ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

# PROGETTAZIONE E SVILUPPO DI UN BDI PERSONAL ASSISTANT AGENT PER LA GENERAZIONE DI WARNING IN AMBITO MEDICO: IL SISTEMA TRAUMA TRACKER COME CASO DI STUDIO.

Elaborata nel corso di: Programmazione avanzata e paradigmi

*Tesi di Laurea di*:
SILVIA VANDI

*Relatore*:
Prof. ALESSANDRO RICCI
*Co-relatori*:
Dott. ANGELO CROATTI

ANNO ACCADEMICO 2016/2017
SESSIONE III

# PAROLE CHIAVE

**Personal Assistant Agent**

**Modello BDI**

**Trauma Tracker**

**Warning Agent**

**Agent Oriented Programming**

Ad Echo IV

# Indice

# Introduzione

Siamo ormai abituati a vivere in un mondo in completa e continua evoluzione. Si è sempre alla ricerca di nuove soluzioni soprattutto nel campo tecnologico. Lo sviluppo in questo ambito è in continuo fermento portando ad un'evoluzione incrementale della tecnologia. Nell'ultimo decennio si è assistito in particolare ad un'importante sviluppo per quanto riguarda le tecnologie wearable e mobile [37], aprendo gli orizzonti a differenti metodi di interazione tra l'utente e le macchine, nel quali principalmente il computer diventa collaboratore attivo ed intelligente.

Al giorno d'oggi avendo a nostra disposizione apparecchi mobili sempre più potenti e performanti [41], paragonabili a normali personal computer, si può pensare alla programmazione ad alto livello anche in questo campo. La programmazione di applicazioni mobile diventa quindi più completa e complessa, in quanto bisogna tenere in considerazione nuovi fattori come la reattività, la proattività, la flessibilità e l'interazione. Si entra così nel mondo dei Personal Assistant Agent, ovvero degli assistenti personali in grado si aiutare l'utente nel quotidiano, sia per quanto riguarda lo svolgimento di mansioni al suo posto, sia per una semplice facilitazione nello svolgimento di certi lavori. L'aiuto che viene proposto è di diversa natura[44], spaziando quindi su domini applicativi eterogenei. Esistono infatti diversi sistemi di questo tipo in grado di fornire per esempio assistenza agli anziani, applicazioni d'ufficio (progetto CALO, Electric Elves, STEAM), applicazioni sociali (progetto RADAR, ANTIPA, COLLAGEN, NewT), applicazioni militari o per il turismo. Ogni applicazione ha un obiettivo diverso: alcune cercano di gestire la coordinazione tra le persone, altre aiutano l'utente nella gestione delle email e dei loro impegni, altre ancora si basano sul fattore intrattenimento ed il turismo suggerendo per esempio quali possono essere posti interessanti per l'utente in base alle proprie abitudini e ricerche, altre ancora hanno il compito di semplificare la consultazione e la ricerca di ar-

ticoli di giornale filtrando gli argomenti interessanti per l'utente. Il fattore comune è la volontà di aiutare l'uomo nel compimento del proprio lavoro. L'aiuto può essere compiuto sia dal punto di vista qualitativo, apportando un' aiuto nello svolgimento di task difficili, oppure può essere un aiuto di tipo quantitativo, ovvero volto a migliorare le prestazioni e la velocità di lavoro dell'individuo. Tramite questi Personal Assistant Agent si vuole, sostanzialmente, diminuire lo sforzo umano necessario ed il carico cognitivo richiesto per svolgere un compito.

Su questa stessa linea di pensiero si colloca Trauma Tracker[32], un assistente digitale per il campo medico. Nello specifico prende il nome di Personal Medical Digital Assistant Agents (PMDAs) ed ha il compito di assistere i medici del Trauma Center per quanto riguarda il tracciamento e la documentazione del loro lavoro. Il progetto è stato sviluppato all'interno dell'università di Bologna ed è attualmente in uso presso l'ospedale Bufalini di Cesena. Essendo il Trauma Center uno dei reparti all'interno dell'ospedale più critico e con la gestione più complicata, diventa un perfetto dominio applicativo per un Personal Assistant Agent, in quanto ci si trova di fronte ad una quantità onerosa di eventi tale da lasciare poco tempo ai medici per il coordinamento del lavoro. Il sistema di assistenza ha il compito di supportare i medici con lo scopo di ridurre l'errore umano e risparmiare tempo incrementando le performance del team di medici.

Trauma Tracker è in grado di creare report automaticamente, richiedendo un'interazione irrisoria con l'utente, in questo caso il Trauma Leader. Questi report rappresentano la documentazione del trauma, e vanno generati per tenere traccia di ogni manovra effettuata sul paziente, di ogni farmaco somministrato e delle caratteristiche vitali di esso durante l'intero trattamento. Tramite l'ausilio di dispositivi elettronici e sistemi di questo genere si ottengono report più accurati, precisi e meno time consuming rispetto al lavoro manuale che dovrebbe compiere il medico senza di loro. Alla base di tale applicazione troviamo un sistema multi-agente e conseguentemente la programmazione ad agenti ed il modello BDI. Ci troviamo quindi davanti ad un chiaro esempio di programmazione ad alto livello portata su dispositivi mobili e wearable.

Il paradigma AOP, Agent Oriented Programming[29], si focalizza sul concetto di agente, ovvero un'entità intelligente in grado di percepire l'ambiente che lo circonda e di reagire eseguendo azioni in seguito agli eventi percepiti. Gli agenti sono entità autonome pro-attive che vivono all'interno

di un environment popolato da artefatti, entità passive che possono essere utilizzate dagli agenti per raggiungere i loro scopi. La caratteristica principale di questo tipo di programmazione è che il programmatore deve fornire agli agenti delle conoscenze e dei lavori da svolgere, senza dover specificare come essi si debbano comportare esplicitando il susseguirsi delle azioni. Gli agenti sono infatti in grado di manipolare le loro conoscenza, organizzate in piani, per raggiungere il loro obiettivo. In questo paradigma, il meccanismo di controllo integrato, implementa un modello razionale corrispondente alla nostra comprensione intuitiva di esprimerci tramite credenze e desideri, grazie al quale sono necessarie meno conoscenze specifiche per utilizzarlo. Seguendo questa linea di pensiero sono stati ideati diversi modelli computazionali ed architetturali, tra i quali troviamo il modello Belief-Desire-Intention (BDI), il quale venne creato seguendo un ragionamento che si basa sul modello del comportamento umano nel mondo quotidiano.

Il fulcro principale della tesi è l'estensione del progetto Trauma Tracker ed un approfondimento particolare sul modello BDI, per capirne a pieno le funzionalità e potenzialità verificandone l'adeguatezza per il progetto in questione. La tesi inizia con una panoramica sullo stato dell'arte dei Personal Asistant Agent ed un dettaglio sullo stato attuale del progetto Trauma Tracker con una particolare attenzione alla sua collocazione all'interno di questo dominio. Una volta analizzata la struttura e le caratteristiche di Trauma Tracker si passa ad esplorare il background e le conoscenze pregresse utili per lo sviluppo. Essendo nata l'esigenza di un'estensione dell'attuale sistema si passa all'analisi e alla progettazione di quest'ultima partendo dai requisiti forniti dai medici dell'ospedale Bufalini di Cesena.

# Capitolo 1

# Il progetto Trauma Tracker

Nel seguente capitolo viene analizzato il caso di studi considerato nella tesi.

## 1.1 Panoramica del progetto

Il progetto Trauma Tracker nasce da una collaborazione tra l'università di Bologna ed il Trauma Center dell'ospedale Bufalini di Cesena, con l'obiettivo di creare un assistente digitale per i medici in grado di assisterli nel loro lavoro [32]. Essendo un'applicazione in campo medico, questo assistente prende il nome di Personal Medical Digital Assistant Agents (PMDAs), ed ha il compito sia di tenere traccia dei parametri vitali del paziente, di tutte le manovre effettuate e delle medicine somministrate, sia di generare automaticamente un report con tutte le informazioni raccolte. Tra i vari dispositivi utilizzati per questa applicazione quelli mobile e wearable occupano una posizione di rilievo, in quanto sono utili per l'aumento dell'efficienza del sistema apportando il meno ingombro possibile ai medici che la utilizzano. Si parla quindi di sistemi hands-free o use-on-the-go, nei quali si può, in modo asincrono, ricevere informazioni e dati dall'applicazione senza distogliere l'attenzione dal lavoro che si sta svolgendo o senza avere una mano impegnata nell'interazione con il device.

La visione alla base di questo progetto è una visione che negli ultimi anni ha preso sempre maggiore concretezza, ed è quella di "augmented physicians working in augmented hospitals", ovvero l'utilizzo di dispositivi che sfruttano la realtà aumentata all'interno di ospedali che anch'essi usufruiscono di tali sistemi. La realtà aumentata è attualmente una delle tecnologie digitali

più promettenti e ha il potenziale per cambiare completamente l'assistenza sanitaria e la medicina quotidiana sia dal punto di vista dei medici sia per i pazienti.

Il dominio su cui è incentrato Trauma Tracker, il campo delle emergenze ospedaliere, è uno dei più critici e difficili da gestire. All'interno del Trauma Center ci si trova in un ambiente concitato ed impetuoso, dove vengono richieste infatti elevata reattività, coordinazione, risposte rapide e decisioni accurate e cruciali da prendere in breve tempo. In un contesto simile viene naturale pensare ad un Personal Assistant Agent per facilitare la gestione della situazione in numerosi modi diversi. Si possono pensare a sistemi in grado di gestire il coordinamento tra i vari componenti del team medico, sistemi in grado di suggerire ai medici quale potrebbe essere la scelta migliore da prendere in determinate condizioni, sistemi dedicati alla gestione dei dati raccolti, e tante altre possibilità che comunque hanno come fattore comune l'aiuto dell'individuo che li utilizza ed il conseguente miglioramento delle performance sia dal punto di vista qualitativo sia dal punto di vista del tempo impiegato nello svolgimento delle attività.

In questo campo tendenzialmente esiste un Trauma Leader, con il compito di dirigere tutta la sessione del trauma e le relative unità che lavorano al suo interno. Esso è il responsabile principale ed ha anche il compito di redigere la documentazione del trauma. Questa dovrebbe essere prodotta durante l'avvenimento del trauma stesso. Ciò implicherebbe che il medico, durante i momenti concitati in cui sta gestendo un trauma, dovrebbe annotarsi i dettagli sul suo lavoro perdendo tempo prezioso e concentrazione. La documentazione è cruciale, perchè viene utilizzata per fare analisi a posteriori e produrre statistiche sull'adeguatezza e le performance del personale. La creazione di documentazione nella maniera standard non è per niente semplice, in quanto spesso durante un trauma c'è un clima frenetico, dato dalla gravità della situazione, e rimane quindi poco tempo per la stesura dei report, oppure può succedere che per trattamenti specifici, particolarmente pericolosi per la vita del paziente, vengano eseguite più manovre contemporaneamente da diversi membri del team e quindi il monitoraggio di tutti non risulta banale. Attualmente, in alcuni paesi è possibile trovare all'interno dell'equipe di medici una persona con il compito di gestire il report, ovvero collezionare annotazioni o comunicazioni verbali durante il trauma per poi aiutare il Trauma Leader nella compilazione della documentazione a trauma ultimato. I fogli contenenti i report vengono poi inviati alla centrale

operativa dove vengono manualmente trascritti per inserirli nel database. Utilizzando questa procedura si ottengono report incompleti e spesso sbagliati dovuti dall'imprecisione dell'acquisizione dei dati e dall'errore umano che può avvenire nel processo di trascrizione, senza considerare la grande quantità di tempo impiegato ed il carico di lavoro richiesto.

Il sistema Trauma Tracker, pensato principalmente per risolvere il problema della documentazione, è in grado di fornire un supporto tecnologico per la compilazione dei report aumentandone l'accuratezza, la precisione e la velocità di stesura. Il primo prototipo di questo progetto è stato implementato seguendo il modello BDI, utilizzando come tecnologia agent-based JaCa-Android, la versione di JaCaMo per dispositivi Android. Le informazioni sul tracking ed i dati relativi al trauma vengono automaticamente raccolti in un server risiedente nella rete locale ospedaliera. Inoltre è stata creata una web app[33] che permette il consulto delle varie informazioni risiedenti sul server, creare e stampare report, ed inoltre generare informazioni statistiche basilari ricavate su di essi. I dati sono interamente disponibili per l'utilizzo tramite altre applicazioni ospedaliere che operano all'interno della stessa infrastruttura. Trauma Tracker è stata creata in maniera modulare[32] per garantire l'estensione in previsioni di future necessità. Per il momento l'applicazione è ad un primo livello di assistenza, si occupa infatti della gestione e tracking degli eventi ed altre informazioni sui pazienti. Si è pensato poi di innalzare il tutto ad livello successivo racchiudendo funzionalità per l'assistenza in tempo reale.

## 1.2 Struttura ed architettura del sistema attuale

### 1.2.1 Funzionalità fornite dal sistema

Trauma Tracker ha come obiettivo principale[33] quello di permettere una forma efficace di tracciamento per gli eventi che avvengono all'interno della struttura ospedaliera in cui il trauma viene trattato e, contemporaneamente, permettere di collezionare e gestire tutte le informazioni raccolte con lo scopo di produrre report dettagliati in grado di supportare l'analisi retrospettiva delle attività. Una caratteristica cruciale del sistema è che non interferisca con il lavoro dei medici ma, al contrario, lo agevoli e semplifi-

chi. A tal proposito l'applicazione è stata creata limitando il quantitativo di informazioni da inserire manualmente. Infatti Trauma Tracker è in grado di inferire automaticamente diverse informazioni riguardo al contesto, come per esempio la stanza dove una procedura ha avuto luogo può essere captato tramite l'utilizzo di Beacon[19], oppure la raccolta automatica dei parametri vitali del paziente nel momento in cui si inizia una nuova procedura o nel momento in cui gli si somministra un medicinale. Inoltre, data la naturale freneticità e complicatezza del contesto in cui opera, è molto importante che i comandi dell'interfaccia utente siano semplici, efficienti e di rapida comprensione, soprattutto per quanto riguarda i comandi per l'archiviazione delle annotazioni.

Le funzionalità e le caratteristiche presenti attualmente nel sistema sono le seguenti:

- L'utente che crea le note ed inserisce le informazioni nel sistema deve essere sempre chiaramente identificato. Tendenzialmente questo è il Trauma Leader, ma essendo una figura altamente dinamica il quale ruolo può cambiare diverse volte durante tutta la durata del trauma, è bene esplicare sempre l'utente.

- Ogni nota deve essere correlata dalle informazioni riguardanti il tempo (data ed ora) ed il luogo, ovvero la specifica stanza in cui si trova il Trauma Leader e il paziente al momento dell'inserimento della nota.

- Principalmente le informazioni da raccogliere sono quelle riguardanti le procedure attuate dai medici ed i farmaci somministrati al paziente.

- Le note possono essere arricchite da informazioni multimediali, come ad esempio foto, video o annotazioni vocali. Non devono mai mancare le informazioni riguardanti spazio e tempo.

- L'interfaccia utente deve essere di facile comprensione e rapido utilizzo. Deve essere immediato l'inserimento nel sistema dei parametri vitali del paziente e lo stato di salute, anche se in maniera qualitativa (se il battito cardiaco è normale o meno, se è in grado di respirare autonomamente o meno, se sta sanguinando, ecc). Questo serve soprattutto nella parte iniziale del trauma, ovvero non appena il paziente varca la soglia del pronto soccorso. Il sistema deve inoltre permettere

al Trauma Leader di annotare cambiamenti sullo stato di salute del paziente, in modo agevole e repentino.

- All'interno dell'ambiente ospedaliero è presente un sistema che, tramite l'utilizzo di sensori connessi al paziente, riesce a monitorarne i parametri vitali in tempo reale. Trauma Tracker deve essere in grado di interagire con esso.

- Il sistema è programmato per annotare automaticamente i parametri vitali del paziente con scadenze regolari ed ogni qualvolta viene effettuata una nuova procedura o medicinale. Il periodo di tempo preimpostato varia a differenza della stanza in cui si sta svolgendo il trauma, questo perchè si presuppone che la stanza sia un indice per la gravità dello stato di salute. Per esempio se si trovasse nella sala rossa (l'area del DEA dotata di apparecchiature tecnologicamente avanzate dedicata al trattamento di pazienti particolarmente critici) sarebbe sicuramente più a rischio rispetto a se si trovasse nella sala raggi.

- Il Trauma Leader deve essere nelle condizioni di annotare la destinazione del paziente una volta che il trauma è terminato, ed il sistema deve automaticamente archiviare la documentazione prodotta, allegati compresi, sul server dell'ospedale. Siccome la copertura WiFi dell'ospedale non è presente in ogni area dell'edificio, oppure la rete potrebbe essere momentaneamente non disponibile per qualsiasi motivo, il sistema deve prevedere la possibilità di ritrasmissione dei dati. In altre parole bisogna far si che sia possibile inviare il report, ed i media relativi ad un trauma, anche in un momento successivo.

- Il sistema deve fornire una piattaforma web per accedere, gestire, stampare ed esportare i report prodotti ed archiviati. All'interno della piattaforma deve essere presente l'idea di gerarchia e filtraggio degli utenti in base al proprio ruolo, per gestire al meglio l'accesso ai dati.

In ultima analisi il sistema deve garantire la tolleranza ai guasti, soprattutto per il discorso sopracitato che la copertura di rete potrebbe non essere sempre presente, ed avere ampia adattabilità dovuta dalla dinamicità del dominio applicativo.

### 1.2.2 Architettura di sistema

Per quanto concerne la parte architetturale, il progetto Trauma Tracker è stato ideato in maniera modulare, per garantirne l'estensibilità ed agevolare l'inserimento di nuove funzionalità, creando diversi livelli di supporto ed utilizzo.

- *Basic Level* - Livello base dell'applicazione. Vengono attuate una minima parte delle funzionalità, infatti per esempio è possibile tracciare gli eventi basilari senza la possibilità di riconoscere le stanze, oppure interagire col sistema solo agendo direttamente sul device fisico. A questo livello di supporto le uniche informazioni che vengono aggiunte automaticamente alle note generate sono quelle riguardanti data ed ora. Inoltre il sistema da la possibilità di specificare solamente l'identità del Trauma Leader. Tutte le restanti informazioni possono essere aggiunte manualmente ad ogni procedura.

- *Basic Plus Level* - A questo livello di supporto l'applicazione offre l'inserimento automatico dell'identificazione della stanza in cui è stato generato l'evento, ed il recupero delle informazioni riguardanti i parametri vitali del paziente.

- *Intermediate Level* - Il livello intermedio dell'applicazione garantisce al Trauma Leader la possibilità di usufruire di display ausiliari sui quali ricevere le informazioni dal sistema. Questi display possono essere di diversa natura e genere, si può pensare sia a normali display, sia a smart glasses o altri dispositivi wearable. L'utilizzo di questi ultimi serve più che altro a supporto dell'idea di permettere ai medici di ottenere informazioni senza distogliere lo sguardo dall'attività che stanno svolgendo.

- *Advance Level* - È il livello più alto dell'applicazione ed introduce nel sistema un supporto completamente hands-free, rendendo possibile la creazione di note usando la voce. A tale scopo sono state implementate nel sistema tecniche di riconoscimento vocale ed algoritmi specifici, insieme alla creazione di vocabolari adhoc per una migliore precisione.

In figura 1.1 vengono mostrati nel dettaglio i diversi livelli appena esplicitati.

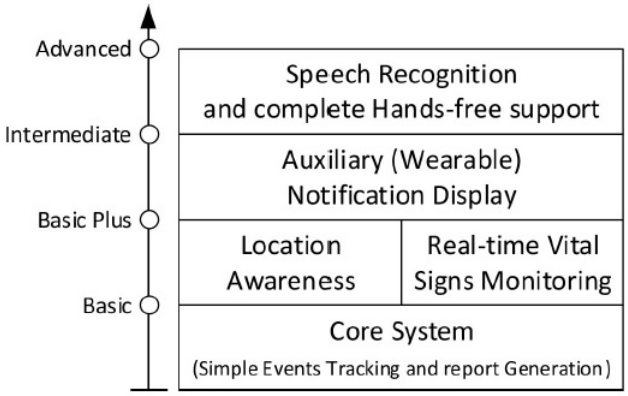


Figura 1.1: Livelli di utilizzo di Trauma Tracker.

Per quanto riguarda la struttura generale del sistema si possono individuare due porzioni principali:

- *sottosistema del Trauma Leader (Trauma Assistant Agent)* - con il compito di supportare il lavoro dei medici, e quindi pensata per essere eseguita su device mobile;
- *sottosistema Reports Dashboard* - con il compito di archiviare e gestire i report munita di un'interfaccia web per l'interazione con essa.

Data la natura distribuita dell'applicazione ed affinché venga garantita la modularità e l'estensione funzionale, a livello di progettazione è stata introdotta una terza parte del sistema (infrastruttura $GT^2$), pensata per fornire i servizi infrastrutturali a supporto delle varie funzionalità dello stesso, tra i quali troviamo la capacità di fornire e nascondere gli aspetti di interazione tra i device e macchinari specifici all'interno del sistema. In figura 1.2 viene fornita una visione generale sull'architettura logica di Trauma Tracker.

