possiamo creare agenti capaci di interagire (cooperare, coordinare, negoziare) con altri agenti con il fine di portare a termine con successo i compiti delegati, soprattutto quando non si può presumere che gli altri agenti condividano gli stessi interessi/obiettivi?

Il primo problema è di progettazione/design di un agente, quindi micro, il secondo 'e di progettazione/design di una società di agenti, quindi macro.

### 2.3 Esempio: Curiosity

Curiosity è un rover robotico delle dimensioni di un'auto che esplora il cratere Gale su Marte come parte della missione Mars Science Laboratory (MSL) della NASA. Curiosity è stato lanciato da Cape Canaveral il 26 novembre 2011.

Gli obiettivi del rover includono: indagini sul clima marziano e geologia; valutazione del fatto che il sito selezionato offra condizioni ambientali favorevoli per vita microbica, inclusa un'indagine sul ruolo dell'acqua; etc.

**Curiosity ha operato in modalità autonoma per circa 25 giorni.**

Per far ciò si è deciso di basare la sonda su **algoritmi di pianificazione** e non solo su sistemi di controllo ingegneristici.

### 2.4 Caratteristiche degli Agenti

In ultima istanza possiamo vedere un agente come un **sistema di computazione** capace di agire **autonomamente**, in maniera indipendente e in modo **flessibile**, in un qualche **ambiente** con il fine di raggiungere gli **obiettivi** per cui è stato progettato.



Un esempio di agente intelligente triviale, quindi banale, non interessante, è il termostato. Infatti il termostato decide cosa fare, ovvero accendere, spegnere il riscaldamento o non fare nulla, sulla base della temperatura rilevata dall'ambiente.

## 2.5 Flessibilità

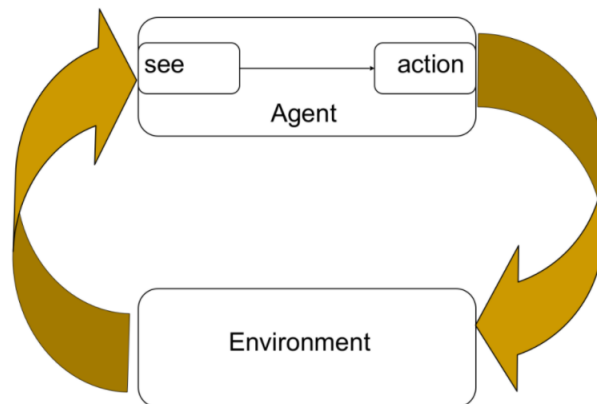
Come anticipato nel precedente paragrafo un agente intelligente è un sistema di computazione capace di esibire azioni autonome in modo *flessibile* in un ambiente. Con flessibile si intende:

- Reattivo
- Proattivo
- Sociale

### 2.5.1 Agente Reattivo

Un agente reattivo è un sistema che mantiene una costante interazione con l'ambiente e risponde ai cambiamenti che occorrono su di esso (in tempo perché la risposta sia utile). Bisogna tenere a mente però che l'ambiente reale è dinamico e le informazioni a volte possono essere incomplete.

Un agente reattivo può essere implementato tramite semplici regole condizionali del tipo: *“if car-in-front-is-braking then initiate-braking”*, se l'agente percepisce che l'auto di fronte sta frenando, allora inizia immediatamente a frenare per evitare una possibile collisione, senza perdere tempo a ragionare, in sintesi quello che viene fatto è, per dato evento/stimolo → una regola di risposta.