Il sottosistema contrassegnato in figura dalla lettera A e B è quello del Trauma Leader, la porzione di sistema più critica sia per il contesto in cui viene utilizzata, sia per la reattività necessaria. Come già annunciato, il dominio sulla quale si opera è estremamente complicato a livello manageriale e quindi è necessario che il sistema abbia alte prestazioni in termini di efficienza ed efficacia nella gestione dei dati e nel compimento delle proprie funzioni, ed una estrema semplicità di utilizzo in termini di interazione

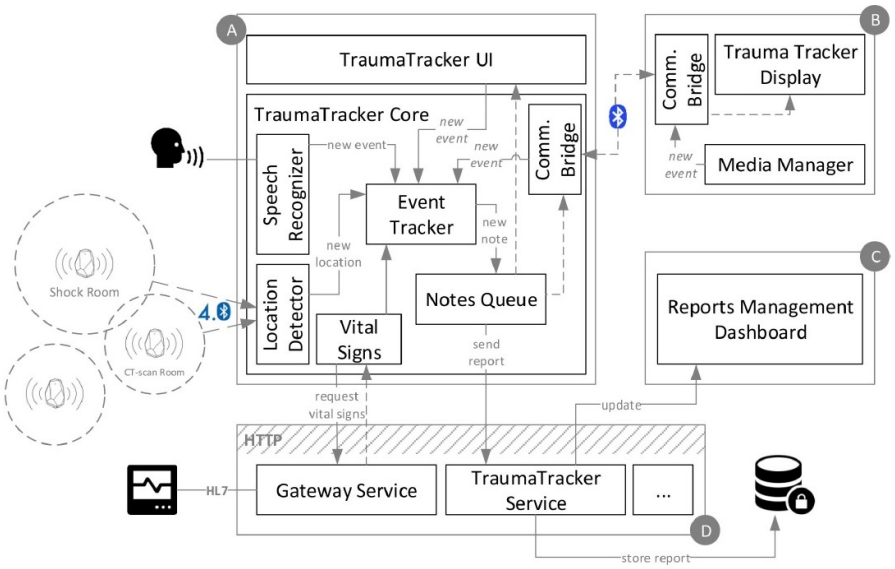


Figura 1.2: Le parti contrassegnate con la lettera A e B indicano il Trauma Assistant Agent, la lettera C indica il sottosistema Reports Dashboard mentre la lettera D indica l'infrastruttura GT[2].

uomo-macchina. Il nucleo del sistema può essere identificato dai componenti EventTracker e NotesQueue. Il primo ha il compito di ricevere tutti gli eventi relativi al trauma, creare le rispettive note per ogni evento, arricchirle con le informazioni aggiuntive richieste (data, ora, luogo in termini di stanza), e salvarle nel NotesQueue. Al termine del trauma EventTracker ha inoltre il compito di redigere ed inviare il report completo. Un altro compito di questa sottoparte del sistema è la funzione di monitoraggio dei parametri vitali del paziente. Nel dettaglio questi vengono periodicamente monitorati e registrati in tempi diversi dettati dalla specifica stanza in cui si trova il paziente in quel momento. Il periodo di monitoraggio e lo stato di salute del paziente vengono gestiti in modo direttamente proporzionale. Ci sono inoltre alcune procedure, e la somministrazione di alcuni farmaci, per le quali è importante sapere i parametri vitali del paziente, nel momento in cui, a fine trauma, si analizzano i report per capire se queste procedure hanno avuto effetto o meno sullo stato di salute. Questa sottoparte di sistema ha in carico anche tutta la gestione della parte multimediale. Trauma Tracker è stato infatti progettato per permettere l'acquisizione di file

multimediali in maniera molto rapida. Le note vocali sono state pensate, di base, come reminder per il Trauma Leader. Vengono infatti utilizzate quando il medico vuole ricordare un qualcosa di specifico che non è stato contemplato esplicitamente dal sistema, mentre le foto ed i video servono per dare maggiore concretezza alla documentazione relativa ad un trauma. Nell' Advanced Level del sistema si possono utilizzare i media vocali anche per registrare eventi senza dover necessariamente utilizzare l'interfaccia di input standard del device, grazie a meccanismi di riconoscimento vocale.

Il sottosistema contrassegnato in figura dalla lettera C è il sottosistema Reports Dashboard ed è la parte del sistema con il compito di analizzare e gestire tutti i report collezionati. È composta da una applicazione web-based che permette la navigazione e la consultazione dello storico dei vari report. Tutta la documentazione viene ricevuta direttamente dal sottosistema del Trauma Leader tramite i servizi offerti dall'infrastruttura $GT^2$. In realtà è quest'ultimo sottosistema che gestisce l'archiviazione dei report e li rende disponibili al sottosistema Reports Dashboard. In sostanza è una tipica applicazione di data management che permette la gestione degli accessi ai dati memorizzati. Ci sono liste di utenti accreditati con diversi livelli di accesso.

Sostanzialmente l'infrastruttura $GT^2$ funge da mezzo di comunicazione ed interazione tra tutte le sottoparti del sistema, ed anche tra il sistema stesso ed eventuali device di terze parti esistenti all'interno della struttura ospedaliera. L'infrastruttura fornisce infatti il servizio di misurazione dei parametri vitali del paziente all'intero sistema, fungendo da tramite tra i componenti del sistema ed i device esterni che eseguono la misurazione vera e propria dei parametri. Tra i diversi servizi di comunicazione offerti troviamo TraumaTracker Service e Gateway Service. Il primo fa da collante tra il sottosistema del Trauma Leader ed il server dove vengono immagazzinati i report, mentre il secondo fa da tramite tra il Trauma Assistant Agent ed i macchinari che tracciano i parametri vitali. Tutti i collegamenti forniti da questa infrastruttura utilizzano il protocollo HTTP utilizzando lo stile architetturale REST per i web service. Questa peculiarità garantisce l'accesso al sistema tramite l'infrastruttura di rete interna dell'ospedale. Entrando nello specifico dei servizi sopracitati osserviamo che TraumaTracker Service fornisce sia un'interfaccia per ricevere i report completi che vengono inviati poi al server collocato nel sottosistema Reports Dashboard, sia la possibilità da parte del Trauma Leader di richiedere la lista degli utenti, con i rela-

tivi ruoli, abilitati all'accesso dei dati. Il Gateway Service invece è stato pensato per la comunicazione del sistema con terze parti, in questo caso con il sistema di monitoraggio dei parametri vitali in tempo reale, utilizzando il protocollo HL7 [10] (uno standard utilizzato in campo medico per lo scambio, integrazione, condivisione e recupero dei dati elettronici relativi alla salute dei pazienti). Le modalità di funzionamento di questo servizio sono la request/response, utilizzata per ottenere i valori attuali, oppure la modalità push per agire non appena i dati siano disponibili.

### 1.2.3 Quadro tecnologico e BDI come modello di riferimento

Dal punto di vista tecnologico [33], nel prototipo sviluppato, il sottosistema del Trauma Leader è stato implementato come applicazione mobile per device Android, mentre il sottosistema Reports Dashboard è stato implementato tramite un'applicazione web-based. Il sottosistema $GT^2$ è stato implementato in Java utilizzando il framework Vert.x [25], mentre per la parte di archiviazione è stato utilizzato un database NoSQL, nello specifico MongoDB5 [16]. Per la parte riguardante i dispositivi wearable sono stati utilizzati gli smart glasses Vuzix7 m100 [26] collegati via Bluetooth tramite un canale TCP dedicato. Questi occhiali vengono utilizzati come un monitor ausiliario a supporto del Trauma Leader, e sono dotati di fotocamera che permette l'acquisizione di dati multimediali. Per la parte riguardante le tecniche ti riconoscimento vocale sono state utilizzati i componenti dedicati di Android [3], i quali sfruttano tecniche di riconoscimento basate su dizionari.

Essendo l'obiettivo principale di Trauma Tracker[32] il supporto dei medici, in particolare del Trauma Leader, nella produzione di una documentazione accurata semplificando il lavoro e minimizzando il più possibile lo sforzo, è chiaro che i compiti principali del sistema sono:

- tracciare gli eventi provenienti dall'ambiente circostante relativi allo specifico paziente;

- inferire il maggior numero possibile di informazioni dal contesto;

- produrre ed inviare il report al server, una volta che il trauma è terminato.

Questi task, soprattutto per quanto riguarda il tracciamento degli eventi, sono attività pressoché reattive e si è pensato quindi che, dal punto di vista implementativo, l'utilizzo di un paradigma ad agenti sarebbe stato un'ottima scelta per il caso di studio. Il Trauma Assistant Agent è stato ideato basandosi sul modello BDI (Belief Desire Intention), ed al suo interno è possibile trovare l'agente Trauma Assistant Agent. Grazie al modello BDI si possono creare sistemi che integrano il comportamento goal-oriented e quello reattivo, infatti gli agenti hanno goal espliciti da soddisfare tramite l'esecuzione di una selezione di piani, pur essendo sempre reattivi agli eventi che avvengono nell'environment che abitano. Questo modello architetturale semplifica la modellazione del Personal Assistant Agent con tutte le peculiarità fino ad ora esplorate. Le capacità reattive dell'agente vengono modellate sotto forma di piani, innescati da eventi che si verificano nell'ambiente. Nello specifico di Trauma Tracker tali eventi riguardano le manovre effettuate dal team di medici sul paziente, la registrazione di note, opzionalmente correlate a contenuti multimediali, o cambiamenti riguardanti i parametri vitali del paziente, il cambio di stanza dove si svolge il trauma o l'organizzazione del team. In accordo col modello concettuale Agent & Artifact (A & A) gli agenti, per raggiungere i propri obiettivi, utilizzano gli artefatti presenti nell'ambiente, ovvero entità passive che possono essere usate per diversi scopi in base alla loro natura. In particolare nel sistema Trauma Tracker sono presenti diversi artefatti ognuno creato ad uno scopo diverso. Per esempio ci sono artefatti per la gestione dell'interfaccia grafica (`StarterUi`, `MainUi`), uno per la gestione dei messaggi da mostrare come output (`ExternalDisplay`), un artefatto per la gestione della posizione del Trauma Leader (`PhysicalContext`), uno per gestire i parametri vitali del paziente (`VitalSignsMonitor`), un altro con il compito di immagazzinare costantemente le note prodotte e conseguentemente spedirle al server (`NotesStream`), e così via. Ognuno di essi fornisce un'interfaccia per mostrare e rendere osservabile una serie di proprietà del sistema al quale fa riferimento in base alla loro funzione.

Il sottosistema del Trauma Leader è stato implementato tramite JaCa-Andoid, ovvero una piattaforma che porta la programmazione multi-agente su dispositivi basati su Google Android. JaCa-Android[41] può essere visto come un porting parziale di JaCaMo [13], una piattaforma che include al suo

interno Jason[29] (linguaggio di programmazione per agenti), CArtAgO[4] (framework per programmare l'environment) e MOISE[15] (framework per la gestione dell'organizzazione del Multi-Agent System)

## 1.3 Un sistema di generazione di Warning

Come evidenziato precedentemente, Trauma Tracker è un sistema pensato per documentare ed effettuare il tracciamento automatico di un trauma [33]. Il prossimo passo è l'estensione dell'applicazione mediante un primo livello di assistenza, ovvero si è pensato di integrare il sistema attuale con un agente BDI in grado di assistere i medici durante il trauma, avvertendoli mediante l'utilizzo di Warning, in caso di situazioni critiche per i pazienti. Tali Warning sono messaggi di allarme che vengono visualizzati su schermo o dispositivi wearable.

Un ulteriore obiettivo della tesi è quello di analizzare nel dettaglio l'architettura BDI, comprenderne a pieno le potenzialità e constatare che sia realmente una valida soluzione per questo tipo di problema.

La generazione dei Warning, intesi come messaggi informativi a volte accompagnati anche da segnali acustici, nel campo medico ha il compito di aiutare i dottori riducendo il rischio di insuccesso delle loro pratiche, aumentando di conseguenza la soddisfazione dei pazienti. Con lo stesso dominio applicativo ci sono numerosi progetti di ricerca il cui lavoro si concentra sul monitoraggio in tempo reale dei parametri vitali del paziente o dell'ambiente circostante per rilevare in anticipo dei sintomi specifici o un declino generico della salute del paziente per poi conseguentemente generare allarmi automatici. Soluzioni di questo tipo sono di solito utilizzate a supporto delle persone anziane, con patologie croniche, per poterle controllare anche mentre sono nella propria casa.

Durante il trattamento di un trauma i medici hanno diversi parametri da controllare, come la risposta di alcuni parametri vitali all'assunzione di determinati farmaci o procedure, oppure tenere traccia dell'andamento del tempo per la gestione di alcune procedure. Per questo motivo, e per la freneticità e criticità del dominio applicativo, gli allarmi ed i Warning risultano un aiuto più che significativo.

L'obiettivo è quello di creare un sistema in grado di eseguire ragionamenti automatici sui parametri del paziente e sull'ambiente in cui opera,

raccogliendo informazioni in tempo reale sul trauma, riducendo così gli errori umani e migliorando il risultato del trauma stesso. La generazione dei Warning sarà una cosa dinamica, ovvero si vuole fare analisi in tempo reale dei dati collezionati, alcuni dei quali scaturiranno la generazione degli allarmi.

# Capitolo 2

# Tecnologie ad agenti e modello BDI: una panoramica

In questo capitolo si esaminano le tecnologie e le conoscenze pregresse emerse dall'analisi del caso di studio.

## 2.1 Agenti e sistemi multi-agente

Un agente è per definizione un'entità intelligente in grado di percepire l'ambiente che lo circonda e di reagire eseguendo azioni in seguito ad eventi[36]. Questa entità vive all'interno di uno spazio, chiamato environment (ambiente), ed ha la capacità di captare ciò che accade intorno a lui tramite dei sensori ed agire di conseguenza, modificando lo spazio, attraverso degli attuatori. In figura 2.1 viene mostrata una rappresentazione concettuale dell'agente e dell'ambiente.

Un environment può essere popolato da più agenti, ed infatti nella realtà raramente si incontrano sistemi con un unico agente. Questi sistemi dove si trovano un insieme di agenti che interagiscono tra loro mediante una opportuna organizzazione prendono il nome di sistemi multi-agente o sistemi ad agenti multipli[42]. Oggetto di ricerche da lunga data in intelligenza artificiale e non solo, visto che costituiscono un'interessante tipologia di modellazione di società, ed hanno quindi vasti campi d'applicazione, che si estendono fino alle scienze umane e sociali (economia, sociologia, etc.).
In figura 2.2 viene mostrata la struttura tipica di un sistema multi-agente[29].

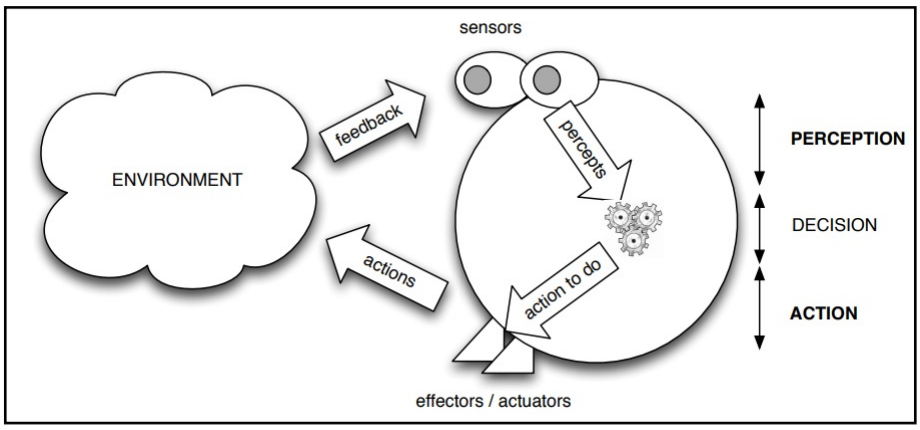


Figura 2.1: Un agente ed il suo rapporto con l'ambiente.

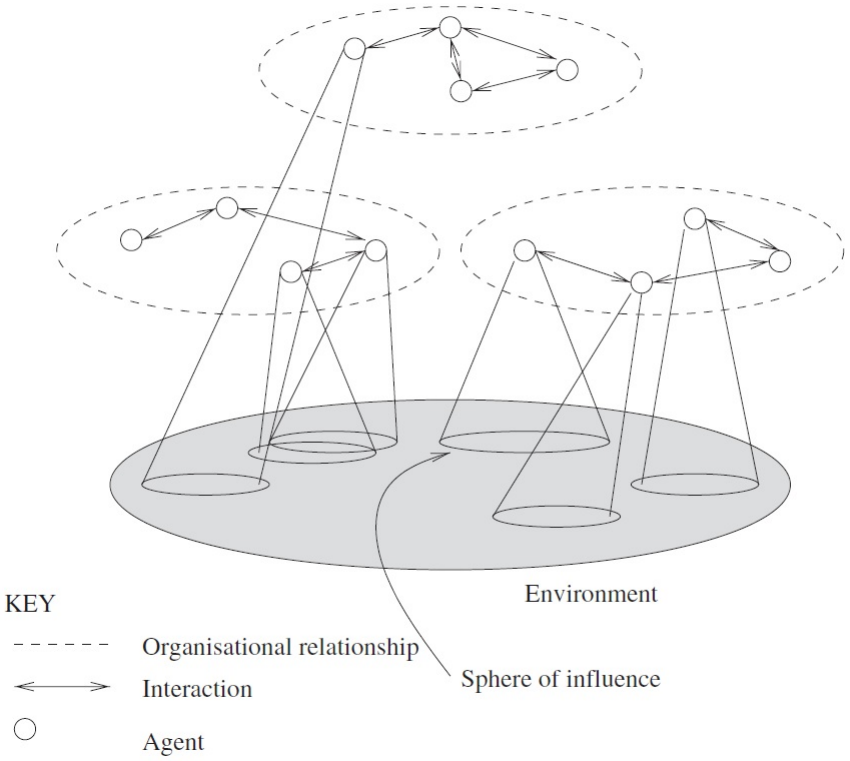


Figura 2.2: Struttura tipica di un sistema multi-agente

L'environment, rappresentato dallo spazio grigio sul fondo dell'immagine, è occupato e condiviso dagli agenti. I cerchi all'interno di questo spazio rappresentano l'ulteriore suddivisione dell'ambiente, ovvero sono le aree su cui il rispettivo agente può agire. Questi sottoinsiemi di environment possono essere visti come "sfere di influenza" degli agenti, perchè infatti rappresentano la parte di ambiente che l'agente è in grado di controllare. Esistono configurazioni nel quale due o più sfere di influenza si sovrappongono parzialmente, ciò implica che l'agente con tale sfera di influenza non ha il pieno controllo su di essa, in quanto una parte può essere modificata anche dall'agente che possiede la sfera di influenza che si va ad intersecare con la propria. Questa situazione rende la vita degli agenti più complicata, in quanto per ottenere un risultato nell'ambiente che l'agente desidera, dovrà tener conto di come possono comportarsi gli altri agenti aventi il possesso della stessa porzione di ambiente. I sottoinsiemi tratteggiati, situati nella parte alta dell'immagine, indicano dei raggruppamenti di agenti che vengono fatti in base a certi criteri di rapporto. È infatti possibile istituire delle vere e proprie organizzazioni e gerarchie tra gli agenti, con caratteristiche diverse in base all'applicazione. In base alle esigenze di sistema, infatti, gli agenti all'interno della gerarchia possono essere a conoscenza l'uno dell'altro, oppure possono essere all'oscuro più totale tra di loro.

Usando questo paradigma di programmazione si possono avere diversi agenti che cooperano tra loro, ognuno dei quali appartiene ad un ambiente, ma possono esistere allo stesso tempo più environment ed un unico agente può lavorare anche su ambienti differenti. All'interno dell'environment si trovano anche gli artefatti, entità passive di supporto al lavoro degli agenti. Pensiamo ad oggetti di uso quotidiano che noi utilizziamo per raggiungere i nostri obiettivi, gli artefatti analogamente possono essere logicamente visti come tali, entità passive utilizzate dagli agenti per raggiungere i loro goal.

Visto la rilevanza di questo modello e la sua grande adattabilità a diversi campi di applicazione, bisogna considerare che il sistema ed i relativi elementi possano essere sia fisici che virtuali. Per esempio se consideriamo un sistema di robot realmente esistenti che agiscono nello spazio, l'environment e gli agenti sono chiaramente entità fisiche, mentre nel caso di agenti software che abitano all'interno di un computer o di una rete, ci troviamo di fronte ad un ambiente e ad un sistema virtuale.

### 2.1.1 Principali caratteristiche degli agenti

Questi agenti, definiti intelligenti o cognitivi, rispecchiano le seguenti caratteristiche:

- *Autonomia* - capacità di operare in modo indipendente per soddisfare gli obbiettivi che sono gli sono stati delegati. I sistemi ad agenti sono formati da entità autonome perchè sono in grado di prendere decisioni indipendenti riguardo al metodo per raggiungere il proprio obbiettivo. I programmi di cui stiamo parlando sono in grado di delegare i goal tra gli agenti, i quali decidono come è meglio agire per raggiungere lo scopo che gli è stato assegnato. Il metodo usato dagli agenti per portare a termine il loro compito, è suddiviso tra piani i quali a loro volta possono essere suddivisi in eventuali sottopiani. Ogni agente è in grado di gestire i suoi piani e di combinarli in modo dinamico per raggiungere il proprio scopo. Le decisioni che vengono prese dagli agenti sono esclusivamente sotto il proprio controllo e non sono vincolate dalle altre entità;

- *Proattività* - capacità di avere un comportamento goals-directed, ovvero incentrato al raggiungimento dei propri obiettivi. Gli agenti, infatti, non agiscono in seguito ad indicazioni impartite, come invece fa un oggetto Java, il quale è essenzialmente passivo fino a che qualcuno non lo invoca. L'agente, una volta che gli viene delegato un compito, trova il modo per portarlo a termine gestendo da solo i suoi piani, senza che nessuno gli debba indicare la via da seguire.

- *Reattività* - capacità di agire in seguito al cambiamento dell'ambiente circostante. Nel caso in cui l'esecuzione di un piano non vada a buon fine, l'agente è in grado autonomamente di cercare una via alternativa per il conseguimento del suo obiettivo, e quindi di reagire al problema verificato. Un punto di forza di questo paradigma di programmazione è la possibilità di creare sistemi allo stesso tempo reattivi e goal-directed, cosa non semplice da raggiungere tramite l'utilizzo di altri paradigmi.