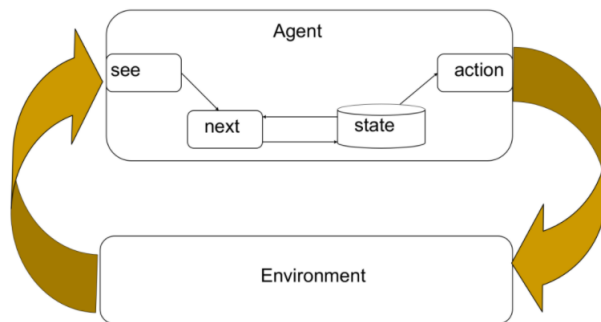
### 2.5.2 Agente Proattivo

Generalmente si vuole che gli agenti siano in grado di agire **autonomamente**, quindi generare e tentare di raggiungere gli obiettivi, non solo guidati dagli eventi, ma che siano in grado di prendere l'iniziativa.

Per agire autonomamente l'agente ha bisogno di informazioni:

- su come l'ambiente evolve
- su come le proprie azioni impattano sull'ambiente
- sull'obiettivo in modo tale che sia in grado di agire per raggiungerlo

Nel tentare di raggiungere un obiettivo l'agente deve essere in grado di ragionare su **piani**.



Si desidera che gli agenti siano reattivi e che rispondano ai cambiamenti per tempo e che lavorino in modo sistematico verso obiettivi di “lungo termine”, ma *progettare un agente che bilanci reattività e proattività è ancora un problema di ricerca aperto*.

### 2.5.3 Agente Sociale

Per abilità sociale di un agente intelligente si intende l'abilità di interagire con altri agenti (anche umani) attraverso un qualche tipo di linguaggio di comunicazione (agent-communication language, ACL) e cooperare con essi.

### 2.5.4 Conclusioni

Un **programma non è un agente** perché il suo output non ha effetto su quello che lui percepisce in seguito e inoltre non c'è continuità temporale.

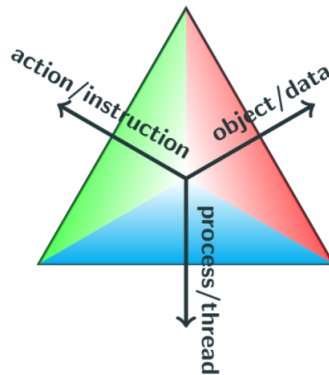
### 3 Il Triangolo della Computazione

L'ingegneria del Software aspira a un software di qualità, quindi un software che cerca di raggiungere i seguenti goal:

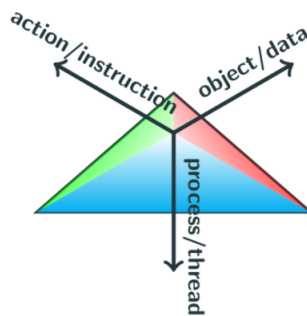
- Correttezza (correctness)
- Robustezza (robustness)
- Estensibilità (extensibility)
- Riusabilità (reusability)

Un ingrediente cruciale per raggiungere questi obiettivi è una corretta **modularizzazione** dell'architettura del software. Il triangolo di Meyer fornisce un terreno comune per confrontare diversi approcci alla modularizzazione, dove le "forze" in gioco sono:

- Process o Thread: CPU fisica, un processo o un thread.
- Action o Istruzione: operazioni che compongono il calcolo (linguaggio macchina, operazioni fino alle subroutine).
- Object: strutture dati a cui le azioni si applicano.



#### 3.1 Decomposizione su Process o Thread





**Pro:**

- Semplice ed intuitivo: decomposizione un programma in funzioni, sotto-funzioni e procedure.
- Algorithmic oriented: appropriato quando viene specificato un unico obiettivo principale.

**Contro:**

- Difficilmente manutenibile: difficile incorporare nuovi "obiettivi principali", quindi allo **sviluppo incrementale**.
- Difficilmente scalabile in presenza di dati condivisi e processi concorrenti.

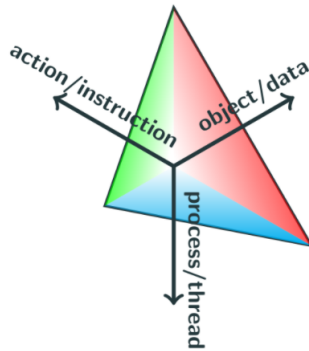
### 3.1.1 Business Processes

Si tratta di un modello che crea una rappresentazione esplicita delle attività di una società/azienda e descrivere come un insieme di attività interconnesse porta a un risultato preciso e misurabile in risposta a un evento esterno. Può essere definito come una sorta di decomposizione funzionale dove il goal si raggiunge a seguito dell'esecuzione di un processo, diviso in sotto-processi.

Un contro è quello dovuto al fatto che l'enfasi è sul processo (vista incentrata sull'attività):

- Se il dato deve influenzare il processo non funziona.
- Nessuna astrazione adeguata per acquisire dati manipolati lungo i flussi.

## 3.2 Decomposizione su Oggetti e Dati



**Pro:**

- Si adatta bene allo sviluppo incrementale, quando i requisiti non sono noti a priori.
- Gli oggetti hanno una vita propria, indipendente dei processi li utilizza.

- Operazioni sui dati: fornire le azioni di su cui è possibile operare su di essi.

**Contro:**

- Gli oggetti sono passivi: i processi esterni prendono le decisioni su quali azioni invocare oggetti.
- Nessun disaccoppiamento tra l'uso di un oggetto e la gestione di tale oggetto.
- Per la gestione dei thread si deve ovviare associando ad ogni thread un oggetto.

### 3.2.1 Actors e Active Object

Data la complessità del tentativo di coordinare gli accessi a un oggetto da più thread, a volte ha più senso evitare del tutto il multi threading e si può utilizzare il pattern degli **Actors e Active Object**.

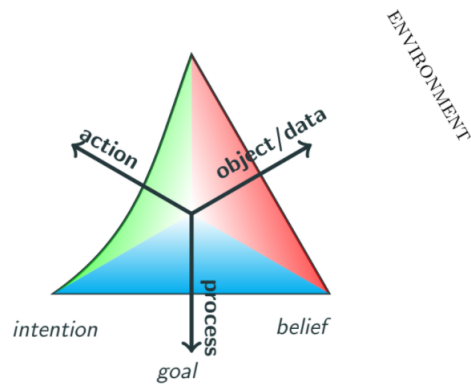
**Active Object** in generale è un pattern per gestire la concorrenza in cui si tenta di separare l'invocazione di un metodo dalla sua esecuzione.

Gli attori sono a un livello di astrazione più elevato rispetto a threads, infatti un **Actor** è un semplice oggetto che riceve dei messaggi e reagisce ad essi con un'azione eseguita sul thread corrente, quindi ad un Actor è associato il suo thread. E' possibile che l'attore decida di eseguire l'attività di azione su un pool di thread.

Uno dei problemi del modello ad attori è quello che non affronta adeguatamente il problema del coordinamento (sebbene sono state proposte estensioni). Inoltre non supporta la progettazione e la modularizzazione dei processi che utilizzano questi oggetti, perché processi esterni agli Attori.

## 3.3 Il Paradigma Agent-Oriented

La programmazione orientata agli agenti (AOP) è un paradigma di programmazione in cui la costruzione del software è centrata sul concetto di agenti software. A differenza della programmazione orientata agli oggetti che ha gli oggetti, che forniscono metodi con parametri variabili, un agente ha uno stato che è costituito da componenti come **convinzioni**, **decisioni**, capacità e **obblighi**; per questo motivo lo stato di un agente è chiamato stato mentale.



Il paradigma dell'agente si basa su due astrazioni di prima classe:

- **Agenti, orientato al processo:**
  - Sono autonomi, quindi il ciclo deliberativo finalizzato al raggiungimento di un obiettivo (proattività).
  - Sono in grado di percepire e manipolare il loro ambiente.
- **Ambiente, orientato ai dati/oggetto.**

Quindi per evidenziare la  *differenza tra Agenti e Oggetti* sottolineiamo che gli Oggetti

- Non hanno il controllo sul proprio comportamento (passivo/reattivo)
- Non mostrano un comportamento flessibile (nessun ciclo deliberativo)
- Sono a thread singolo

La principale  *differenza tra Agenti e Attori*: Gli attori non sono intenzionali/proattivi.