- *Abilità sociali* - capacità di cooperazione e collaborazione al fine di raggiungere un obiettivo. Con questa caratteristica si va oltre al semplice scambio di informazioni, ma si parla proprio di un puro lavoro

cooperativo tra gli agenti. Per concretizzare questa abilità è opportuno che gli agenti possano comunicare tra loro sia in termini di scambio di bytes ed invocazione di metodi, sia a livello di conoscenze, ovvero i propri belief, goal e plan.

### 2.1.2 Agent Oriented Programming come paradigma di programmazione per lo sviluppo di sistemi multi-agente

Per la programmazione di questi tipi di sistemi si utilizza l'Agent Oriented Programming (AOP), un paradigma di programmazione che può essere considerato "post-dichiarativo"[29] per i seguenti motivi:

- Nella programmazione procedurale si deve dire al sistema quali sono precisamente le azioni da svolgere tramite algoritmi dettagliati, ma per l'uomo risulta particolarmente impegnativo pensare al livello di dettaglio richiesto;
- In quella dichiarativa si cerca di spostare l'attenzione dagli aspetti di controllo, lasciando al meccanismo di controllo incorporato il compito di capire come raggiungere lo scopo richiesto. Per poter scrivere programmi efficienti in linguaggi dichiarativi (come Prolog) bisogna che il programmatore abbia una conoscenza profonda di come lavora il meccanismo built-in.

L'idea alla base dell'AOP è che il programmatore decida i goal, mentre al meccanismo di controllo rimane il compito di decidere come raggiungere l'obbiettivo imposto. In questo paradigma, il meccanismo di controllo integrato implementa un modello razionale corrispondente alla nostra comprensione intuitiva di esprimerci tramite credenze e desideri(livello di astrazione alto), garantendo un utilizzo minore di conoscenze specifiche.

## 2.2 Modello BDI

Tra i vari approcci che si possono utilizzare per la programmazione di sistemi Agent based si è riposta maggiore attenzione sul modello Belief-Desire-Intention (BDI), ideato a metà degli anni 80 da un'insieme di filosofi all'interno del progetto Rational Agency che prese luogo alla Stanford

University[29]. All'origine di questo modello si trova la teoria sul "ragionamento pratico umano" del filoso Michael Bratman[22], il quale trasse ispirazione dal modello di comportamento umano ed incentrò la sua attenzione sul ruolo delle intenzioni nel ragionamento pratico. L'idea principale su cui si basa tutta la modellazione è quella di poter parlare di programmi per computer come se fossero stati mentali. Sostanzialmente, quando parliamo di un sistema di Belief-Desire-Intention[29], stiamo parlando di programmi per computer con elementi computazionali analoghi a credenze, desideri ed intenzioni.

Quando si parla di Agent Oriented Programming e sistemi multi-agente, si parla di paradigmi computazionali che permettono la creazione di sistemi software complessi composti da un set aperto di entità autonome chiamate agenti che lavorano ed interagiscono all'interno di un environment comune. Seguendo questa direzione sono stati creati diversi modelli computazioni ed architetture ad agenti, tra cui l'esempio più eclatante ed apprezzato è appunto il modello BDI. Di seguito verrà fatta una distinzione approssimativa delle tre entità che compongono il modello per cercare di capire maggiormente la loro essenza.

- *Beliefs* - Informazioni che l'agente possiede riguardo al mondo esterno. Queste informazioni possono essere inconsistenti, inaccurate o non aggiornate.

- *Desires* - Tutto ciò che l'agente può desiderare di fare, praticamente avere un desiderio. Questa situazione, d'altro canto, non implica che l'agente soddisfi il suo desiderio, ma sicuramente ne influenza il comportamento.

- *Intentions* - Ciò che l'agente ha deciso di fare, quello per cui sta lavorando. Le intenzioni possono essere dei compiti delegati all'agente, oppure possono giungere da un ragionamento, per esempio se l'agente aveva delle opzioni e decide di scegliere questa.

Con il termine "sistema intenzionale" ci si riferisce, in letteratura filosofica, a quei comportamenti che possono essere predetti e spiegati in termini di attitudini tramite credenze, desideri ed intenzioni. Questo approccio si basa su ragionamenti effettuati sulla vita di tutti i giorni, sul mondo quotidiano. Infatti viene naturale utilizzare la "psicologia popolare" per spiegare e comprendere il comportamento dei sistemi intelligenti complessi, cioè le

persone. Se per esempio, si utilizzano frasi del tipo "Michael ha intenzione di scrivere un paper" per spiegare il suo comportamento, ci aspettiamo di trovare Michael fare un piano per dare priorità alla scrittura del paper accantonando altri impegni. Ci aspettiamo, per esempio, di vederlo passare molte ore davanti al suo computer ed allo stesso tempo saremmo sorpresi di vederlo ad una festa durante la notte. Questo meccanismo di comprensione del sistema tramite l'attribuzione di atteggiamenti come credenze e desideri è semplicemente un sistema di astrazione. Un metodo conveniente per parlare di sistemi complessi che ci permette di predirre e spiegare in modo conciso il loro comportamento senza dover comprendere come effettivamente lavorano. Per quanto riguarda il mondo dell'informatica, si è alla costante ricerca di buone tecniche di astrazione che permettano di gestire la complessità con maggiore facilità e si è quindi pensato di utilizzare questa metodologia appena spiegata come strumento di astrazione per spiegare, capire e programmare sistemi complessi di computer come possono essere i sistemi multi-agente.

All'interno della comunità ATAL (International Workshop on Agent Theories, Architectures, and Languages)[43], il modello BDI divenne presto il modello più conosciuto ed approfonditamente studiato. Le ragioni per il suo grande successo sono numerose, ma probabilmente la più convincete è la sua capacità di unire il modello filosofico di ragionamento pratico umano con un alto numero di implementazioni (sistemi basati sull'architettura IRMA[30] o diversi sistemi PRS-like già esistenti[34]), numerosi sistemi funzionanti ed utili applicazioni (tra cui il famoso sistema di diagnosi di problemi per lo space shuttle Discovery della NASA, oppure sistemi di controllo dei processi di fabbrica e gestione dei processi aziendali) ed ultimo, ma non meno importante, una elegante astrazione della logica semantica ampiamente rivisitata e studiata all'interno della comunità di ricerca che opera sugli agenti.

### 2.2.1 Funzionamento di un agente BDI

Detto questo sorge spontanea la domanda: come fa un agente a passare dalle sue credenze, desideri ed intenzioni a decidere come comportarsi e come agire? Utilizzando il modello che prende il nome di "Practical Reasoning", ovvero il ragionamento pratico. Consiste nel valutare considerazioni contrastanti a favore e contro le opzioni che si presentano, dove le considerazioni

pertinenti vengono fornite da ciò che l'agente desidera, valuta e crede. Questa forma di ragionamento esiste anche per gli umani e consiste in due ben distinte attività:

- *Deliberazione* - quel momento in cui la volontà considera, soppesa e subisce i vari moventi, prima di determinarsi. In altre parole la ponderazione che precede la decisione;

- *Means-ends reasoning* - il momento in cui si decide la strategia comportamentale per il raggiungimento del nostro scopo.

Nel modello BDI la fase di deliberazione si rispecchia nell'utilizzo delle intenzioni. Essendo queste delle pro-attitudini, è loro intento quello di condurre ad azioni. Bratman stesso notò che le intenzioni ricoprono un ruolo molto importante nell'influenza delle azioni rispetto ad altre pro-attitudini, come per esempio il puro desiderio. Inoltre le intenzioni sono persistenti fino a che non le si soddisfa, infatti se si adotta un'intenzione si è inclini al dedicarsi anima e corpo al suo perseguimento. Se si abbandona immediatamente l'intenzione senza dedicare particolari risorse al suo perseguimento, è lecito dire che in realtà quella non fosse mai stata un'intenzione.

La fase di Means-ends reasoning invece è il processo che, nel campo dell'intelligenza artificiale, viene chiamato "planning" (pianificazione). Un sistema che si occupa della pianificazione prende in input l'obiettivo che l'agente vuole raggiungere, le sue attuali convinzioni e le azioni disponibili, generando come output il piano, ovvero un corso di azioni. Se il piano risulta essere formato correttamente allora l'agente è in grado, seguendolo, di raggiungere il proprio obiettivo e quindi di adempire il proprio lavoro. Sostanzialmente l'agente, eseguendo completamente il piano, passa da un mondo descritto tramite le sue credenze, ad un mondo in cui il suo obiettivo e le sue intenzioni sono raggiunte e soddisfatte.

L'agente, per passare dall'input ottenuto dai sensori all'azione di output da eseguire, lavora manipolando i propri piani[29]. Il contenuto informativo di un agente è composto da una belief base (insieme di credenze) e compiti da svolgere (goal). Le azioni all'interno dei piani, possono essere triggerate dall'avvenimento di un certo evento che può essere scaturito o dall'ambiente o dall'utente. A questo proposito viene mostrato in figura 2.3 il "reasoning cycle" di un generico agente BDI in versione semplificata, in quanto non viene spiegato il meccanismo di scambio di messaggi tra agenti.

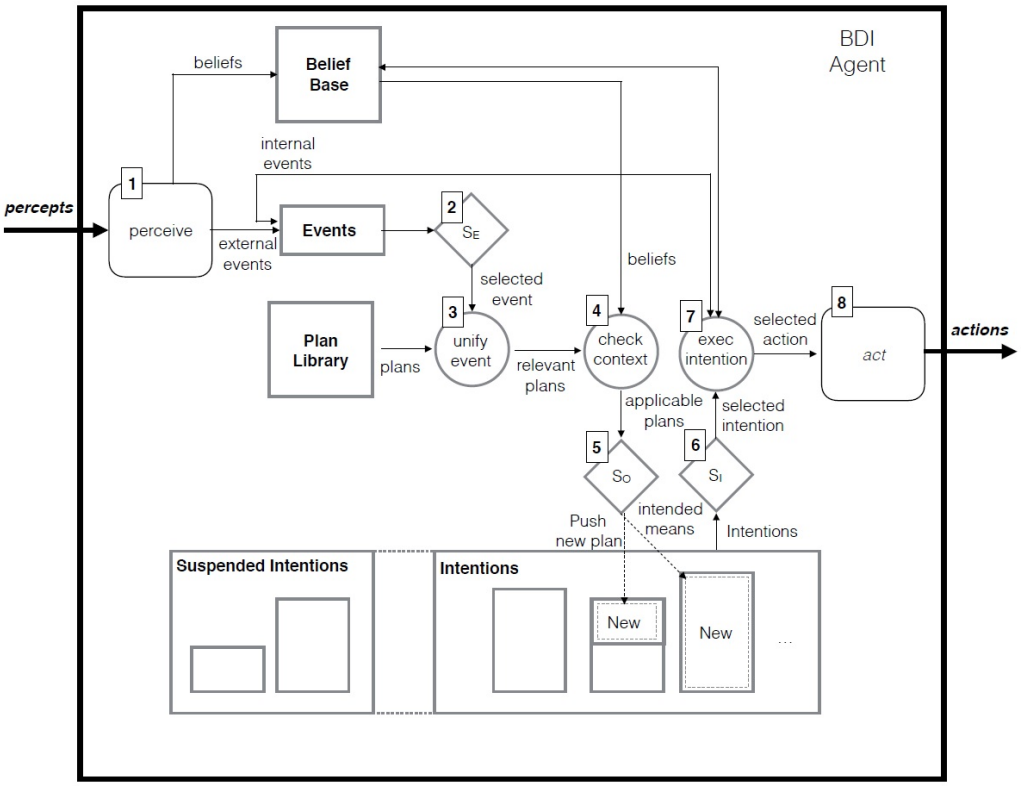


Figura 2.3: Rappresentazione grafica del funzionamento di un agente in accordo col modello BDI.

I numeri all'interno dell'immagine indicano i diversi stage in cui viene suddiviso il comportamento dell'agente. Il primo stage è quello della percezione in cui l'agente capta gli eventi dall'ambiente, aggiorna la propria belief base in accordo con eventuali cambiamenti scaturiti dagli eventi percepiti, i quali possono includere anche nuovi goal o fallimenti. Dopodiché l'agente decide l'azione da attivare prima di tutto recuperando un evento dalla coda (step 2), se disponibile, e seleziona un piano applicabile dato lo stato corrente della belief base (step 3 e 4). A differenza dell'evento, il nuovo piano viene posizionato in cima allo stack delle intenzioni o come sub-goal da raggiungere, oppure come nuova intenzione (step 5). Dato l'insieme di intenzioni in esecuzione ne viene selezionata una da portare avanti (step 6) per poi eseguirla (step 7) recuperando la prossima azione da svolgere seguendo il piano. Infine si esegue l'azione selezionata (step 8) coinvolgendo interazioni con l'ambiente o cambiamenti interni, come l'inserimento di un nuovo belief o la sospensione di un'intenzione. Questo insieme di azioni viene ripetuto continuamente dall'agente formando un loop.

## 2.3 Personal Assistant Agent

Come anticipato dell'introduzione, nel mondo odierno si assiste ad una continua evoluzione dal punto di vista tecnologico e ci si spinge verso una visione sempre più cooperativa del computer, ovvero il device non viene più visto come elemento passivo da utilizzare, ma vero e proprio collaboratore attivo, intelligente e personalizzato[37]. Seguendo il corso degli eventi iniziano a comparire sul mercato dispositivi incentrati sul supporto dell'utente, applicazioni in grado di fornire aiuti concreti alle persone. Questi sistemi prendono il nome di Personal Assistant Agent [44] e sono sostanzialmente esempi di applicazioni dell'Agent Oriented Programming dal dominio applicativo estremamente vasto ed eterogeneo. Il livello di supporto garantito da questi sistemi può differire al variare delle esigenze, viene sempre considerato però come obiettivo principale l'incremento delle performance e la diminuzione del carico di lavoro e lo sforzo cognitivo che l'utente deve impiegare nello svolgimento del proprio lavoro. Nello specifico i diversi livelli di intervento di un agente di questo tipo, indipendentemente dal dominio applicativo, possono essere:

- aiutare direttamente l'utente eseguendo compiti al suo posto;

- collaborare direttamente con l'utente supportandolo nello svolgimento del compito (possono addestrare l'utente o insegnargli cose);

- collaborare indirettamente con l'utente fornendo informazioni aggiuntive utili allo svolgimento del compito (suggerimenti, promemoria, informazioni triangolate, informazioni sul contesto del dominio).

Il concetto di Personal Assistan Agent e di agente, soprattutto se accostato all'aggettivo intelligente, può destare sospetti e causare incertezza nella cognizione delle persone[38]. La sua caratteristica di agire autonomamente senza particolare controllo può portare del giustificato timore nell'utente. La concezione di macchina intelligente fa parte da molti anni del bagaglio culturale dell'utente medio, ma questi nuovi sistemi portano un'innovazione notevole nell'iterazione uomo-macchina. Si parla infatti di device in grado di interagire con gli umani in maniera naturale, prendere decisioni, imparare ed evolversi autonomamente cooperando con l'utente. Utilizzando tecniche di Intelligenza Artificiale [37] hanno la capacità di acquisire informazioni ed imparare sia dall'utente che dagli altri agenti e macchine presenti nell'ambiente circostante. Viene cambiata radicalmente la user experience, infatti per quanto riguarda l'interazione, non serve necessariamente che la comunicazione inizi lato utente impartendo comandi diretti all'agente tramite interfaccia apposita, ma vi è una collaborazione cooperativa tra l'utente e gli agenti che comunicano e collaborano per il raggiungimento dell'obiettivo comune. Un'altra peculiarità di questo tipo di sistemi è la loro predisposizione all'apprendimento, infatti l'agente diventa gradualmente sempre più efficiente grazie alla sua attitudine ad imparare dagli interessi dell'utente, dalle sue abitudini e preferenze.

L'utilità e l'importanza dei Personal Assistan Agent è stata inoltre rafforzata da studi etnografici compiuti sulle abitudini lavorative dell'uomo[44] i quali hanno evidenziato il fatto che le persone tendenzialmente portano a termine tutti i compiti importanti, sono brave nel multitasking e riescono a ricordare bene tutto ciò che conta, però peccano nel raggiungimento di compiti senza scadenze prestabilite o nel ricordare i dettagli meno salienti. In altre parole se le persone sono sovraccariche di lavori ed impegni tendono a dimenticare e trascurare i compiti meno importanti concentrandosi sugli altri. Questo è il caso ideale in cui un Personal Assistant Agent diventa realmente efficiente ed efficace.

Per avere un aiuto continuativo e pervasivo sarebbe ideale portare questi sistemi su dispositivi mobile e wearable. Grazie all'eccezionale sviluppo di queste tecnologie[33] nell'ultimo decennio è ora possibile avere un Personal Assistan Agent sempre con te pronto ad aiutarti. I dispositivi mobili ormai sono un must have per ogni persona, e grazie alla potenza di calcolo che si è raggiunta in questo campo (paragonabile a quella dei computer) e della grande varietà di sensori di cui sono dotati (fotocamera, GPS, NFC, ecc) si è giunti ad una notevole diffusione di applicazioni per smartphone e tablet con l'obbiettivo di assistere l'utente nell'ottica dell'Assistant Agent. Anche nel campo delle tecnologie wearable, si è raggiunti una maturità tale da immaginare un utilizzo vero e proprio di questi dispositivi, e non solo come oggetto di studio e ricerca all'interno di laboratori. Questi diventano gli elementi fondamentali per la realizzazione di sistemi hands-free o use-on-the-go che cambiano ulteriormente il modo di interagire con gli agenti.

### 2.3.1 Stato dell'arte

Attualmente esistono diversi Personal Assistant Agent, molti dei quali utilizzabili tramite pc, ma nulla vieta la trasposizione di tali progetti su dispositivi mobile. Di seguito un elenco di agenti esistenti con i relativi domini applicativi[44] [37] [31].

- *CALO* - assistente personale per impiegati d'ufficio. L'idea del progetto è quella di affiancare un assistente digitale come se fosse un assistente amministrativo, in grado di facilitare i lavori di routine e supportare il processo decisionale dell'utente.
- *Electric Elves* - progetto con dominio applicativo simile a CALO, ovvero supportare un lavoratore d'ufficio impegnato. In aggiunta vengono incluse una serie di capacità proattive pensate per la gestione e lo scheduling personale di riunioni ed incontri. L'attenzione in questo progetto, al contrario di CALO, è focalizzata sull'organizzazione e gestione di attività condivise tra i vari colleghi di lavoro.
- *RADAR* - assistente con il compito di aiutare l'utente nella gestione delle email. Questo agente è in grado, analizzando i messaggi, di individuare task all'interno delle email ed offrire la possibilità di iniziare questi lavori al posto dell'utente.

- *ANTIPA* - assistente cognitivo progettato per gestire proattivamente le informazioni per conto di utenti sovraccaricati di lavoro. L'agente è in grado di integrare le informazioni raccolte per fornire all'utente, in tempo reale, informazioni a livello riconciliato pertinenti al suo scopo.

- *COLLAGEN* - assistente per la gestione di task cooperativi. Il sistema è in grado di aiutare le persone a collaborare per il raggiungimento di obbiettivi comuni.

- *STEAM* - assistente usato a supporto della risoluzione di problemi di coordinazione in team flessibili, quindi sempre in ambito lavorativo. È sostanzialmente un sistema di agenti per il monitoraggio e la ripianificazione di compiti condivisi e task riguardanti la comunicazione all'interno di team di lavoro.

- *STAPLE* - assistente utilizzato per la gestione della comunicazione con altri sistemi simili. L'agente è in grado di generare dialoghi con altri agenti con lo scopo di creare piani di collaborazione ed azione per la risoluzione di problemi di cooperazione e coordinamento.

- *Maxims* - assistente per la gestione delle email, stesso dominio applicativo di RADAR. L'agente ha il compito di monitorare continuamente l'utente mentre utilizza il programma di posta e nel frattempo memorizza tutte le coppie situazioni - azioni ampliando così la sua base di conoscenza. Così facendo l'agente riesce quindi a prevedere quale azione è appropriata alla situazione ed a determinare un livello di confidenza per ogni sua predizione. I concetti alla base di questo agente, sono stati utilizzati anche per l'implementazione di un *Meeting Scheduling Agent*, ovvero un assistente in grado di gestire l'organizzazione di incontri tra utenti, ognuno con le proprie priorità ed impegni. Il metodo di apprendimento di questi agenti permette di assistere gli utenti in maniera estremamente personalizzata automatizzando l'organizzazione in base alle proprie abitudini.

- *NewT* - assistente con il compito di filtrare le informazioni interessanti per l'utente all'interno di documenti. L'agente è in grado di scremare le informazioni all'interno di uno stream di dati alla ricerca di articoli interessanti per l'utente. Questo genere di sistema nasce dalla necessità di supporto nella selezione di articoli di interesse tra la

grande quantità di materiale diffuso quotidianamente tramite internet. All'interno del sistema possono esserci più agenti, che lavorano contemporaneamente, addestrati tramite esempi di articoli da considerare o da scartare. L'agente è in grado di fare una completa analisi del testo alla ricerca di parole chiave che lo aiutino a comprendere il contenuto, e di conseguenza valutarlo, rispetto alle esigenze dell'utente. Una volta che gli articoli vengono proposti all'utente, viene creata una classificazione in base a dei valori di feedback generati dall'utente, utili per il raffinamento delle ricerche successivo e quindi per l'apprendimento dell'agente stesso.

- *Ringo* - assistente impiegato nel campo dell'intrattenimento. L'agente ha il compito di consigliare canzoni in base ai gusti musicali dell'utente. Questo tipo di agente funziona grazie ad un sistema di filtraggio basato sul sociale, ovvero confronta gli utenti tra di loro, se vede interessi musicali simili, allora suppone che le canzoni ascoltate da uno possano piacere all'altro e viceversa. Quest'ultimo assistente lavora in maniera differente rispetto ai precedenti, in quanto non vengono confrontati gli interessi dell'utente con il contesto, ma vengono analizzate le correlazioni tra diversi utenti.

- *UbiMedic2* - framework multi-agente a supporto delle operazioni di soccorso. Ha il compito di supportare e gestire la comunicazione durante le emergenze territoriali. Il sistema è stato pensato per semplificare l'integrazione in un unico sistema di molteplici entità complesse, siano queste persone o dispositivi medici. Ha il compito di fornire una piattaforma di informazioni integrata, sulla quale possono essere distribuiti una serie di servizi intelligenti per l'assistenza sanitaria.

- *CASCOM* - sistema multi-agente distribuito con il compito di gestire le emergenze sanitarie in maniera intelligente. È in grado di fornire assistenza remota in maniera efficiente in caso di eventi inaspettati. Nel sistema possono essere distribuiti dati ed informazioni, in modo tale da renderle disponibili ai medici di tutto il mondo, consentendo così scelte più facili e veloci in casi di emergenza.

Esistono tanti altri lavoro in fase di sviluppo con l'obiettivo, per esempio, di progettare agenti in grado di guidare l'utente verso luoghi di interesse in

base alle proprie attitudini ed interessi, oppure sistemi nell'ambito healthcare in grado di aiutare persone disabili nella gestione della propria giornata. Quest'ultimo agente dovrebbe essere in grado di monitorare le azioni della persona, capire il suo comportamento ed interagire con lei in maniera appropriata fornendo informazioni sulla situazione e fornendole un aiuto concreto nella risoluzione di problemi. Un agente di questo tipo probabilmente è il più difficoltoso dal punto di vista progettuale, in quanto dovrebbe essere altamente personalizzabile ed agire diversamente in base alla disabilità dell'individuo che lo utilizza. Inoltre delle situazioni che per noi sono naturali e scontate, possono riscontrare invece dei problemi per persone disabili, quindi anche la fase di analisi richiede uno sforzo cognitivo non indifferente.

## 2.4 Linguaggi e framework per la creazione di sistemi multi-agente

Per programmare un sistema ad agenti esistono diversi linguaggi e framework a disposizione, ognuno dei quali deve rispecchiare le seguenti caratteristiche [29]:

- supporto della delegazione a livello di goal. Il framework che si utilizza deve permettere la delegazione dei compiti agli agenti senza dover necessariamente impartire la sequenza di statement da seguire per giungere al risultato desiderato. Il programmatore deve poter utilizzare un linguaggio ad alto livello per comunicare con gli agenti ed anche nella descrizione degli obiettivi.

- supporto della risoluzione di problemi in maniera goal-directed. Gli agenti del sistema devono avere la capacità di risolvere gli obiettivi a loro delegati cercando sistematicamente di soddisfarli.

- utilizzo di un linguaggio idoneo alla creazione di sistemi responsive all'environment.

- piena integrazione con il comportamento responsive e goal-directed.

- supporto della comunicazione e cooperazione a livello di conoscenza.

### 2.4.1 La piattaforma JaCaMo

Un framework molto utilizzato per la programmazione multi-agente su piattaforme desktop è JaCaMo[13], il quale combina tre diverse tecnologie:

- *Jason* - per programmare gli agenti autonomi;
- *CArtAgO* - per programmare l'ambiente e gli artefatti;
- *Moise* - per programmare l'organizzazione multi-agente.

Un sistema programmato in JaCaMo è in grado di coprire tutti i livelli di astrazione richiesti dallo sviluppo di sistemi multi-agente, grazie all'intreccio di queste tre tecnologie da tempo affermate nel settore. Per capire appieno le astrazioni del framework, basta guardare la rappresentazione in figura 2.4 dove vengono mostrate le tre piattaforme indipendenti con le rispettive interazioni. È necessario considerare fondamentale la definizione di

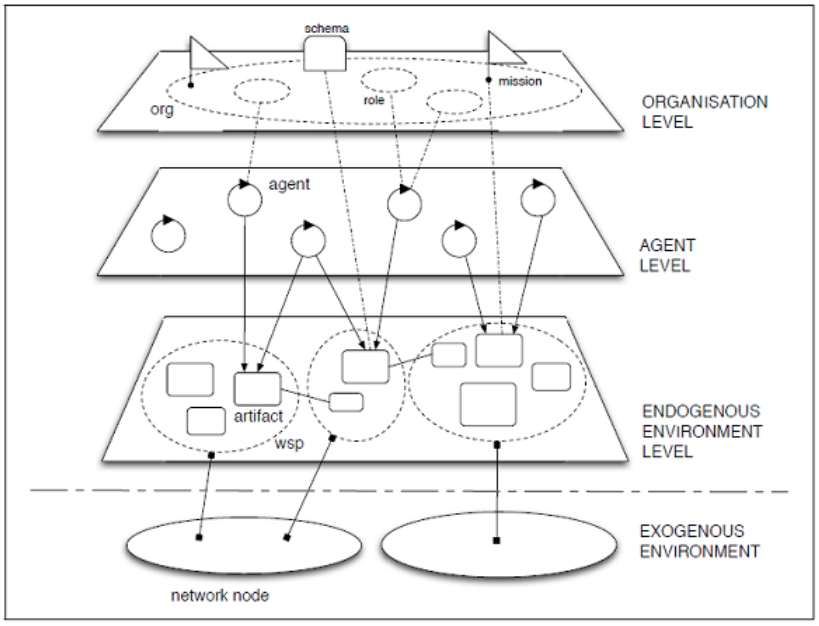


Figura 2.4: Piattaforme indipendendi che costituiscono JaCaMo

un meta-modello globale di programmazione che inglobi tutte le astrazioni

che ogni piattaforma presenta. Questo meta-modello ha lo scopo di definire rigorosamente tutte le connessioni e dipendenze tra le tre astrazione dei tre diversi livelli. Nello specifico si effettua il mapping concettuale tra la dimensione degli agenti e quella dell'environment, tra la dimensione degli agenti e quella dell'organizzazione, tra la dimensione dell'organizzazione e quella dell'environment.

A livello di agenti ci si riferisce al meta-modello di Jason dove le astrazioni sono principalmente ispirate all'architettura BDI. Per approfondire questo argomento si rimanda l'attenzione al paragrafo successivo dove si andrà nello specifico di un programma scritto in Jason.

Per quanto concerne l'ambiente, viene composto tramite una o più entità workspace istanziate tramite CArtAgo. Ogni workspace è composto da una serie di artefatti che offrono operazioni e proprietà osservabili, definendo l'interfaccia dell'artefatto stesso. L'esecuzione delle diverse operazioni può apportare modifiche alle proprietà osservabili degli artefatti oppure può generare eventi che a loro volta triggerano l'esecuzione di altri piani e quindi operazioni. Oltre agli artefatti all'interno dell'environment si trova un'entità chiamata "manual" con il compito di rappresentare la descrizione delle funzioni offerte da un artefatto.

Il meta-modello di Moise fornisce astrazioni per l'organizzazione degli agenti. Questi infatti possono essere suddivisi in gruppi e gerarchie in base al loro ruolo all'interno del sistema. Abbiamo quindi la specifica strutturale, che serve per descrivere la struttura dell'organizzazione, ed è formata dai gruppi e dalle entità di ruolo, mentre lo schema sociale serve per descrivere l'aspetto funzionale, ovvero la struttura dell'organizzazione in base alle varie missioni ed all'insieme dei goal delle entità. Esiste poi una normativa per legare i ruoli alle missioni e serve per vincolare il comportamento di un'agente quando entra in un gruppo che possiede già un determinato ruolo.

### 2.4.2 Jason

Jason[28] è un interprete per una versione estesa di AgentSpeak(L) che implementa la semantica operazionale di questo linguaggio e fornisce una piattaforma per lo sviluppo di sistemi multi-agenti con molte funzioni personalizzabili dall'utente. AgentSpeak(L) è un linguaggio di programmazione astratto basato sulla programmazione logica e sull'architettura BDI per

agenti cognitivi[39]. Questo linguaggio è un'estensione naturale della programmazione logica per architetture di agenti BDI e fornisce un framework astratto per programmarli[28]. Un agente in AgentSpeak(L) è quindi creato specificando belief, goal, intention e tutta la serie di piani che l'agente può gestire.

Analogamente in Jason[41], per definire un agente, si struttura il sistema individuando i seguenti elementi:

- *Belief* - rappresenta la conoscenza degli agenti. La rappresentazione dei belief avviene tramite litteral (predicati stile Prolog) i quali possono essere correlati allo stato interno dell'agente, oppure correlati allo stato osservabile degli artefatti. Possono essere aggiunti o rimossi a runtime, in accordo con lo stato dell'agente e con l'artefatto che dinamicamente si decide di osservare. È possono inoltre implementare delle regole Prolog-like, da utilizzare nella belief base dell'agente, che permettono di inferire nuovi belief partendo da quelli già esistenti.

- *Goals* - rappresentano i compiti che sono assegnati agli agenti. In Jason la rappresentazione dei goals e dei belief avviene tramite l'utilizzo di formule atomiche Prolog precedute dal simbolo "!". In un programma di solito i goal iniziali vengo dichiarati subito sotto alla belief base dell'agente. La restante parte del programma è composta dai piani che vengono utilizzati e combinati dinamicamente dall'agente per compiere i propri goal.

- *Plans* - rappresentano le procedure che gli agenti conoscono per raggiungere i goal. I piani di un agente sono composti da regole del tipo

  **Event : Context <- Body**

  Event fa riferimento ad un evento correlato a dei cambiamenti e delle modifiche all'interno dei goal dell'agente. Questi cambiamenti possono essere l'effetto sia del compimento di un certo goal e quindi dell'assegnazione di un nuovo goal, sia del fallimento durante l'esecuzione di un piano che porta di conseguenza al fallimento del raggiungimento del relativo goal. L'evento in questione potrebbe in alternativa provenire dall'ambiente esterno, ovvero dalla percezione di un cambiamento in relazione all'update dei valori osservabili degli artefatti.

Context è un'espressione booleana, all'interno della belief-base, utilizzata per specificare la condizione sotto la quale il piano viene mandato in esecuzione una volta che è stato triggerato dalla ricezione o dal rilevamento di un determinato evento.

Body indica l'insieme delle azioni da eseguire nel momento in cui il piano viene attivato. Le azioni possono essere sia interne, ovvero unicamente correlate allo stato interno dell'agente, sia esterne, ovvero fornite dall'ambiente circostante.

per quanto riguarda la programmazione dell'environment e degli artefatti, ovvero la programmazione delle parti del sistema functional-oriented, si può utilizzare CArtAgO, un framework che utilizza la piattaforma Java e le API Java-based.

Per comprendere al meglio tutto ciò appena spiegato, riporto un seguito un semplice esempio di programma scritto in Jason.

```
1  !print_fact(5).
2
3  +!print_fact(N)
4      <- !fact(N,F);
5          .print("Factorial of ", N, " is ", F).
6
7  +!fact(N,1) : N == 0.
8
9  +!fact(N,F) : N > 0
10     <- !fact(N-1,F1);
11         F = F1 * N.
```

Il programma in questione ha il compito di calcolare il fattoriale. La prima riga indica il goal iniziale dell'agente (stampare il fattoriale di 5), tutte le righe seguenti che iniziano con il simbolo "+" sono dei piani per l'agente. Il significato di questo simbolo può essere concepito come "Quando acquisisci il belief..", in questo caso specifico diventa "Quando acquisisci il belief print _ fact", sostanzialmente sta ad indicare l'evento che triggera l'esecuzione del piano in questione. Dopo il simbolo ":" abbiamo la condizione da verificare in seguito all'avvenuta dell'evento iniziale. Se la condizione è verificata si passa al body del piano. Il simbolo "." è un separatore sintattico, indica uno stop, come il ";" in Java.

Il programma scritto precedentemente può essere spiegato come segue:

- piano a riga 3 -> quando acquisisci il goal di stampare il fattoriale il corso delle azione da prendere è prima ottenere il goal !fact(..,..) e poi stampare la stringa "Factorial of .. is .." .
- piano a riga 7 -> quando acquisisci il goal di calcolare il fattoriale di 0, è risaputo che deve essere 1, quindi non devi fare altro.
- piano a riga 9 -> quando acquisisci il goal di calcolare il fattoriale di N, se N>0, il corso delle azioni da prendere è prima calcolare il fattoriale di N-1, e poi moltiplicare il valore ottenuto per N, per ottenere il fattoriale di N.

### 2.4.3 JaCa-Android per la creazione di sistemi ad agenti su smartphone

JaCa-Android[41] è stato invece introdotto per portare la programmazione agent based su smartphone. Questo framework porta la programmazione ad alto livello in ambito mobile rendendo possibile la creazione di applicazioni reattive (ad eventi generati dall'utente ed in generale dal contesto) e proattive (interagiscono e cooperano per raggiungere determinati obbiettivi). JaCa-Android è quindi il porting su Android di JaCa, antenato di JaCaMo quando ancora non era stata implementata la versione del sistema comprendente anche Moise per la gestione dell'organizzazione. L'infrastruttura JaCa è stata appositamente modificata, con l'aggiunta di un layer apposito che permette il controllo e l'accesso delle risorse fornite dal framework Android tramite artefatti, per essere applicata al dispositivo mobile. In figura 2.5 troviamo una rappresentazione grafica della piattaforma ed una generica applicazione eseguita su essa. In particolare si evidenziano gli artefatti creati per la gestione delle funzionalità base del sistema che possono essere direttamente utilizzati dagli agenti dell'applicazione.

Essendo la piattaforma completamente sviluppata in Java, viene eseguita come una normale applicazione Android. JaCa-Android, ed i relativi sistemi da cui deriva, è ideato partendo dal modello concettuale A& A, ovvero Agent& Artifacts [4], un meta-modello per modellare sistemi multi-agente in grado di introdurre la metafora ad alto livello di ambiente di lavoro cooperativo. Analogamente alle persone, gli agenti sono entità computazionali complesse in grado di compiere attività, mentre gli artefatti sono risorse e strumenti costruiti, usati e manipolati dinamicamente dagli agenti che

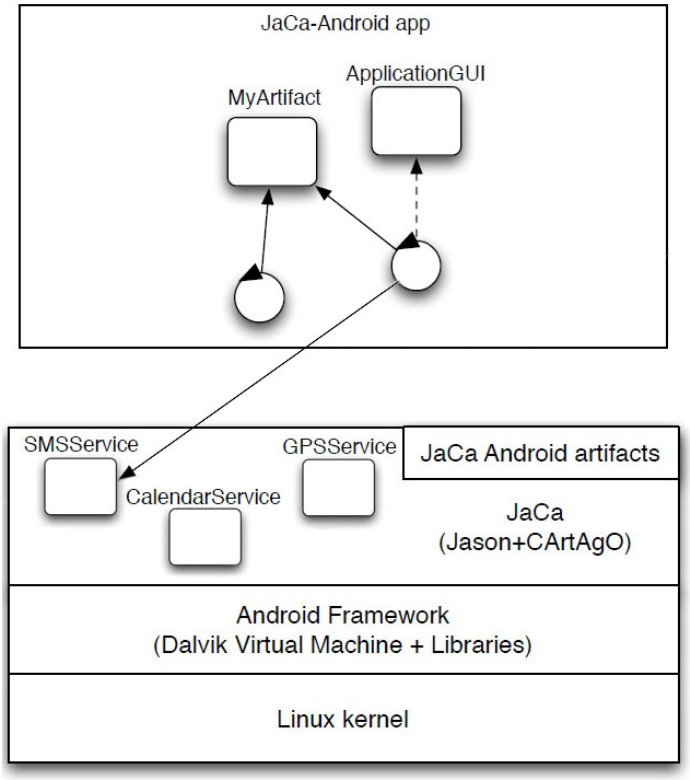


Figura 2.5: Rappresentazione della piattaforma JaCa-Android con un'applicazione in esecuzione su essa.

possono essere rapportati agli oggetti ed utensili che utilizziamo quotidianamente per svolgere i nostri lavori. Computazionalmente parlando [41] gli agenti sono entità autonome che incapsulano la logica decisionale ed esecutiva delle proprie azioni, gli artefatti sono entità passive che incapsulano le funzionalità ed i servizi utili agli agenti per raggiungere i loro obiettivi e l'ambiente è costituito in uno o più workspace, luoghi fisici o logici popolati da agenti ed artefatti. Questi ultimi presentano un'interfaccia composta da operazioni e proprietà osservabili. Le operazioni sono le azioni messe a disposizione degli agenti per l'interazione, mentre le proprietà osservabili sono dei valori utili per comprendere lo stato dell'artefatto stesso. Le proprietà vengono anche utilizzate per capire l'andamento di un'operazione, infatti i valori all'interno delle proprietà cambiano a differenza del compimento o meno dell'operazione.

Nello specifico delle applicazioni mobile, in particolare in JaCa-Android, gli artefatti hanno il compito di fare da tramite tra la piattaforma Android e l'applicazione fornendo le funzionalità per la gestione delle risorse del device e per l'interazione con le altre applicazioni presenti sul telefono. Per esempio, gli artefatti presenti all'interno dei workspace di JaCa-Android comprendono:

- artefatti per la gestione della console, fornendo funzionalità per stampare in standard output;
- artefatti per la gestione del workspace corrente, fornendo funzionalità per la creazione e gestione degli artefatti.

Per quanto concerne l'interazione con la piattaforma Android vengono forniti una serie di artefatti che incapsulano le funzionalità principali fornite dal sistema operativo, per permettere ai programmatori l'utilizzo di esse direttamente dal codice degli agenti, senza dover concentrarsi sui dettagli implementativi di basso livello. Nel dettaglio si possono trovare i seguenti artefatti:

- `SMSService / MailService` - fornisce le funzionalità per la ricezione, l'invio e la gestione generale di SMS ed email;
- `GPSService` - fornisce le funzionalità per accendere, spegnere il GPS e ricevere le informazioni sulla geolocalizzazione;

- `AccelerometerSensorService / GyroscopeSensorService / altri sensori` - fornisce le funzionalità per acquisire le informazioni riguardanti il relativo sensore;
- `CallService` - fornisce le funzionalità per gestire le telefonate;
- `ConnectivityService` - fornisce le funzionalità per gestire tutti i tipi di connessioni possibili supportate dal device;
- `CalendarService` - fornisce le funzionalità per gestire il calendario;
- `PhoneSettingsService` - fornisce le funzionalità per gestire le impostazioni del telefono per quanto concerne la suoneria e la vibrazione.

La piattaforma fornisce inoltre dei tipi predefiniti per la creazione di artefatti, tra i quali troviamo:

- `GUIArtifact` - artefatto base da estendere per definire un'interfaccia grafica;
- `FragmentGUIArtifact` - artefatto base da estendere per definire una parte della GUI modificabile e sostituibile dinamicamente;
- `BroadcastReceiverArtifact` - artefatto base da estendere per ricevere informazioni broadcast dai servizi Android;
- `ContentProviderArtifact` - artefatto base da estendere per lo sviluppo di applicazioni JaCa-Android conformi con applicazioni legacy già esistenti;
- `ServiceArtifact` - artefatto base da estendere sempre per applicazioni conformi con altre già esistenti, ma in particolare usato per fornire funzionalità a lungo termine in background;
- `ServiceConnectionArtifact` - artefatto base da estendere per l'interazione con applicazioni legacy che forniscono servizi basilari di Android.

Utilizzando questi artefatti adeguatamente è quindi possibile realizzare applicazioni JaCa-Android perfettamente integrate con il sistema, sia utilizzando componenti pre-sviluppati tramite l'SDK standard, sia implementando parti riutilizzabili in futuro da altre applicazioni non necessariamente di

JaCa-Android. L'integrazione è infatti caratteristica fondamentale di questo sistema, permettendo ai programmatori il riutilizzo del codice, eliminando l'onere di dover implementare l'intero sistema da zero.

# Capitolo 3

# Analisi dei requisiti e del problema

Nel seguente capito per prima cosa vengono esplicati i requisiti della tesi progettuale ed in un secondo momento vengono analizzati focalizzandosi sul problema in questione.

## 3.1 Formalizzazione dei requisiti tramite regole

Per prima cosa i requisiti forniti dai medici sono stati formalizzati in regole, per poi effettuare la successiva mappatura in piani BDI. Le regole generate sono le seguenti:

- se somministro "zero negativo" dopo 5min compare "hai somministrato fibrinogeno e acido tranexamico?" se ancora non sono stati cliccati e/o somministrati
- se somministro zero negativo 3U e acido tranexamico e fibrinogeno compare "attivare PTM?"
- se clicco "TIOPENTONE" compare "attenzione all'ipotensione!"
- se inizio ALS ogni 3 min compare "somministra adrenalina 1mg" (magari associato ad un allarme sonoro)

- se inizio ALS dopo 5min compare "decompressione pleurica?" se non già fatta e cliccata (magari associato ad allarme sonoro)
- se clicco "Frattura Esposta SI" quando cambio stanza uscendo da shock room e non ho ancora somministrato antibiotico chiedere "antibiotico?"
- ogni 15min dopo inizio delle manovre "TOURNIQUET" o "REBOA" o "TORACOTOMIA RESUSCITATIVA" fare comparire " TOURNIQUET (o REBOA o TORACO...) da 15 min, poi ...da 30min ... poi "da 45 min" ecc... (magari associato ad allarme sonoro)
- quando cambio stanza in uscita da shock room compare "hai posizionato SNG e foley vescicale?"

In funzione dell'acquisizione in maniera continua dei parametri vitali:

- se paziente con tubo tracheale o dopo intubazione tracheale dopo 5 min compare "check EtCO2" se ancora non è arrivato alcun dato di EtCO2
- se SpO2 $< 90\%$ compare "FiO2 impostata al 100% ?"
- se clicco "infusione a pressione" e la pressione arteriosa sistolica supera 110mmHg compare "sospendere infusione a pressione?"

### 3.1.1 Glossario e disambiguazione

In prima istanza si è deciso di identificare un glossario per spiegare al meglio ogni termine significativo ricorrente all'interno dei requisiti.