## 4 Agent Architectures

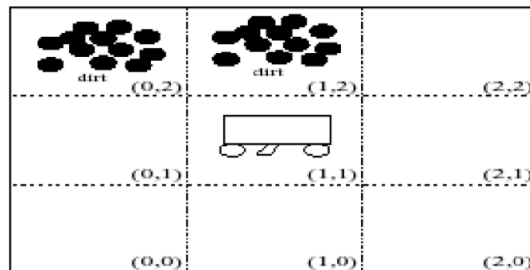
Il classico approccio per costruire agenti intelligenti è quello di vederli come casi particolare di sistemi basati sulla conoscenza, questo paradigma è noto come “symbolic AI”. L’architettura di un agente deliberativo, infatti, contiene una esplicita rappresentazione (modello simbolico) dell’ambiente e prende decisioni (ad esempio, quale azione eseguire) attraverso un ragionamento simbolico.

Sono due i problemi da affrontare:

- Il problema della **trasduzione**: ovvero il problema della traduzione del mondo reale, dell’ambiente, in una descrizione simbolica accurata, in tempo perché sia utile (si pensi ai problemi di visione, riconoscimento del parlato, apprendimento, ...).
- Il problema della **rappresentazione/ragionamento**: ovvero il problema di come rappresentare simbolicamente le informazioni su entità e processi complessi del mondo reale, e come fare in modo che gli agenti ragionino con queste informazioni in tempo perché i risultati possano essere utili.

### 4.1 Agenti con ragionamento deduttivo

Un esempio è quello del robot delle pulizie, il cui obiettivo è quello di pulire tutto lo sporco presente in un area della casa. Il primo passaggio è sicuramente quello di codificare l’ambiente, in questo caso rappresentiamo la stanza come una matrice.



*Ma come un agente può decidere cosa fare utilizzando tecniche di ragionamento?*

L’idea di base ‘è usare la logica per codificare una teoria permettendo di ottenere la migliore azione da eseguire in qualsiasi situazione. Definiamo 3 predicati per risolvere il problema:

- $In(x, y)$ , l’agente è nella posizione  $(x, y)$ .
- $Dirt(x, y)$ , c’è dello sporco nella posizione  $(x, y)$ .
- $Facing(d)$ , l’agente è rivolto in direzione  $d$ .

Possibili azioni:  $Ac = turn, forward, suck$ .

Definiamo le regole per determinare cosa fare:

$$In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) \Rightarrow Do(forward)$$

$$In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) \Rightarrow Do(forward)$$

...

Possibili **problemi** possono essere come scoprire l'ingresso della videocamera in  $Dirt(0, 1)$  e che il processo decisionale basato sulla logica del primo ordine è indecidibile.

## 5 Agenti Razionali

Quando si spiega l'attività umana, è spesso utile fare dichiarazioni come le seguenti: *"Janine ha preso l'ombrello perché lei crede che possa piovere"*. Queste affermazioni fanno uso di una **psicologia popolare** (folks psychology) che fa riferimento al complesso di teorie, credenze e abilità cognitive e comprende la capacità di intuire il funzionamento della mente e di predirne il comportamento.

L'azione *"Janine ha preso l'ombrello"* a seguito della credenza/intuizione *"lei crede che possa piovere"*.

### 5.1 Sistema Intenzionale

Il filosofo Daniel Dennett ha coniato il termine **sistema intenzionale** per descrivere entità *"Il cui comportamento può essere previsto dal metodo di attribuzione di credenze, di desideri e di acume razionale"*. Quindi secondo questa definizione è opportuno chiedersi se è legittimo o utile attribuire credenze e desideri ai sistemi informatici.