#### Glossario

L'interpretazione in linguaggio naturale è legato alla mia conoscenza di base e alle mie ricerche sul dominio di interesse.

- **Zero negativo** - Sangue Rh negativo del gruppo 0, considerato il più puro in assoluto, raro e pregiato. Questo tipo di sangue[9] può essere donato a tutti i tipi di sangue umano, a parte qualche rara eccezione, ma può riceverlo solamente da donatori anch'essi Rh negativi. Viene

utilizzato per ogni trasfusione nel dipartimento del DEA grazie alla sua versatilità, visto che è appunto accettato da tutti gli altri gruppi sanguigni.

- **Fibrinogeno** - Sostanza[6] presente nel sangue circolante, dalla quale, nella coagulazione, si forma la fibrina. È una glicoproteina del plasma sanguigno sintetizzata dal fegato e dal tessuto endoteliale, importante per il processo di emostasi.

- **Acido tranexamico** - È una molecola usata in medicina per l'inibizione del sistema della fibrinolisi e per questo classificata come antifibrinolitico[24]. Il meccanismo d'azione di questa molecola si basa su un blocco della formazione di plasmina, attraverso l'inibizione dell'attività proteolitica degli attivatori del plasminogeno, risultandone alla fine un'inibizione della lisi del coagulo di sangue. Viene utilizzata tendenzialmente per trattare o prevenire un'eccessiva perdita di sangue da traumi importanti.

- **Protocollo Trasfusione Massiva (PTM)** - Procedura[14] nella quale si pratica una sostituzione di una grande quantità di sangue, a volte anche la totalità del sangue, in breve tempo.

- **Tiopentone** - Anestetico appartenente alla classe dei barbiturici a rapida azione, usato per via endovenosa. Per la rapidità con cui induce il sonno e la breve durata d'azione, è usato come anestetico negli interventi chirurgici, anche durante la gravidanza. Questo medicinale, fino a poco tempo fa, era un farmaco cardine della World Health Organization's List of Essential Medicines[27], una lista di farmaci essenziali per il sistema sanitario di base stilata dall'organizzazione mondiale per la sanità.

- **Ipotensione** - Condizione clinica[11] che si riscontra quando in un soggetto si rileva una pressione sanguigna arteriosa massima inferiore ai 100 mmHg, poiché i valori riconosciuti come normali nella popolazione sana oscillano fra i 110-130 mmHg di sistolica.

- **Advanced Life Support (ALS)** - Protocollo di rianimazione cardiopolmonare avanzata [2] utilizzato dallo staff medico ed infermieristico

con la finalità di monitorare e stabilizzare lo stato di salute del paziente. Durante questa procedura si possono somministrare farmici o attuare manovre invasive fino all'arrivo in ospedale.

- **Adrenalina** - È un ormone sintetizzato nella porzione midollare del surrene che prende il nome anche di epinefrina[1]. Una volta secreta e rilasciata in circolo, l'adrenalina accelera la frequenza cardiaca, restringe il calibro dei vasi sanguigni, dilata le vie aeree bronchiali ed esalta la prestazione fisica. Sostanzialmente l'adrenalina migliora la reattività dell'organismo, preparandolo in tempi brevissimi alla cosiddetta reazione di "attacco o fuga". In ambito clinico l'adrenalina è correntemente usata nella terapia dello shock anafilattico, dell'arresto cardiaco e aggiunta agli anestetici locali per ritardarne l'assorbimento.

- **Decompressione pleurica** - Stato di decompressione del cavo pleurico [12]. In presenza di un trauma toracico o di un pneumotorace iperteso bisogna effettuare la decompressione per drenare il torace.

- **Frattura esposta** - Frattura con fuoriuscita del tessuto osseo al di fuori della cute[7] e quindi a contatto con l'ambiente esterno. Questo tipo di frattura, a differenza delle fratture chiuse (il mantello cutaneo rimane integro), può degenerare un una complicanza molto temibile in campo ortopedico, ovvero l'infezione. Nel momento in cui il tessuto osseo entra in contatto con l'ambiente 'esterno subisce una contaminazione germica e questi germi trovano nell'ambito del tessuto osseo un terreno estremamente fertile per la loro proliferazione causando infezioni importanti.

- **Tourniquet** - Procedura nella quale si fa uso di dispositivi di compressione [23] per controllare il flusso di sangue arterioso e venoso. La pressione viene applicata attorno ad una porzione di arto nella posizione desiderata, dopodiché questa viene trasferita alle pareti dei vasi sanguigni causandone l'occlusione o la limitazione temporanea. In ambito chirurgico la tourniquet viene utilizzata per occludere il flusso sanguigno arterioso dopo dissanguamento, per produrre un campo operatorio relativamente esangue e per ridurre al minimo la perdita di sangue. In situazioni di emergenza invece viene utilizzato per fermare il sanguinamento traumatico in modo tale che le cure mediche possa-

no essere fornite in tempo utile prima che la persona ferita sanguini eccessivamente.

- **Reboa** - Tecnica utilizzata per i pazienti che stanno sanguinando eccessivamente e avvicinandosi rapidamente alla morte per lesioni al torace, all'addome o al bacino. Questa tecnica[20] comporta il posizionamento rapido di un catetere flessibile nell'arteria femorale, manovrandolo nell'aorta e gonfiando un palloncino sulla punta. Ciò interrompe il flusso di sangue oltre il palloncino, bloccando sostanzialmente qualsiasi sanguinamento, fermando anche tutto il flusso sanguigno più lontano dalla posizione del palloncino. Questa è una manovra temporanea, è infatti usata tendenzialmente per guadagnare tempo quando si trasferisce un paziente gravemente ferito in sala operatoria.

- **Toracotomia resuscitativa** - Pratica[40], conosciuta anche con il nome di toracotomia d'emergenza (EDT), svolta o immediatamente sulla scena del trauma, o al dipartimento d'emergenza come parte integrante del processo di rianimazione iniziale. Si tratta di una metodologia rianimatoria estrema che, anche nei sistemi meglio organizzati ed efficienti, è caratterizzata da un'alta mortalità e gravata da difficoltà operative.

- **Shock room** - Struttura del Dipartimento Emergenza e Accettazione istituita come centro specializzato per intervenire tempestivamente sui pazienti che arrivano al pronto soccorso in condizioni critiche.

- **Sonda gastrica per nutrizione (SNG)** - Dispositivo medico usato per provvedere alla nutrizione artificiale[8] di pazienti che non sono in grado o rifiutano di nutrirsi mangiando normalmente. Questa pratica di alimentazione indotta prende il nome di nutrizione enterale e si riserva ai pazienti che, pur mantenendo l'integrità funzionale del tratto gastroenterico, non possono assumere gli alimenti in maniera naturale. Consiste infatti nella somministrazione di nutrienti in forma liquida attraverso una sonda. Per posizionare il sondino si può ricorrere ad un accesso naturale (sonda nasogastrica, nasoduodenale, nasodigiunale) oppure ad uno stoma artificiale aperto a livello faringeo, gastrico o digiunale. La collocazione finale della punta della sonda posizionata attraverso queste vie è sempre lo stomaco o il primo tratto dell'intestino.

- **Foley verticale** - Il catetere Foley[5] è un utensile medico che prende il nome dal Dottor Frederick Foley che lo inventò quasi un secolo fa. Si tratta di un tubo in gomma, generalmente lattice o silicone, flessibile e di vari diametri che viene inserito lungo l'uretra ed ancorato alla vescica mediante un palloncino gonfiato con una soluzione fisiologica. Viene utilizzato in moltissime situazioni come per pazienti paralizzati, incontinenti, in stato comatoso o anestetizzati.

- **Intubazione tracheale** - Pratica tra le più comuni tra le varie intubazioni possibili [35]. La pratica generica consiste nell'inserimento di un tubo attraverso le corde vocali per permettere la respirazione di una persona non in grado di respirare oppure per proteggere le vie aeree da inalazioni di materiale gastrico. L'intubazione tracheale può avvenire per via rinotracheale, ovvero facendo passare il tubo dal naso, oppure più comunemente orotracheale, ovvero facendo passare il tubo dalla bocca.

- **End tidal Carbon Dioxide (EtCO2)** - Indicatore della pressione parziale, o concentrazione massima, di CO2 a fine dell'espirazione[17], la quale viene tendenzialmente espressa in percentuale di CO2 o mmHg. I valori normali variano tra il 5% ed il 6% CO2, o anche 35-45 mmHg. Questo valore rispecchia il flusso sanguigno polmonare, poiché il gas è trasportato per mezzo delle vene alla parte destra del cuore e di conseguenza pompato ai polmoni per mezzo del ventricolo destro. Quando la CO2 si diffonde fuori dai polmoni, nell'aria che verrà esalata, si utilizza un apparecchio chiamato capnometro per misurarne la pressione parziale o concentrazione massima.

- **SpO2** - Indicatore della saturazione di ossigeno nel sangue[21]. La saturimetria, tecnica di misurazione del livello di SpO2, consente di conoscere in tempi brevi la percentuale di ossigeno presente nell'emoglobina, un parametro utile per controllare se la funzione respiratoria di un paziente è adeguata. Questa metodologia è usata con successo anche per valutare la presenza e la gravità di alcune malattie respiratorie, per lo screening nelle cardiopatie congenite e per impostare una terapia idonea a ripristinare i corretti valori di saturazione. Per effettuare la saturimetria si possono utilizzare sia dei campioni di sangue, sia il pulsossimetro, un piccolo dispositivo che si aggancia ad un dito.

I valori che vengono considerati normali oscillano tipicamente tra il 95% ed il 100%, se scendono sotto il 90% bisogna attuare operazioni sul paziente per stabilizzare nuovamente lo stato di salute ottimale.

- **Infusione a pressione** - Pratica utilizzata per iniettare in breve tempo una grande quantità di fluidi all'interno del sangue. Utilizzata solitamente per un monitoraggio rapido della pressione e per procedure di trasfusione.
- **Pressione arteriosa sistolica** - Comunemente chiamata pressione massima[18], è il valore di pressione arteriosa nel momento in cui il cuore è in fase di contrazione, al fine di spingere il sangue in circolo. In altre parole, è la pressione sanguigna a ogni battito del cuore. Se la pressione sistolica subisce dei cali o degli aumenti permanenti, è indice di malessere nell'organismo del paziente.

### 3.1.2 User Stories

In un secondo momento sono state create le user stories, pratica comune in Scrum ed altre metodologie Agile, per cercare di contestualizzare i requisiti ed esprimere al meglio quale dovrebbe essere il comportamento del sistema. Tendenzialmente le User Stories vengono utilizzate per esprimere le funzionalità del sistema tramite il formalismo "In qualità di...", "vorrei..." "affinché...". Essendo il problema trattato in questa tesi un problema formalizzato tramite regole, in base alle sue peculiarità si è deciso di specificare una User Stories unica in grado di accomunare tutte le regole dei requisiti.

- In qualità di utente, vorrei premere un pulsante (diverso dal tipo di farmaco somministrato o dalla procedura attivata) affinchè il sistema mi invii delle notifiche immediatamente o dopo un certo lasso di tempo predefinito.
  - Verificare che la notifica di risposta sia pertinente con il pulsante premuto.

## 3.2 Prima analisi delle regole

Allo stato attuale, basandomi sulle mie conoscenze e ricerche, in letteratura non esistono contributi validi nei quali si applichi la tecnologia ad agenti

ed il modello BDI a questo tipo di problema. Le varie applicazioni ad agenti che operano nel campo medico possono essere divise in quattro diverse categorie[31]:

- *Gestione dei dati clinici* - accesso, integrazione e condivisione di dati per facilitare il lavoro dei medici in sistemi distribuiti e per analisi statistica;
- *Sistemi di supporto alle decisioni, pianificazione e allocazione di risorse* - tecniche di coordinamento utilizzate per riallocare le figure professionali al meglio in campo medico;
- *Cura da remoto* - monitoraggio di pazienti da remoto garantendo controlli continuativi ed assistenza immediata in caso di malesseri.

Le regole servono per incapsulare al loro interno la conoscenza pratica del dominio. Nel caso specifico del modello BDI viene catturato il concetto di conoscenza pratica operativa, in quanto tramite le regole si cattura la conoscenza necessaria per agire sul mondo. Una volta creata la regola, la si mappa sotto forma di piani, così da poterla inserire direttamente all'interno dell'agente, fornendogli le conoscenze di cui ha bisogno.

Essendo il sistema già esistente basato sul modello BDI, il quale prevede la composizione dei piani secondo la struttura **Event : Context <- Body**, ed essendo inoltre un requisito la creazione di un ulteriore agente BDI in grado di fornire le funzionalità necessarie per la generazione dei Warning, si è deciso di analizzare ogni regola evidenziando i tre componenti caratteristici della struttura dei piani. Di seguito si è utilizzato il colore verde per Event, il rosso per Context e il blu per Body. Inoltre, per un fattore di comodità e rapidità di lettura, d'ora in avanti si farà riferimento alle varie regole tramite una numerazione.

1. se somministro "zero negativo" dopo 5min compare "hai somministrato fibrinogeno e acido tranexamico?" se ancora non sono stati cliccati e/o somministrati
2. se somministro zero negativo 3U e acido tranexamico e fibrinogeno compare "attivare PTM?"
3. se clicco "TIOPENTONE" compare "attenzione all'ipotensione!"

4. se inizio ALS ogni 3 min compare "somministra adrenalina 1mg" (magari associato ad un allarme sonoro)

5. se inizio ALS dopo 5min compare "decompressione pleurica?" se non già fatta e cliccata (magari associato ad allarme sonoro)

6. se clicco "Frattura Esposta SI" quando cambio stanza uscendo da shock room e non ho ancora somministrato antibiotico chiedere "antibiotico?"

7. ogni 15min dopo inizio delle manovre "TOURNIQUET" o "REBOA" o "TORACOTOMIA RESUSCITATIVA" fare comparire " TOURNIQUET (o REBOA o TORACO...) da 15 min, poi ...da 30min ... poi "da 45 min" ecc... (magari associato ad allarme sonoro)

8. quando cambio stanza in uscita da shock room compare "hai posizionato SNG e foley vescicale?"

In funzione dell'acquisizione in maniera continua dei parametri vitali:

9. se paziente con tubo tracheale o dopo intubazione tracheale dopo 5 min compare "check EtCO2" se ancora non è arrivato alcun dato di EtCO2

10. se SpO2 $< 90\%$ compare "FiO2 impostata al 100% ?"

11. se clicco "infusione a pressione" e la pressione arteriosa sistolica supera 110mmHg compare "sospendere infusione a pressione?"

## 3.3 Rappresentazione delle regole come piani

Analizzando i requisiti espressi si può da subito effettuare una divisione tra le varie regole. Alcune infatti sono tempo dipendenti mentre altre no (dipendono solamente o dalle manovre effettuate e dai farmaci somministrati, o dai parametri vitali del paziente). Per l'individuazione della tempo dipendenza si è valutato lo stato del paziente e non la modalità di visualizzazione dei Warning, in altre parole si considerano tempo dipendenti quelle regole

nelle quali è presente un lasso di tempo all'interno del quale bisogna osservare l'occorrenza di determinati fenomeni mentre non sono considerate tempo dipendenti le regole che prevedono la visualizzazione ripetuta di uno stesso Warning dopo un certo periodo di tempo. Di seguito viene esplicitata questa distinzione delle categorie ed inoltre si riscriveranno i requisiti in maniera meno discorsiva mantenendo la logica della struttura dei piani scritti in Jason (Consigli di lettura: leggere ed interpretare il simbolo ":" come "se" ed il simbolo "<-" come "allora").

Le regole tempo dipendenti sono:

1. Somministro "Xero Negativo" al tempo t : al tempo t+5min non si ha somministrato fibrinogeno e acido tranexamico <- compare per 10 sec "hai somministrato fibrinogeno e acido tranexamico?"

5. Inizio ALS (Advanced Life Support) al tempo t : al tempo t+5min non è stata fatta la decompressione pleurica <- compare "decompressione pleurica?"

9. Se è stata fatta l'intubazione tracheale al tempo t : al tempo t+5min non sono stati ricevuti valori dall'EtCO2 <- compare "check EtCO2"

Mentre le regole tempo indipendenti sono:

2. Somministro 3 unità di zero negativo, acido tranexamico e fibrinogeno <- compare "attivare PTM?"

3. Somministro tiopentone <- compare "attenzione all'ipotensione!"

4. Se inizio ALS <- compare "somministra 1mg di adrenalina" ogni 3 min

6. Quando esco da shock room : se il paziente ha una frattura esposta e non ho ancora somministrato antibiotico <- compare "attivare profilassi antibiotica?"

7. Attivando le procedure tourniquet, reboa o toracotomia resuscitativa <- "tourniquet/reboa/ toracotomia resuscitativa attivata da n min" ogni 15 min.

8. Uscendo da shock room <- compare "hai posizionato SNG e foley vescicale?"

10. Quando SpO2 scende sotto al 90% <- compare "FiO2 impostata al 100% ?"

11. Usando l'infusore a pressione : la pressione arteriosa sistolica supera 110mmHg <- compare "sospendere infusione a pressione?"

Il passo successivo sarà la formalizzazione delle regole in piani seguendo il modello BDI utilizzando del pseudocodice[31] nel quale:

- `+ev` indica l'evento che triggera il piano. Nel caso di studi in questione ci sono solamente due tipi di eventi:
  - l'aggiunta di un nuovo belief dovuto dalla percezione di un nuovo evento relativo al trauma rappresentato con `+ev(EvInfo, Time)`, oppure di un evento temporale rappresentato con `+trig( TrigName, Time)` tipicamente usato per la gestione delle regole caratterizzate dal Timeout. A sua volta `EvInfo` può essere di diverse tipologie in base alla provenienza:
    - la somministrazione di un nuovo farmaco viene rappresentato con `drug(DrugType, Quantity)`;
    - la pratica di una procedura eseguibile istantaneamente, non prolungata nel tempo, viene rappresentata con `procedure( ProcName)`;
    - la pratica di una procedura tempo dipendente, viene rappresentata con `procedure_started(ProcName)` e `procedure_stopped(ProcName)`;
    - l'entrata o l'uscita da una determinata stanza rappresentata rispettivamente da `room_in(RoomName)` e `room_out(RoomName)`;
    - un'azione di input inserita direttamente dall'utente rappresentata da `action(InputAction, Time)`.
  - la modifica di un belief mentre si tiene traccia di un particolare stato (in seguito ad una determinata regola). Come per esempio nella regola 9 tramite `intubation(true/false)` si controlla lo stato della procedura.

- `co` indica il contesto nel quale il piano si può considerare applicabile. Può essere considerata come condizione che si basa sui belief attualmente presenti nella conoscenza dell'agente. Nel caso di studi attuale si possono trovare due tipi di eventi all'interno del contesto:
  - quelli la cui occorrenza o meno è avvenuta nel passato, rappresentati rispettivamente tramite `ev(EvInfo, Time)` e $\nexists$`ev(EvInfo, Time)`
  - la presenza di un Warning rappresentato da `warn(WarnName, Time)`
- `act` indica la sequenza di azioni da compiere quando il piano è in azione. Nello specifico di Trauma Tracker le sequenze sono formate da azioni singole le quali possono essere:
  - l'inserimento o la sottrazione di un belief riguardante un Warning, rappresentati rispettivamente da `+warn(WarnTime, Time)` e `-warn(WarnTime, Time)`. Quest'ultimo belief avviene quando si vuole cancellare la visualizzazione di una certa notifica.
  - la generazione di un trigger, ovvero di un evento utilizzato per la gestione delle regole tempo dipendenti. La sintassi dell'azione è `set_trig(TrigName, When)` ed ha il compito di generare un nuovo `trig(trigName, Time)` al tempo indicato da `When`

Regole tempo dipendenti:

- ```
  // Regola N.1

  +ev(drug(zero_negative,_),T)
      <- set_trig(checkTranAndFibr, T+300).

  +trig(checkTranAndFibr, T): ∄ev(drug(tranex, _),_) ∨∄ev(
      drug(fibri,_),_)
      <- +warn(checkTranAndFibri, T).

  +ev(drug(tranex,_)) : warn(checkTranAndFibr,_) ∧ev(drug(
      fibri,_),_)
      <- -warn(checkTranAndFibr,_).

  +ev(drug(fibri,_)) : warn(checkTranAndFibr,_) ∧ev(drug(
      tranex,_),_)
  ```
  ```

```
13      <- -warn(checkTranAndFibr,_).
14
15 +warn(checkTranAndFibr, T) : message(checkTranAndFibr,
       WarnMsg)
16      <- displayWarnMsg(checkTranAndFibr,   WarnMsg).
17
18 -warn(checkTranAndFibr, T)
19      <- hideWarnMsg(checkTranAndFibr).
```

Il primo piano serve per tenere traccia del tempo, infatti dopo 5 min (300 sec) della pressione del pulsante relativo a "zero negativo" viene inviato un evento di tipo trig, il quale triggera il secondo piano nel quale, se non sono ancora stati somministrati fibrogeno o acido tranexamico si genera un Warning. Il terzo e quarto piano servono per rimuovere il messaggio di Warning nel momento in cui i farmaci in questione siano stati somministrati, mentre gli ultimi due servono per visualizzare o meno il messaggio.

```
1 // Regola N.5
2
3 +ev(procedure_started(als),T)
4      <- set_trig(checkPleuDecomp, T+300).
5
6 +trig(checkPleuDecomp, T): #ev(procedure(pleu_decop,_),_)
7      <- +warn(pleuDecomp, T).
8
9 +ev(procedure_stoped(als, T))
10     <- -warn(pleuDecomp, T).
11
12 +warn(pleuDecomp, T) : message(pleuDecomp, WarnMsg)
13     <- displayWarnMsg(pleuDecomp,   WarnMsg).
14
15 -warn(pleuDecomp, T)
16     <- hideWarnMsg(pleuDecomp).
```

Il primo piano viene triggerato con l'inizio della procedura ALS generando, dopo 5 min, un evento di tipo trig, il quale a sua volta genera un Warning nel caso in cui non sia stata avviata nel frattempo la procedura di decompressione pleurica. Il terzo piano viene triggerato quando la procedura termina generando l'eliminazione dei Warning. Gli ultimi due piani servono per visualizzare o meno il messaggio.

```
// Regola N.9

+ev(procedure_started(intub),T)
    <- set_trig(checkEtCO2, T+300).

+intubation(true)
    <- set_trig(checkEtCO2, T+300).

+trig(checkEtCO2, T): ∄ev(vs(etCO2,_),_)
    <- +warn(checkEtCO2, T).

+ev(procedure_stoped(intub, T))
    <- -warn(checkEtCO2, T).

+warn(checkEtCO2, T) : message(checkEtCO2, WarnMsg)
    <- displayWarnMsg(checkEtCO2, WarnMsg).

-warn(checkEtCO2, T)
    <- hideWarnMsg(checkEtCO2).
```

I primo piano viene triggerato all'avvenuta intubazione del paziente, generando, dopo 5 min, un evento di tipo trig. Lo stesso evento viene generato anche nel caso in cui il paziente sia già stato intubato. La creazione dell'evento Warning avviene nel terzo piano nel caso in cui non si abbiano informazioni sull'EtCO2 (uno dei parametri vitali). Il quarto piano viene triggerato al termine della procedura provocando l'eliminazione del Warning. Gli ultimi due piani servono per visualizzare o meno il messaggio.

Regole tempo indipendenti:

```
// Regola N.2

+ev(drug(zero_negative,T)): zero_negative(3) ∧∃ ev(drug(
    tranex,_),_) ∧∃ ev(drug(fibri,_),_)
    <- +warn(activateMTP, T).

+warn(activateMTP, T) : message(activateMTP, WarnMsg)
    <- displayWarnMsg(activateMTP, WarnMsg).
```

Il primo piano viene triggerato con la somministrazione di sangue zero negativo (considerato per semplicità come farmaco). Nel caso in cui se ne siano somministrate 3 unità, più acido tranexamico e

fibrogeno, allora viene attivato il Warning. L'altro piano serve per la visualizzazione del messaggio.

```
// Regola N.3

+ev(drug(thiopental,T))
    <- +warn(checkHypoTh, T).

+warn(checkHypoTh, T) : message(checkHypoTh, WarnMsg)
    <- displayWarnMsg(checkHypoTh, WarnMsg).
```

Il primo piano viene triggerato con la somministrazione del tiopentone ed immediatamente viene generato il Warning. L'altro piano serve per la visualizzazione del messaggio.

```
// Regola N.4

+ev(procedure_started(als,T))
    <- +warn(adminAdren, T).

+ev(procedure_stopped(als,T))
    <- -warn(adminAdren, _).

+warn(adminAdren, T) : message(adminAdren, WarnMsg)
    <- displayWarnMsg(adminAdren, WarnMsg).

-warn(adminAdren,T)
    <- hideWarnMsg(adminAdren).
```

Il primo piano viene triggerato quando si inizia la procedura ALS generando il Warning, mentre il secondo piano viene triggerato al termine della procedura, causando lo scomparsa del Warning. Gli ultimi due piani servono per visualizzare o meno il messaggio.

```
// Regola N.6

+ev(room_out(shock_room,T)) : fracture(true) ∨∄ ev(drug(abt
    ),_)
    <- +warn(checkABT, T).

+warn(checkABT, T) : message(checkABT, WarnMsg)
    <- displayWarnMsg(checkABT, WarnMsg).
```

Il primo piano viene triggerato quando si esce dalla shock room, nel caso in cui ci sia una frattura esposta e non si abbia ancora somministrato dell'antibiotico, mentre l'altro serve per la visualizzazione del messaggio.

```
// Regola N.7

+ev(procedure_started(tourniq,T))
    <- +warn(tourniquet, T).

+ev(procedure_stopped(tourniq,T))
    <- -warn(tourniquet, _).

+warn(tourniquet, T) : message(tourniquet, WarnMsg)
    <- displayWarnMsg(tourniquet, WarnMsg).

-warn(tourniquet,T)
    <- hideWarnMsg(tourniquet).

+ev(procedure_started(reb,T))
    <- +warn(reboa, T).

+ev(procedure_stopped(reb,T))
    <- -warn(reboa, _).

+warn(reboa, T) : message(reboa, WarnMsg)
    <- displayWarnMsg(reboa, WarnMsg).

-warn(reboa,T)
    <- hideWarnMsg(reboa).

+ev(procedure_started(tora,T))
    <- +warn(toracotomia, T).

+ev(procedure_stopped(tora,T))
    <- -warn(toracotomia, _).

+warn(toracotomia, T) : message(toracotomia, WarnMsg)
    <- displayWarnMsg(toracotomia, WarnMsg).

-warn(toracotomia,T)
    <- hideWarnMsg(toracotomia).
```

I piani di questa regola sono analoghi a gruppi di quattro, visto che

il comportamento deve essere lo stesso ma per tre procedure diverse (tourniquete, reboa e toracotomia resuscitativa). Il primo piano viene triggerato all'inizio della procedura tourniquete generando il Warning, il secondo piano viene triggerato al termine della stessa procedura causando l'eliminazione del messaggio di Warning, mentre il terzo e il quarto piano servono rispettivamente per la visualizzazione o meno dei messaggi.

```
// Regola N.8

+ev(room_out(shock_room,T))
    <- +warn(checkFTandFoley, T).

+warn(checkFTandFoley, T) : message(checkFTandFoley,
    WarnMsg)
    <- displayWarnMsg(checkFTandFoley, WarnMsg).
```

Il primo piano viene triggerato quando si esce dalla shock room, creando direttamente il Warning, mentre l'altro serve per la visualizzazione del messaggio.

```
// Regola N.10

+ev(vs(spo2,V),T) : V < 90
    <- +warn(checkFIO2, T).

+warn(checkFIO2, T) : message(checkFIO2, WarnMsg)
    <- displayWarnMsg(checkFIO2, WarnMsg).
```

Il primo piano viene triggerato al momento della misurazione dell'SpO2. Se il valore misurato è minore di 90 viene generato il Warning. L'altro piano serve per la visualizzazione del messaggio.

```
// Regola N.11

+ev(procedure_started(infus,P),T) : P > 110
    <- +warn(infusSuspension, T).

+warn(infusSuspension, T) : message(infusSuspension,
    WarnMsg)
    <- displayWarnMsg(infusSuspension, WarnMsg).
```

Il primo piano viene triggerato alla pressione del pulsante relativo alla procedura "infusione a pressione". Se in quel momento la pressione sistolica è maggiore di 110, viene generato il Warning. L'altro piano serve per la visualizzazione del messaggio.

Nel codice appena spiegato non sono stati scritti i messaggi testuali che compaiono. Nell'elenco seguente verranno quindi mostrati gli identificatori usati precedentemente con i relativi messaggi per i Warning.

- `checkTranAndFibr` - Hai somministrato fibrinogeno e acido tranexamico?
- `activateMTP` - Attivare Protocollo Trasfusione Massiva (PTM)?
- `checkHypoTh` - Attenzione all'ipotensione!
- `adminAdren` - Somministra 1mg di adrenalina.
- `pleuDecomp` - Decompressione pleurica?
- `checkABT` - Attivare profilassi antibiotica?
- `tourniquet` - Tourniquet attivata da n min.
- `reboa` - Reboa attivata da n min.
- `toracotomia` - Toracotomia resuscitativa attivata da n min.
- `checkFTandFoley` - Hai posizionato SNG e Foley vescicale correttamente?
- `checkEtCO2` - Controllare EtCO2.
- `checkFIO2` - FiO2 impostata al 100% ?
- `infusSuspension` - Sospendere infusione a pressione?

# Capitolo 4

# Progettazione e sviluppo

Nel capitolo seguente verrà mostrata tutta la fase di progettazione, implementazione e sviluppo dell'estensione di Trauma Tracker.

## 4.1 Architettura del sistema

Il prototipo attualmente in uso dell'applicazione Trauma Tracker [32] è stato progettato basandosi sull'architettura BDI, in quanto le caratteristiche offerte da questo modello architetturale, facilitano la modellazione di un Personal Assistant Agent con la reattività richiesta per il lavoro di tracking e la pro-attività richiesta per il compito di assistenza. Nello specifico, le capacità reattive richieste vengono modellate in termini di un insieme di piani predefiniti i quali vengono innescati da eventi che si verificano nell'ambiente. Questi eventi, nel caso di Trauma Tracker, sono per la maggior parte azioni che il team di medici svolge sul paziente. Possono essere per esempio la somministrazione di un farmaco, l'attuazione di una procedura, lo spostamento del paziente da una stanza all'altra all'interno del DEA, l'acquisizione di file multimediali ed annotazioni relativi allo stato di salute del paziente o anche cambiamenti dei parametri vitali. Sempre seguendo il modello concettuale BDI ed A & A, sono stati utilizzati gli artefatti per modulare l'insieme di percezioni ed azioni disponibili agli agenti. Tra i vari artefatti creati all'interno dell'applicazione, sui quali si trovano maggiori dettagli nel paragrafo 1.2.3, si pone una particolare attenzione al `NotesStream` il quale ha il compito di memorizzare continuamente il flusso di note che trasportano informazioni sugli eventi e fornire altre funzionalità relative alla generazio-

ne di report e consegna al servizio di gestione dei report. Questo artefatto rappresenta il punto di giunzione tra il prototipo già esistente e l'espansione che si vuole creare nello svolgimento della tesi.

Per quanto concerne gli agenti, troviamo diversi agenti che vengono considerati a livello logico come un unico agente concettuale di alto livello. Questi agenti, allo stato attuale sono:

- *Trauma Initializer Agent* - responsabile dell'inizializzazione della fase di tracking e conseguentemente del collezionamento di tutte le informazioni iniziali essenziali per la creazione del report, tra cui lo stato di salute iniziale del paziente e l'identità del trauma Leader.

- *Tracker Agent* - rappresenta la parte principale del Personal Assistan Agent, in quanto incapsula la maggior parte delle funzionalità offerte. Ha il compito di tenere traccia degli eventi captati dall'ambiente, costruire man mano il report e rendere consapevole il Trauma Leader di specifiche informazioni particolarmente necessarie.

- *Speech Interpreter Agent* - responsabile dell'interpretazione dei comandi vocali. In particolare, questo agente è in grado di riconosce se uno specifico comando può essere accettato in quel preciso istante del processo di gestione del trauma, basandosi sulle azioni del flusso di lavoro, e tradurlo in un evento appropriato per essere percepito dal Tracker Agent.

Per quanto riguarda la progettazione del livello di assistenza tramite l'inserimento dei Warning si è pensato di integrare all'interno del sistema un ulteriore agente dal nome Warning Agent. Questo agente ha come compito l'intera gestione dei Warning. La sua belief base è basata sulle note provenienti da `NodeStream`, artefatto che fornisce le funzionalità per la creazione e gestione delle note. Si è pensato inoltre di aggiungere un ulteriore artefatto, che prende il nome di `EventStream`, il quale serve per gestire gli eventi prodotti in base alle note generate. In figura 4.1 abbiamo la rappresentazione dell'architettura del sistema nel suo complesso, ed in particolare nella parte evidenziata in rosso nell'immagine compaiono questi elementi appena inseriti ed i vari collegamenti necessari per l'integrazione con il sistema esistente. Nello specifico, una nuova nota viene creata dal Tracker Agent tramite l'utilizzo dell'artefatto `NotesStream`, il quale se necessario, genera un evento interpellando `EventStream`. La generazione dell'evento dipende

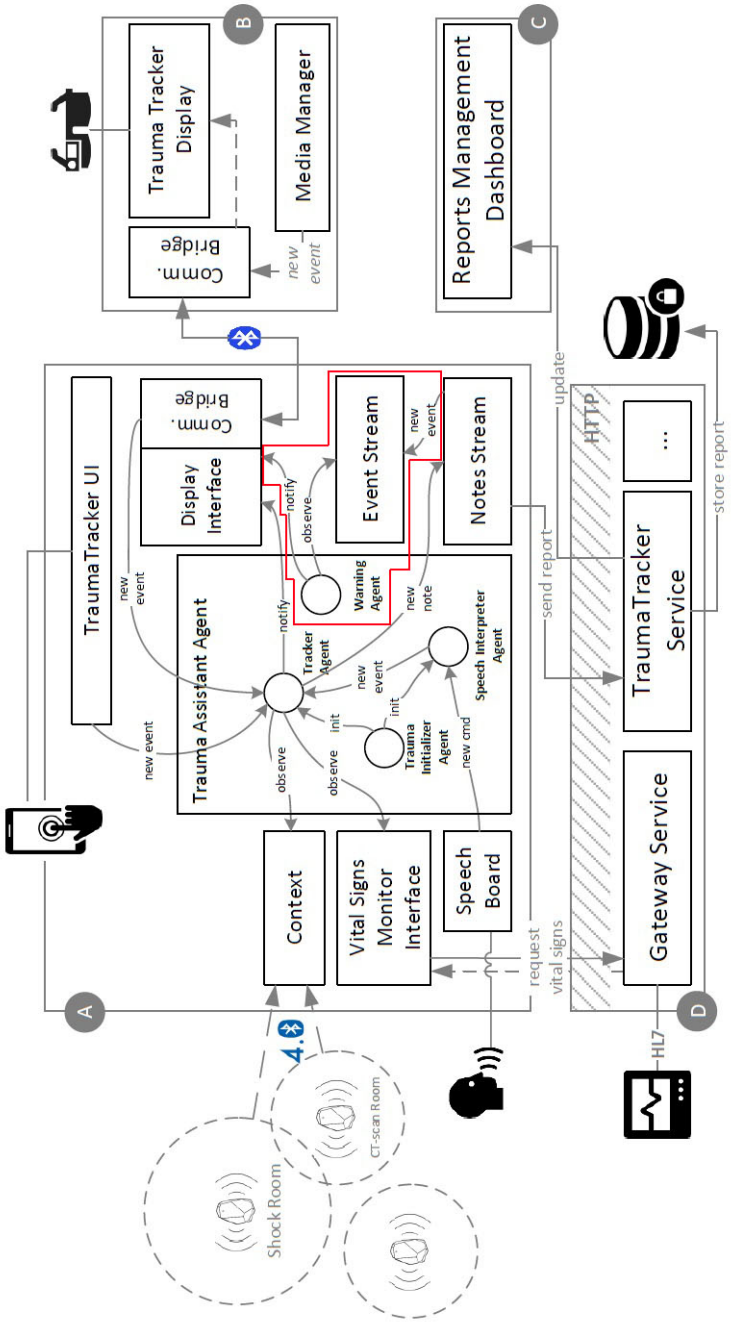


Figura 4.1: Architettura logica del sistema Trauma Tracker con particolare attenzione sugli agenti ed artefatti utilizzati. Le lettere A e B rappresentano il sottosistema TraumaLeader ed in particolare il Personal Assistan Agent, la lettera C rappresenta il sottositema Reports Management e la lettera D l'infrastruttura $GT^2$.

tal tipo di nota che è stata creata, in quanto solo alcune tipologie di note possono potenzialmente scaturire dei Warning. Il Warning Agent rimane in osservazione dell'`EventStream` e nel momento in cui percepisce un nuovo evento agisce manipolando i propri piani. Se il piano in esecuzione porta alla generazione di un Warning, allora questo viene notificato al display, il quale a sua volta può essere un semplice display video o degli smart glasses collegati tramite bluetooth.

### 4.1.1 Warning Agent

Le funzionalità richieste al Warning Agent sono state quindi implementate tramite una serie di piani predefiniti i quali vengono triggerati dagli eventi generati dal Trauma Leader nel momento in cui somministra nuovi farmaci al paziente, attua procedure sempre sul paziente, oppure nel momento in cui ci sono delle variazioni nei parametri vitali dell'assistito o avviene un cambiamento di stanza. Partendo dal risultato ottenuto in seguito all'analisi dei requisiti e del problema si è deciso di strutturare il Warning Agent con una serie di piani i quali si possono raggruppare come segue:

- Piani di servizio - piani i quali contengono funzionalità di supporto agli altri piani, ovvero funzionalità utili al corretto funzionamento dell'agente ed all'integrazione di altri piani.

- Piani tempo indipendenti - semplici piani indipendenti dagli eventi temporali. La loro struttura tipica è formata dalla dichiarazione del piano seguita subito dal piano avente la comparsa del Warning.

- Piani tempo dipendenti - piani i quali hanno al loro interno riferimenti temporali, oppure sono piani che vengono mandati in esecuzione in prossimità di un evento temporale.

Nella tabella seguente viene mostrata una rappresentazione strutturale dell'agente in termini di piani. All'interno dell'agente troviamo i goal iniziali ed i tre tipi di piani appena esplicitati. Alla ricezione di un evento l'agente combina autonomamente i diversi piani per raggiungere i propri goal.

| | Event | Body | Funzionalità |
|---|---|---|---|
| Piani di servizio | `new_event` | | Utilizzato per aggiungere il valore riguardante il tempo all'evento. |
| | `start` | | Utilizzato per la configurazione iniziale. |
| | `warning` | | Utilizzato per visualizzare o meno il testo del messaggio di allarme. |
| | `check...` | `warning(W,T)` | Utilizzati in determinate regole (numero 1, 5, 7 e 9) per effettuare controlli di contesto necessari alla comparsa del Warning. |
| Piani tempo dipendenti | `event(..,T)` | `set_trig(Time, checktrig)` | Utilizzati per le regole in cui è necessaria la gestione degli eventi temporali. |
| | `manage...` | `set_trig(Time, checktrig)` | Usati per la gestione delle regole numero 4 e 5 le quali vengono triggerate dallo stesso evento ma a tempi diversi hanno comportamenti diversi. |
| Piani tempo indipendenti | `event(..,T)` | `warning(W,T)` | Usati per la gestione delle regole tempo indipendenti. |

Nel codice riportato 4.1 viene mostrato un esempio di piano di servizio. Il piano `+!start` viene infatti utilizzato per la configurazione iniziale dell'agente. Serve per creare i vari artefatti necessari all'agente e li collega tra loro tramite la primitiva `linkArtifacts`.

```
+!start
    <-  makeArtifact("evStream","output.EventStream",[],IdEv);
        makeArtifact("ntStream","input.NoteStream",[],IdNt);
        makeArtifact("timer","util.Timer",[],IdTimer);
        linkArtifacts(IdNt,"eventStream",IdEv);
        .print("artifacts linked");
        focus(IdEv);
        focus(IdTimer).


+new_event(E)
    <-  timestamp(T); //operazione di Timer che prende il tempo
    corrente e lo associa alla nota
        +event(E, T);
        println("event",E,T).
```

Listing 4.1: Esempio pratico di piani di servizio.

Nel codice riportato 4.2 abbiamo `+new_event(E)`, un ulteriore piano di servizio con la funzione di accostare ad ogni evento le informazioni riguardanti al tempo. Dopodichè viene mostrata l'implementazione della regola numero 2, la quale è formata da piani tempo indipendenti. Il piano `+warning("..",T)` invece è un altro piano di servizio in quanto viene utilizzato per mostrare sul display il messaggio di Warning relativo alla regola in questione.

```
// Regola N.2
+event(drug("zero-negative",_),T) : zero_negative(3) & event(
    drug("tranex",_),_) & event(drug("fibrinogen",_),_)
    <-  +warning("activatePTM",T).

+event(drug("tranex",_),T) : zero_negative(3) & event(drug("
    fibrinogen",_),_)
    <-  +warning("activatePTM",T).

+event(drug("fibrinogen",_),T) : zero_negative(3) & event(drug("
    tranex",_),_)
        <-  +warning("activatePTM",T).

+warning("activatePTM", T)
    <-  displayWarning("Attivare Protocollo Trasfusione Massiva
    (PTM)?", T, "activatePTM").
```

Listing 4.2: Esempio pratico di piani tempo indipendneti e piani di servizio.

Nel codice riportato 4.3 si ha invece un esempio di piani tempo dipendenti, infatti per la gestione della regola numero 5 è stato necessario l'utilizzo di piani i quali al loro interno prevedono la gestione del fattore tempo. Anche per la regola numero 4, non considerata tempo dipendente per le motivazioni espresse nell'analisi del problema, si utilizzano comunque piani che considerano la gestione del tempo (`+!manageAdrenalineAdministration`) dato che da requisiti il Warning deve comparire e scomparire a tempo debito. La regola numero 3 invece è un'altro esempio di piano tempo indipendente collegato ad un piano di servizio.