L'accademico Jhon McCarthy afferma che ci sono occasioni in cui ascrivere una macchina come sistema intenzionale è appropriata, ad esempio, quando l'iscrizione ci aiuta a capire la struttura della macchina, il suo comportamento passato o futuro, o come ripararla o migliorarla.

Dobbiamo tenere a mente però che più i sistemi di calcolo diventano complessi, più hanno bisogno di astrazioni e metafore potenti per spiegare il loro funzionamento, le spiegazioni di basso livello spesso diventano impraticabili. **Le nozioni intenzionali sono quindi astrazioni**, che forniscono un modo semplice e familiare per descrivere, spiegare e prevedere il comportamento di sistemi complessi. Quindi qualsiasi sistema, più o meno complesso, può essere definito come sistema intenzionale (e.g. interruttore della luce, Amazon Alexa, robot aspirapolvere, ...) le quali nozioni intenzionali sono tendiamo a vederle come astrazioni, non ci preoccupiamo del loro funzionamento nel dettaglio.

Ma anche importanti sviluppi nella computazione sono basati su astrazioni, basti pensare ai linguaggi di programmazione che ci astraggono dal linguaggio macchina, e anche i concetti di:

- Oggetti
- Procedure/funzioni
- Tipi di dati

Gli agenti intelligenti e, in particolare, i sistemi intenzionali, rappresentano un ulteriore (e potente) astrazione.

## 5.2 Practical Reasoning (Ragionamento su Azioni)

Il practical reasoning è il ragionamento sulle azioni, sul processo di capire cosa fare, l'uso della ragione per decidere come agire. Il practical reasoning si distingue dal Theoretical Reasoning, che riguarda solo le credenze non le azioni.

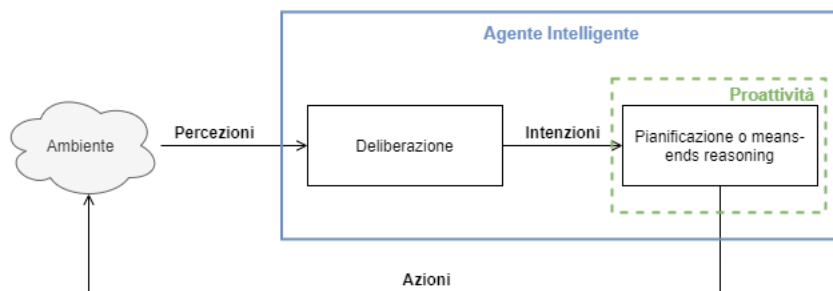
Il Practical Reasoning è caratterizzato da due attività:

- Deliberazione: Quale stato di cose vogliamo raggiungere (l'output sono le **intenzioni**).
- Pianificazione o means-ends reasoning: Come possiamo raggiungere gli obiettivi (l'output è il soddisfacimento delle intenzioni, raggiungere lo stato di cose).

Deliberazione e means-ends reasoning sono processi computazionali, come tali in tutti gli agenti reali questi processi avranno risorse limitate (e.g. il tempo, lo spazio, ...). Quindi il calcolo è una risorsa preziosa per gli agenti perché serve per controllare il suo ragionamento ed è importante che non agiscano a tempo indeterminato, perché anche il tempo è una risorsa. **Con intenzioni ci riferiamo allo stato di cose che un agente ha scelto di raggiungere.**

## 5.3 Proattività e Intenzioni

Anche se un agente è un sistema informatico è intelligente, non perché ha consapevolezza o capacità extra-programma, ma se riusciamo ad attribuirgli alcune caratteristiche, tra cui, la **proattività**. Il ruolo delle intenzioni è quello di favorire la proattività, cioè tendono portare ad agire, all'azione.