```
1  // Regola N.3
2
3  +event(drug("thiopental",_),T)
4      <-  +warning("checkHypoTh",T).
5
6  +warning("checkHypoTh", T)
7      <-  displayWarning("Attenzione all'ipotensione!", T, "
       checkHypoTh").
8
9
10
11 // Regola N.4 e N.5
12
13 +event(procedureStarted("als"),T)
14     <-  +als(active);
15         !!manageAdrenalineAdministration;
16         !!managePleuralDecompressionCheck.
17
18 +event(procedureStopped("als"),_)
19     <- -als(active);
20        -warning("pleuralDecompression", T).
21
22 +!manageAdrenalineAdministration : als(active)
23     <-  setTrig(180000,"manageAA").    //180000
24
25 +!manageAdrenalineAdministration : not als(active)
26     <- -warning("adminAdrenaline", T).
27
28 +manageAA
29     <-  timestamp(T);
30         +warning("adminAdrenaline", T);
```

```
31          !manageAdrenalineAdministration.
32
33 +warning("adminAdrenaline", T)
34     <- displayWarning("Somministra 1mg di adrenalina.", T, "
       adminAdrenaline").
35
36 +!managePleuralDecompressionCheck
37     <-   setTrig(300000,"checkPleuralDecompression"). // 300000
38
39 +checkPleuralDecompression : not event(procedure("chest-tube"),_
       )
40     <-   timestamp(T);
41          +warning("pleuralDecompression",T).
42
43 +warning("pleuralDecompression", T)
44     <-   displayWarning("Decompressione pleurica?", T, "
       pleuralDecompression").
```

Listing 4.3: Esempio pratico di piani tempo dipendenti, indipendenti e di servizio.

Un'importante caratteristica da tenere presente sia a livello di progettazione sia a livello implementativo, è che per come è stato ideato Jason la sequenza in cui si dispongono i piani, all'interno del corpo dell'agente, non è irrilevante. Infatti alla ricezione di un evento, l'agente scorre tutti i suoi piani in ordine partendo dal primo e non appena trova le condizioni per attuare il piano, ovvero non appena trova il piano il quale evento e contesto corrisponde, lo esegue. Questo comportamento implica un'accurata riflessione sull'ordinamento delle regole che vanno implementate, ed a maggior ragione se in un secondo momento si vogliono aggiungere nuove regole e nuove funzionalità all'agente, bisogna pensare con esattezza al giusto luogo in cui inserire i nuovi piani. Questa caratteristica può essere considerata sia come pro che come contro, in quanto è vero che serve uno sforzo cognitivo maggiore per l'individuazione dell'ordine dei piani, però allo stesso tempo è una caratteristica sfruttabile dal punto di vista del coordinamento e della gestione dell'agente.

### 4.1.2 Elementi funzionali aggiuntivi

#### Event Stream

L'artefatto con cui il Warning Agent si interfaccia maggiormente è `Event Stream`. Questo ha il compito di generare eventi in seguito alla creazione di determinate note. Ovvero alcune delle note che si possono creare scaturiscono la creazione di un certo tipo di evento. Questi eventi sono tipo diverso in funzione della nota che li ha generati, per esempio una nota creata in seguito alla somministrazione di un farmaco triggererà la generazione di un evento di tipo farmaco, mentre analogamente una nota relativa al cambio stanza causerà la generazione di un evento relativo al cambio stanza, ecc.

#### Timer

`Timer` è un artefatto con il compito di scandire il tempo e generare eventi temporali in relazione allo scorrere di esso. Si è pensato inoltre di creare un ulteriore artefatto, `VirtualTimer`, in grado di modificare la percezione temporale dell'agente, così da riuscire ad ottimizzare i tempi nella fase di testing. All'interno dei piani del Warning Agent si ha quindi, dove richiesto in base alle regole, l'attesa di un evento temporale per far progredire il comportamento dell'agente stesso. Questo evento temporale può essere inviato o regolarmente seguendo il naturale scorrere del tempo tramite l'utilizzo dell'artefatto `Timer`, oppure anticipatamente tramite l'utilizzo dell'artefatto `VirtualTimer`, funzionalità che a regime non verrà utilizzata, ma che al contrario ottimizza il tempo quando si testano le varie funzionalità dell'applicazione.

In figura 4.2 viene mostrato il diagramma delle interazioni riguardanti gli elementi utilizzati per la realizzazione dell'estensione. In azzurro è rappresentato l'agente, mentre tutti gli artefatti sono contrassegnati con il colore giallo. A livello implementativo l'`EventStream` è stato realizzato utilizzando LinkedOperation, ovvero è composto da operazioni che possono essere chiamate da un altro artefatto. Queste sono operazioni indipendenti, contrassegnate dalla notazione `@LINK`, le quali, una volta triggerate, hanno la stessa esecuzione di normali operazioni di classici artefatti. L'unica differenza è che l'evento generato da un'operazione linkata è reso osservabile da un agente solo usando o osservando l'artefatto che ha attivato l'esecuzione dell'operazione, oppure in caso di catene dove un agente A esegue

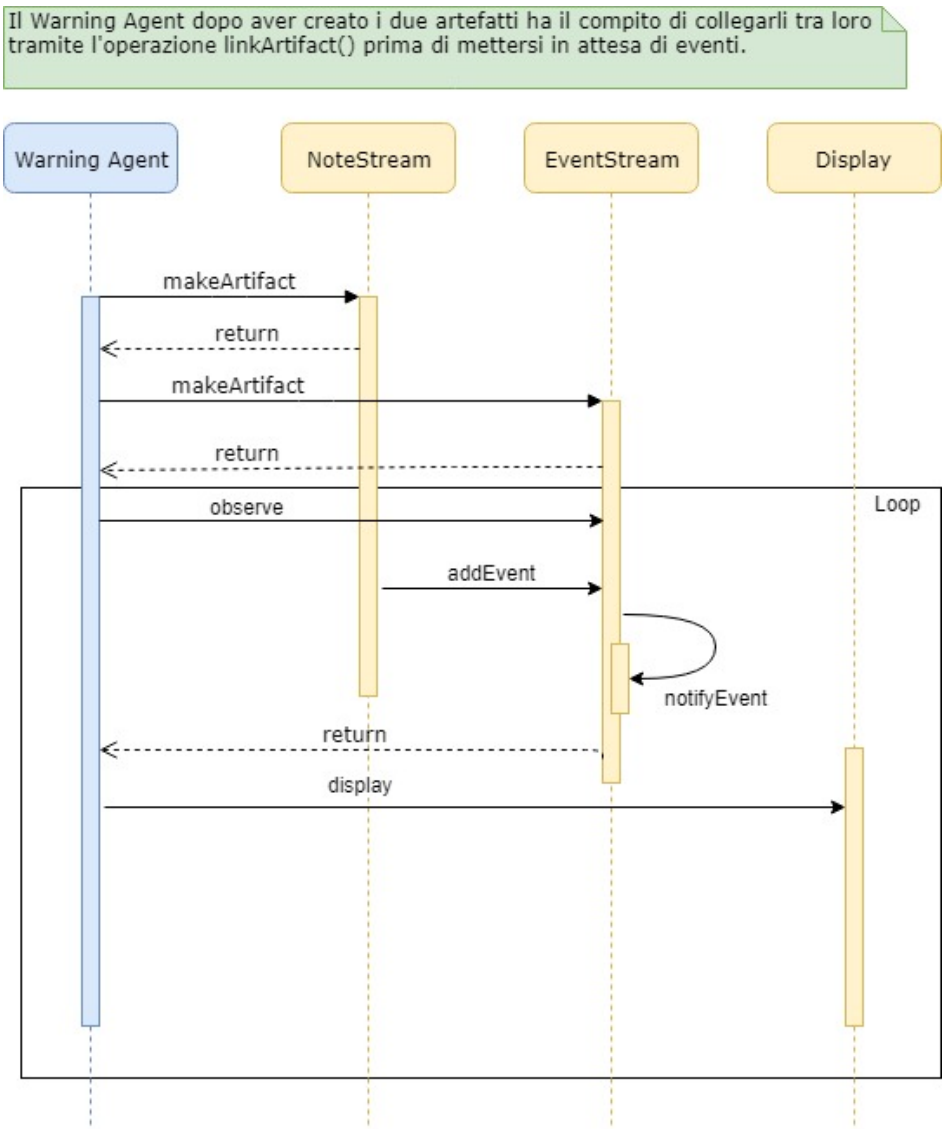


Figura 4.2: Diagramma delle interazioni per quanto riguarda gli elementi inseriti per la gestione dei Warning.

un'operazione su un artefatto, il quale linka l'operazione di un altro artefatto B, il quale a sua volta linka un'ulteriore operazione di un artefatto C, tutti gli eventi osservabili generati da B e C tramite operazioni linkate sono osservabili dall'agente A. Nello specifico di Trauma Tracker le operazioni linkate di `EventStream` hanno tutte come fine la generazione di eventi. La chiamata dell'operazione dall'artefatto di collegamento, in questo caso `NoteStream`, viene eseguita utilizzando la primitiva `execLinkedOp`, mentre il collegamento tra i vari artefatti viene fatto tramite un agente. Il Warning Agent infatti per prima cosa genera tutti gli artefatti necessari (`EventStream`, `NoteStream` e `Display`), dopodichè collega i primi due tramite l'operazione linkArtifacts, infine fa il focus sull'ultimo artefatto e si mette in attesa di eventi. Nel momento in cui viene generata una nota, il `NoteStream` chiama la funzione di `EventStream` relativa al tipo di nota creata, la quale genererà l'evento desiderato. Il Warning Agent percepisce l'evento appena creato e manipola i suoi piani per gestirlo giungendo alla visualizzazione del Warning sul display.

## 4.2 Sviluppo dell'estensione per la gestione dei Warning

Per la realizzazione dell'estensione riguardante i Warning si è deciso di creare un "simulatore di Trauma Tracker", ovvero una versione molto ridotta dell'applicazione, comprendente solo le parti del sistema essenziali per l'aggancio dell'estensione. Si è analizzato il sistema per intero individuando quali sono le funzionalità di interesse per la gestione dei Warning e si è cercato di estrapolarle ottenendo così il punto di partenza dal quale è partita poi l'implementazione dell'estensione. Utilizzando questo approccio, l'estensione creata è perfettamente integrabile con il sistema completo senza dover modificare nulla dal punto di implementativo. L'unico refactoring necessario sarà l'eliminazione dell'interfaccia grafica, che è stata creata per agevolare sia la fase di debugging che quella di testing, ed il collegamento al sistema esistente.

Il simulatore presenta infatti un'interfaccia grafica che permette la generazione di note sia manualmente che automaticamente. È infatti possibile sia generare una nota per volta inserendo manualmente i parametri richiesti (diversi in base al tipo di nota), sia generare delle routine preimpostate di

note le quali vanno in esecuzione con parametri standard tramite la pressione di specifici pulsanti. La note che sono state contemplate, durante la creazione del simulatore, sono solamente quelle che scaturiscono la creazione di eventi i quali a loro volta possono potenzialmente generare Warning. Durante l'ideazione e creazione del simulatore si è rimasti in linea con l'implementazione attuale del sistema per cercare di semplificare e ridurre al minimo il lavoro di refactoring da effettuare sul codice nel momento dello sgancio dell'interfaccia grafica per l'input ed il riallacciamento con il sistema in se per se.

### 4.2.1 Interfacciamento con il sistema

Mandando in esecuzione il simulatore vengono generate, oltre alla console del MAS, due finestre, una per l'interfaccia utente di input dal nome GUI, mentre l'altra per l'output, ovvero il Warning Display 4.3. L'interfaccia grafica di input è divisa concettualmente in due parti, la parte superiore viene utilizzata per l'inserimento manuale di un evento, mentre la parte inferiore presenta tre pulsanti per l'esecuzione di routine di eventi preimpostati. Per l'inserimento di un evento singolo viene richiesto, in prima istanza, il tipo di evento che si vuole generare e l'utente che lo genera. La selezione del tipo di evento avviene tramite l'utilizzo del menù a tendina, mentre l'utente tendenzialmente è il Trauma Leader, valore impostato di default, ma può essere cambiato manualmente. Una volta premuto il tasto ok compare una nuova finestra per l'inserimento dei parametri4.4. I parametri richiesti sono diversi in base al tipo di evento da inserire, di conseguenza la finestra che compare ha elementi diversi. Per esempio come si può vedere nell'immagine all'inserimento di un DrugEvent, ovvero alla somministrazione di un farmaco, viene richiesto il tipo di farmaco somministrato selezionabile tramite menù a tendina e la quantità4.5 4.6. Se il campo quantità non viene riempito l'evento non viene creato. Per quanto riguarda invece gli eventi di tipo RoomOutEvent nella finestra di inserimento parametri si ha un campo per inserire manualmente la stanza, ed una checkbox che rappresenta la "frattura esposta" 4.7. In realtà la presenza di frattura esposta fa parte delle condizioni generali del paziente nel momento in cui arriva al pronto soccorso. Siccome è un parametro che la sua presenza o meno implica la comparsa di un Warning, si è ritenuto necessario dare la possibilità di modifica di tale

parametro tramite interfaccia grafica, al fine di testare completamente le funzionalità dell'estensione. Se si vuole generare un evento del tipo TimeDependentProcedureNote 4.8, ovvero un evento riguardante una procedura dipendente dal tempo, nella finestra di inserimento dei parametri bisogna selezionare tramite un menù a tendina la specifica procedura ed inoltre si trova un campo modificabile sottostante che di default riporta la scrittura "start/end". Se si vuole indicare l'inizio di una procedura bisogna modificare il campo lasciando solamente la scritta "start", mentre al contrario se si vuole indicare la terminazione di una procedura bisogna modificare il campo lasciando solamente la scritta "end". Terminato l'inserimento dei parametri si viene sempre rimandati alla finestra iniziale dal nome GUI così da poter inserire un nuovo evento.

I tre pulsanti situati nella parte bassa della GUI sono associati a delle routine preimpostate di eventi 4.9 le quali sono state create per snellire e velocizzare la fase di testing. I parametri per i diversi eventi che vengono generati sono preimpostati direttamente dal codice e non c'è possibilità di modificarli tramite la grafica.

Per quanto riguarda il Warning Display, ovvero la finestra in output dove compaiono i Warning, anch'essa si può suddividere in due parti. La parte di sinistra è il vero e proprio display dove leggere i Warning, mentre nella parte destra, inizialmente vuota, compariranno diversi pulsanti, uno relativo ad ogni Warning generato, da utilizzare per l'eliminazione dei messaggi stessi4.10. In base alle esigenze questo display può assumere diverse sembianze, può essere sia un semplice monitor a muro, un tablet gestito da un componente del team, oppure degli smart glasses indossati dal Trauma Leader.

## 4.3 Fase di test

Come accennato nei capitoli precedenti la gestione del tempo è gestita tramite gli artefatti `Timer` e `VirtualTimer`. Quest'ultimo è stato realizzato esclusivamente per agevolare la fase di testing e debugging, in quanto è in grado di fornire un tempo virtuale, ovvero può far si che l'agente percepisca una versione distorta del tempo. Grazie a questo artefatto si può infatti controllare il funzionamento delle regole tempo dipendenti in maniera più efficiente. Per esempio per la regola N.7 non è necessario attendere 15

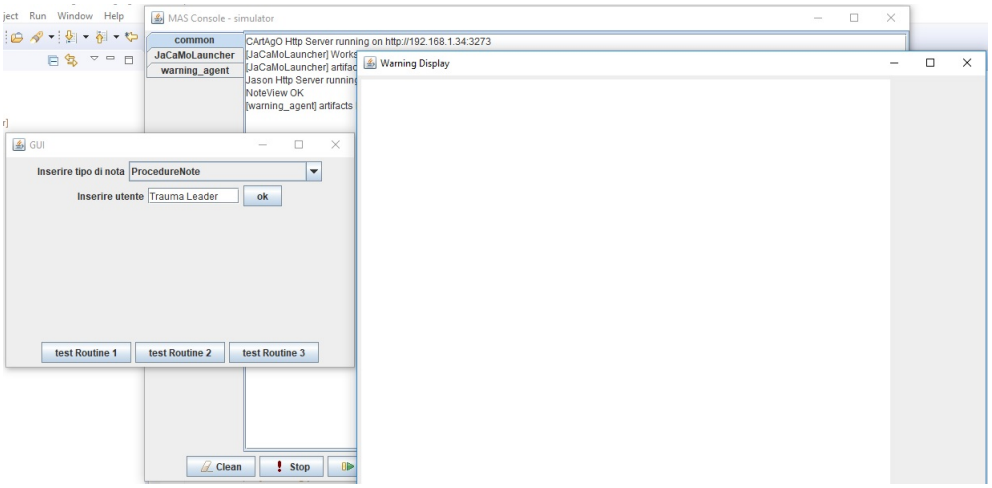


Figura 4.3: Situazione iniziale all'avvio del simulatore.

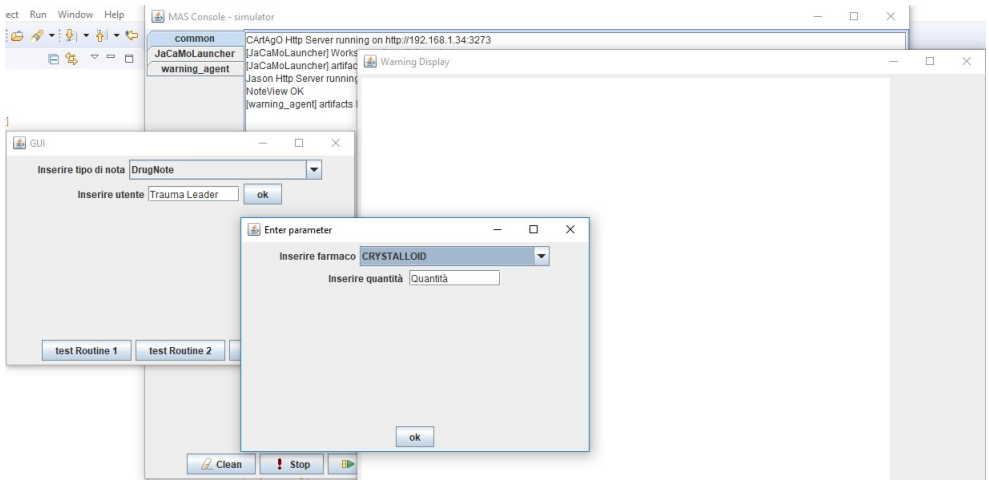


Figura 4.4: Inserimento manuale di un evento di tipo DrugNote con rispettiva finestra per l'inserimento dei parametri.

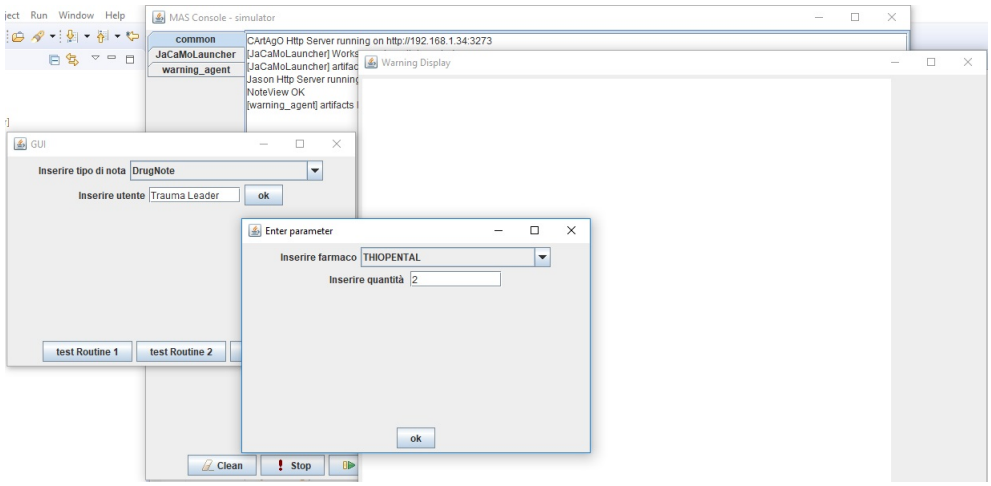


Figura 4.5: Esempio di inserimento di una nota relativa alla somministrazione 2 unità del farmaco Tiopentolo.

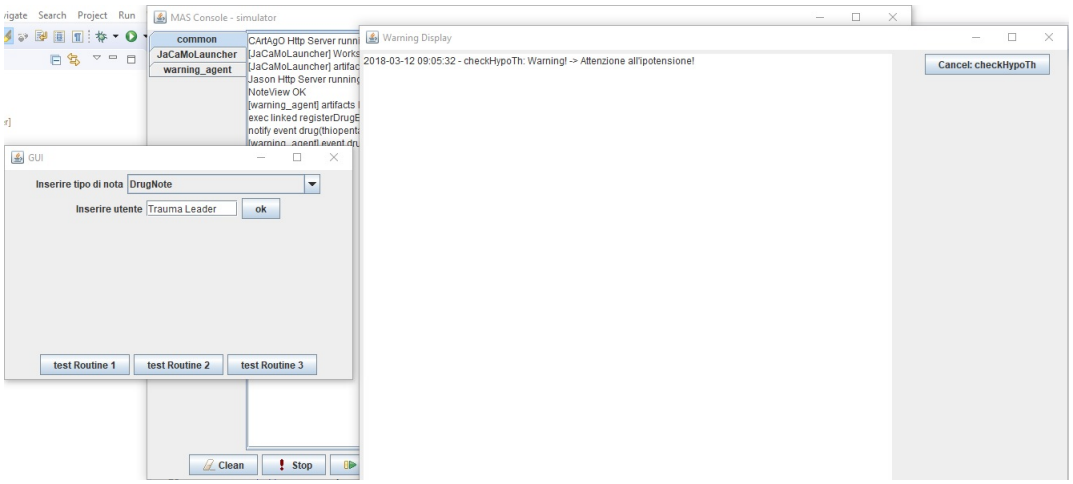


Figura 4.6: Risultato dell'inserimento della nota dell'immagine precedente.

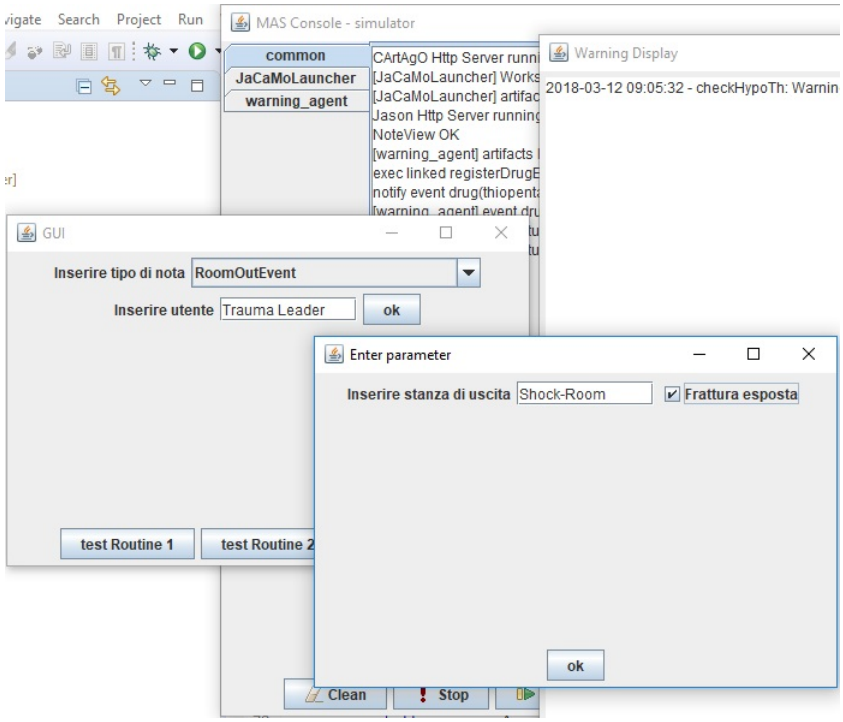


Figura 4.7: Visualizzazione della finestra per l'inserimento dei parametri relativa ad una nota ti tipo RoomOutEvent.

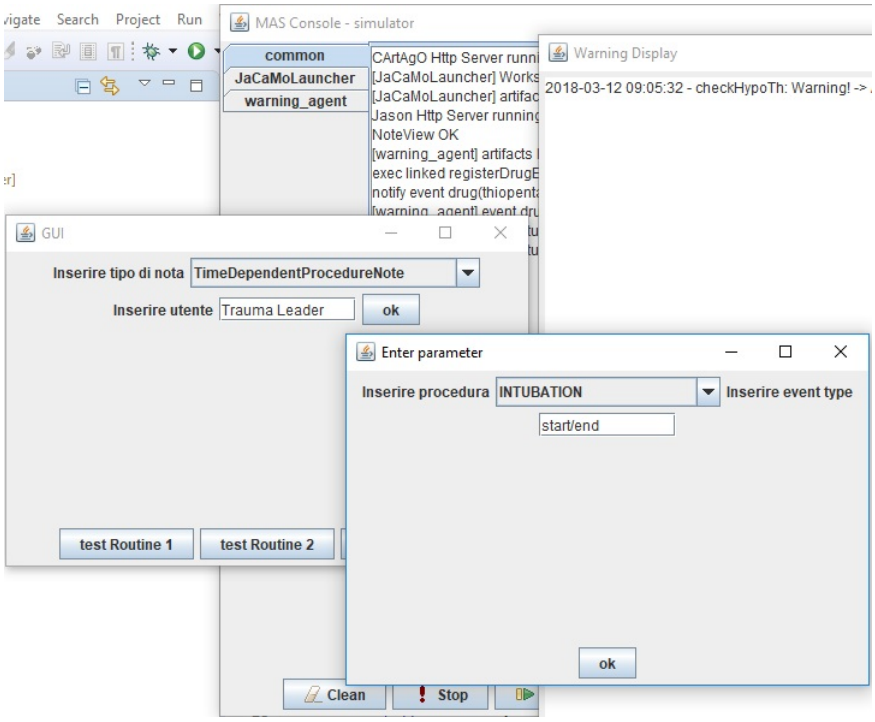


Figura 4.8: Visualizzazione della finestra per l'inserimento dei parametri relativa ad una nota ti tipo TimeDependentProcedureEvent.

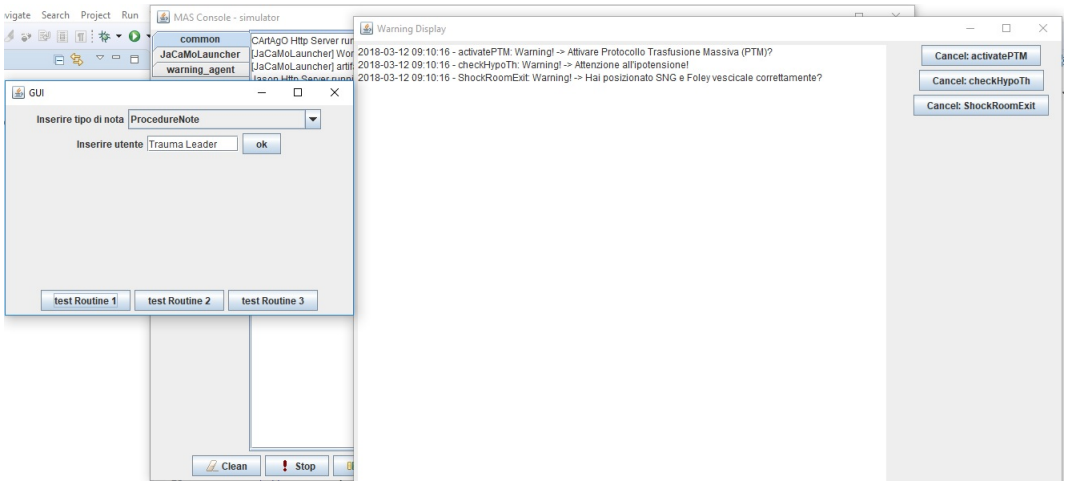


Figura 4.9: Risultato dell'esecuzione della routine di eventi relativa al primo pulsante.

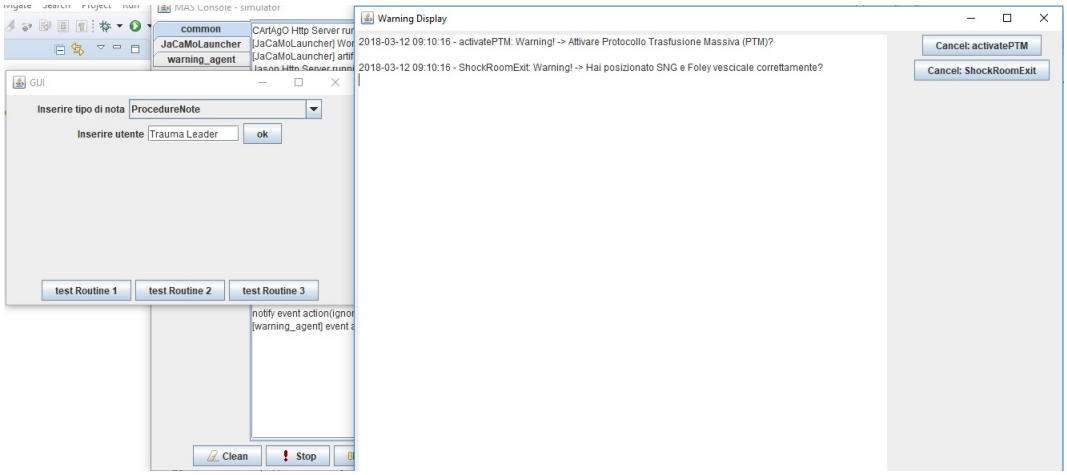


Figura 4.10: Dalla situazione dell'immagine precedente premendo il pulsante per l'eliminazione del secondo messaggio si ottiene la sparizione del Warning e del relativo pulsante.

min dalla ricezione dell'evento sulla procedura, prima della comparsa del Warning, ma basta che l'artefatto invii l'evento temporale dopo qualche secondo. Essendo appunto la gestione delle regole tempo dipendenti gestita tramite la ricezione di un evento temporale, l'implementazione dell'agente non cambia, ma cambia solo il momento in cui l'artefatto genera l'evento (nell'esempio precedente 1 secondo invece di 15 min). Se si fosse gestito il tempo direttamente dall'agente, per esempio con l'utilizzo di `wait()` o funzioni affini, per ottenere gli stessi benefici si sarebbe dovuto intervenire sul codice dell'agente, andando ogni volta a modificare il valore che identificava il tempo di attesa. Si è pensato quindi di ottimizzare il tutto modellando il tempo tramite eventi temporali, ottenendo quindi un sistema più solido e pulito.

In aggiunta a tale modellazione, è stata creata la possibilità di generare delle routine di eventi, comprendenti anche gli eventi temporali, per facilitare la verifica del sistema. Tramite la console del MAS si possono verificare ogni singolo evento generato, correlato con le informazioni riguardanti i parametri in ingresso ed il tempo. Confrontando tali informazioni sulla console con il Warning Display si può agevolmente verificare l'effettivo funzionamento del sistema.

Per quanto riguarda la fase di test vera e propria inizialmente si sono

testate tutte le regole una ad una utilizzando l'apposita interfaccia grafica creata per l'inserimento dei dati, e controllando il corretto funzionamento del sistema visualizzando se veniva mostrato sul display il giusto Warning, relativo alla nota creata. Una volta terminata questa prima fase di test si è passato ad un test generale del sistema. A regime infatti si avrà uno stream di eventi continuo composto da serie di eventi eterogenei alcuni dei quali scaturiranno la generazione di Warning. Per testare agevolmente questa situazione si è deciso di generare routine di eventi preconfezionate, così da confrontare agevolmente l'output ottenuto con quello atteso. A tale proposito si è predisposta l'interfaccia grafica di pulsanti appositi (test Routine 1, test Routine 2, test Routine 3). Avendo due diversi tipi di regole, in base alla tempo dipendenza, si è pensato di simulare tre situazioni differenti che possono avere luogo. La prima situazione è generata solamente da regole tempo indipendente, la seconda, al contrario, è creata da regole tempo dipendenti, mentre la terza è la situazione in cui si ha un insieme eterogeneo delle diverse regole. Ogni pulsante è stato quindi assegnato ad una situazione diversa. Nello specifico, come mostrato nel codice 4.4, per il primo pulsante si è creata una serie di eventi per far partire la regola numero 2, seguita dalla 3 ed infine la 8, quindi tutte regole tempo indipendenti.