Il filosofo Brattman osserva che le intenzioni giocano un ruolo molto più forte nell'influenzare l'azione rispetto ad altri atteggiamenti proattivi come il desiderio. Infatti **le intenzioni non possono essere in contrasto, mentre invece i desideri si.**

Quindi possiamo dire che le intenzioni hanno le seguenti proprietà:

- **Problemi:** Le intenzioni pongono problemi agli agenti, se ho un intenzione  $\varphi \Rightarrow \text{alloco risorse} : \exists \text{ piano per eseguire } \varphi$

- **Le intenzioni vincolano il futuro practical reasoning**, quindi non posso generare intenzioni che vanno in conflitto tra loro: Se ho il desiderio di andare in piscina ma devo andare a lavoro, la piscina rimane un desiderio, il lavoro l'intenzione.
- **Persistenza**: Gli agenti monitorano il successo delle loro intenzioni e sono inclini a riprovare se i loro tentativi falliscono, solo se la motivazione dell'intenzione non sussiste più, è razionale abbandonare tale intenzione.
- Gli agenti credono che le loro intenzioni siano possibili. Se voglio andare in piscina credo che la piscina sia aperta e che io posso fare la mia attività.
- Gli agenti non credono che non porteranno avanti le loro intenzioni.
- In determinate circostanze, gli agenti credono che realizzeranno le loro intenzioni..
- **Effetti collaterali**: Gli agenti non devono necessariamente avere intenzione su tutti gli effetti. collaterali delle loro intenzioni (e.g. ho una carie, affronto il dolore per curarla).

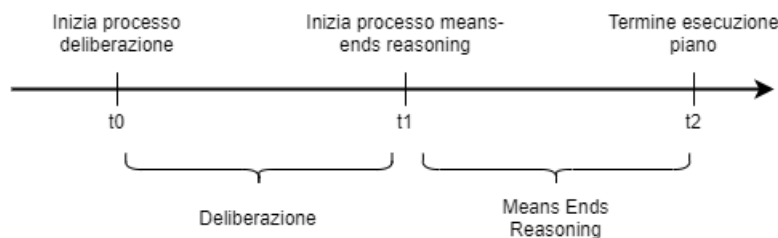
## 5.4 Realizzazione di Agenti che Sfruttano il Practical Reasoning

Agent Control Loop Version 1:

```
while true do:
  # deliberazione
  observe the world;
  update internal world model;
  deliberate about what intention to achieve next;

  # pianificazione
  use means-ends reasoning to get a plan;
  execute the plan;
```

I processi di deliberazione e means-ends reasoning non sono istantanei, hanno un costo in termini di tempo, questo significa che quando ottengo un piano potrebbe non essere più quello che mi serve. Osservando questa timeline possiamo fare alcune deduzioni:





Definiamo come tempo per deliberare:

$$T_{deliberazione} = t_1 - t_0$$

$$T_{means-ends-reasoning} = t_2 - t_1$$

La **deliberazione è ottimale** se al tempo  $t_1$ , l'agente ha selezionato l'intenzione da raggiungere che sarebbe stata ottimale se fosse stata raggiunta all'istante  $t_0$ . A meno che il tempo  $T_{deliberazione}$  sia estremamente piccolo, l'agente corre il rischio che l'intenzione selezionata non sia più ottimale nel momento in cui l'agente l'ha deliberata. Ma la fase di deliberazione è solo metà del problema: l'agente deve ancora determinare come realizzare l'intenzione.

Quindi, **l'agente avrà un comportamento complessivo ottimale** nelle seguenti circostanze:

- Quando il tempo impiegato per i processi di deliberazione e di means-ends reasoning è incredibilmente piccolo.
- Quando l'ambiente è garantito rimanere statico mentre l'agente sta eseguendo i processi di deliberazione e di means-ends reasoning.-
- Quando un'intenzione ottimale se raggiunta al tempo  $t_0$  (momento in cui si osserva l'ambiente) è garantita rimanere ottimale fino al tempo  $t_2$  (momento in cui l'agente ha determinato le azioni per raggiungere l'intenzione).