```
1  test1.addActionListener(new ActionListener() {
2      @Override
3      public void actionPerformed(ActionEvent ev) {
4          stream.beginExternalSession();
5
6          //regola n2
7          stream.addEvent("registerDrugEvent", Drug.ZERO_NEGATIVE,
       (double)3);
8          stream.addEvent("registerDrugEvent", Drug.FIBRINOGEN, (
      double)1);
9          stream.addEvent("registerDrugEvent", Drug.
      TRANEXAMIC_ACID, (double)1);
10
11         //regola n3
12         stream.addEvent("registerDrugEvent", Drug.THIOPENTAL, (
      double)1);
13
14         //regola n8
15         stream.addEvent("registerRoomExitEvent", "Shock-Room");
16
17         stream.endExternalSession(true);
```

```
18      }
19  });
```

Listing 4.4: Codice della routine di eventi, con i relativi parametri utilizzati, collegata al pulsante "test Routine 1".

Nel secondo pulsante, codice 4.5, si è creata una routine composta solamente da regole tempo dipendenti ovvero la regola numero 1, seguita dalla regola 5 ed infine dalla 9.

```
1  test2.addActionListener(new ActionListener() {
2      @Override
3      public void actionPerformed(ActionEvent ev) {
4          stream.beginExternalSession();
5
6          //regola n1 dopo 5 min da zero negativo compare il
       warning se non vengono somministrati fibrogeno e acido
       tranex.
7          stream.addEvent("registerDrugEvent", Drug.ZERO_NEGATIVE,
        (double)3);
8
9          //regola n5  dopo 5 min da als start compare il warning
       se non si avvia prima la decompressione pleurica
10         stream.addEvent("registerTimeDependentProcedureEvent",
       Procedure.ALS, "start");
11
12         //regfola n9 dopo 5 min dall'inizio dell'intubazione
       compare il warning se non si hanno info sull'EtCO2
13         stream.addEvent("registerTimeDependentProcedureEvent",
       Procedure.INTUBATION, "start");
14
15         stream.endExternalSession(true);
16     }
17 });
```

Listing 4.5: Codice della routine di eventi, con i relativi parametri utilizzati, collegata al pulsante "test Routine 2".

Infine nel terzo pulsante, codice 4.6, si è impostata una routine di eventi composta dalla regola numero 4 e 5 che vengono triggherate dallo stesso evento, seguite dalla regola numero 6, numero 1, numero 7, numero 10 ed infine la numero 11, creando così una routine eterogenea composta sia da regole tempo dipendenti che tempo indipendenti.

```
1  test3.addActionListener(new ActionListener() {
2      @Override
```

```
3       public void actionPerformed(ActionEvent ev) {
4           stream.beginExternalSession();
5
6           //regola n4 e n5
7           stream.addEvent("registerTimeDependentProcedureEvent",
    Procedure.ALS, "start");
8
9           //regola n6
10          stream.addEvent("changeFracture", true);
11          stream.addEvent("registerRoomExitEvent", "Shock-Room");
12
13          //regola n1
14          stream.addEvent("registerDrugEvent", Drug.ZERO_NEGATIVE,
     (double)3);
15
16          //regola n7
17          stream.addEvent("registerTimeDependentProcedureEvent",
    Procedure.TOURNIQUET, "start");
18
19          //regola n10
20
21          stream.addEvent("registerVitalSignEvent", VitalSign.
    SPO2_VALUE, 80);
22
23          //regola n11
24          stream.addEvent("registerTimeDependentProcedureEvent",
    Procedure.INFUSER, "start");
25          stream.addEvent("registerVitalSignEvent", VitalSign.
    BLOOD_PRESSURE_VALUE, 120);
26
27          stream.endExternalSession(true);
28      }
29  });
```

Listing 4.6: Codice della routine di eventi, con i relativi parametri utilizzati, collegata al pulsante "test Routine 3".

# Conclusioni

Grazie all'estensione progettata ed implementata durante lo svolgimento della tesi si è creato un prototipo di estensione di Trauma Tracker in grado di offrire il nuovo livello di assistenza. Questo sistema, insieme a tutti gli altri agenti intelligenti ideati per lavorare all'interno dello stesso dominio applicativo, ovviamente non vuole sostituire il medico nel suo lavoro, in quanto le decisioni vengono comunque prese interamente dall'equipe del Trauma Center, ma vuole solamente semplificare il suo lavoro sollecitando la sua attenzione nel momento in cui si stiano verificando delle situazioni pericolose per la vita del paziente.

Visto che gli obiettivi della tesi erano la progettazione e lo sviluppo di un agente BDI per integrare, all'interno di Trauma Tracker, le funzionalità di supporto ai medici tramite Warning e lo studio del modello BDI per verificarne la sua effettiva efficacia applicato al dominio di interesse, si può affermare che:
PRO

- l'utilizzo di un linguaggio di alto livello semplifica sicuramente la progettazione e la scrittura dei programmi perchè più vicino al linguaggio degli esseri umani e quindi più facilmente comprensibile. A maggior ragione in un dominio complesso, come quello trattato da Trauma Tracker, il vantaggio apportato dall'utilizzo di tali linguaggi è più evidente.

- l'utilizzo dell'Agent Oriented Programming e del modello BDI porta un forte beneficio, dal punto di vista progettuale, in quanto, data la complessità del caso di studi, ci permette di focalizzare l'attenzione sul problema in sé tralasciando gli aspetti di controllo. Utilizzando un diverso approccio, che potrebbe essere l'utilizzo del paradigma Object Oriented, si sarebbe avuto lo stesso potere espressivo, però sarebbe

stato necessario un ulteriore sistema per la gestione della concorrenza, il quale invece è già incapsulato all'interno della concezione di agente cognitivo.

- avendo utilizzato Jason come linguaggio per questo progetto, si è dovuto tenere in considerazione il fatto che l'ordine dei piani che compongono l'agente è altamente significativo visto la caratteristica valutazione sequenziale dei piani. Questa peculiarità può essere sfruttata a favore del progettista per il controllo parziale sul comportamento e gestione dell'agente.

CONTRO

- la caratteristica riguardante l'organizzazione dei piani rende necessaria una conoscenza approfondita della struttura e composizione dell'agente, soprattutto nel momento in cui si vogliano implementare nuove regole con il conseguente inserimento di nuovi piani;

- un programma agent based, caratterizzato dall'autonomia d'azione dell'agente e dalla sua indipendenza, può risultare più difficile da controllare e gestire, dal punto di vista dello sviluppatore, rispetto ad un programma scritto utilizzando un paradigma di programmazione procedurale o ad oggetti.

Allo stato attuale il sistema è funzionante e gli obiettivi preposti sono stati raggiunti, portando in luce diversi sviluppi futuri. Il simulatore di Trauma Tracker che è stato creato può essere sicuramente migliorato ed aggiornato al fine di poter compiere simulazioni più efficaci. Sicuramente con l'utilizzo reale del sistema compariranno situazioni che non sono state adeguatamente considerate. Il simulatore può essere quindi adattato per testare e gestire più accuratamente queste specifiche situazioni critiche. Un'ulteriore sviluppo futuro deriva dall'ampiezza del dominio, ovvero se viene richiesta l'implementazione di regole aggiuntive le quali non rientrano nella classificazione esistente al momento (regole tempo dipendenti / regole tempo indipendenti). Questa situazione implicherebbe uno studio approfondito della letteratura medica ala ricerca di una classificazione più adeguata in grado di racchiudere anche le nuove regole. Come sviluppo futuro troviamo anche la validazione sul campo. Considerando l'utilizzo del sistema esteso da parte dei medici in casi controllati, mettendo in campo

un percorso di sperimentazione, si potrebbero apportare migliorie al sistema attuale dal punto di vista delle performance, riuscendo ad ottenere un rapporto più fedele tra il sistema di simulazione ed il sistema reale.

Ormai è chiaro a tutti che la tecnologia sta prendendo una posizione sempre più rilevante all'interno delle nostre vite. Si è partiti da semplici computer utilizzati per velocizzare il nostro lavoro che grazie all'evoluzione tecnologica si sono trasformati in potentissimi calcolatori sempre più piccoli ed intelligenti, anche capaci di ragionare. Con le loro peculiarità si sono resi sempre più indispensabili diventando parte integrante della vita dell'uomo, aggiudicandosi così un posto tra i bisogni primari dell'essere umano.

# Ringraziamenti

Ringrazio i miei genitori per avermi dato la possibilità di intraprendere questo percorso di studi con serenità.

Ringrazio il Patato per avermi spronato a continuare gli studi dopo la laurea triennale e per avermi supportato e sopportato lungo tutto il cammino.

Desidero ringraziare il mio relatore Alessandro Ricci e co-relatore Angelo Croatti sempre disponibili ad ascoltare le mie esigenze fornendo un ottimo supporto e preziosi consigli per la realizzazione della tesi.

Un ringraziamento va a tutti i miei compagni di corso per aver alleggerito il clima di questi ultimi due anni di università.

Ringrazio infine tutti i miei amici per il sostegno ottenuto nei diversi momenti.

# Bibliografia


[1] Adrenalina. http://medicinaurgenza.com/adrenalina-adrenalina-f-1-mi-1-mgml/.

[2] Als (advanced life support). http://www.treccani.it/enciclopedia/als_(Dizionario-di-Medicina)/.

[3] Andoid speechrecognizer. https://developer.android.com/reference/android/speech/SpeechRecognizer.html.

[4] Cartago common artifact infrastructure for agents open environments. http://cartago.sourceforge.net/.

[5] Catetere vescicale: la gestione. http://www.abilitychannel.tv/catetere-vescicale/.

[6] Fibrinogeno alto e basso: i valori di riferimento. https://www.analisidelsangue.net/fibrinogeno-alto-basso/.

[7] Fratture chiuse ed esposte. https://www.tesionline.it/v2/appunto-sub.jsp?p=3&id=748.

[8] Gestire sng, peg, pej. http://www.ausl.rn.it/Materiali/infermieri-per-il-cittadino/cura_paziente/gestire-sng-peg-pej.html.

[9] Gruppi sanguigni. http://www.donatoridisangue.it/site/index.php%3Foption%3Dcom_content%26view%3Darticle%26id%3D64%26Itemid%3D69.

[10] Health level seven international. http://www.hl7.org/implement/standards/index.cfm?ref=nav.

[11] Hypotension. https://www.nhlbi.nih.gov/health-topics/hypotension.

[12] Il drenaggio toracico in emergenza sanitaria. https://medest118.com/2013/10/01/il-drenaggio-toracico-in-emergenza-sanitaria/.

[13] Jacamo project. http://jacamo.sourceforge.net/.

[14] Massive transfusion and massive transfusion protocol. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4260305/.

[15] The moise organisation oriented programming framework. http://moise.sourceforge.net/.

[16] Mongo db. https://www.mongodb.com/.

[17] Monitoraggio dell'etco2 durante una cpr: previsione dell'outcome. http://www.soccorritori.ch/?p=2066.

[18] Pressione sistolica o pressione massima tratto. http://www.my-personaltrainer.it/ipertensione/pressione-sistolica.html.

[19] Proximity marketing e beacon. http://www.beacon.it/.

[20] Reboa: Resuscitative endovascular balloon occlusion of the aorta. http://www.traumaready.com/reboa/#.WnQzPYjOWUk.

[21] Saturimetria. https://www.dottori.it/salute/saturimetria.

[22] Stanford university - michael e. bratman. https://philosophy.stanford.edu/people/michael-e-bratman.

[23] The tourniquet in surgery. http://www.boneandjoint.org.uk/content/jbjsbr/44-B/4/937.full.pdf.

[24] Tranexamic acid injection. https://www.drugs.com/pro/tranexamic-acid-injection.html.

[25] Vert.x. http://vertx.io/.

[26] Vuzix: View the future. `https://www.vuzix.com/Products/m100-smart-glasses`.

[27] Who model list (revised march 2009). `http://apps.who.int/iris/bitstream/10665/70642/1/a95055_eng.pdf`.

[28] Rafael H Bordini and Jomi F Hübner. Jason–a java-based interpreter for an extended version of agentspeak. 2007.

[29] Rafael H. Bordini, Jomi Fred Hubner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8 of *Wiley Series in agent technology*. John Wiley & Sons, 2007.

[30] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[31] Angelo Croatti, Sara Montagna, Alessandroangelo croatti Ricci, Emiliano Gamberini, Vittorio Albarello, and Vanni Agnoletti. Personal medical digital assistant agents for trauma tracking and assistance. 2017.

[32] Angelo Croatti, Sara Montagna, and Alessandro Ricci. A personal medical digital assistant agent for supporting human operators in emergency scenarios. In *Agents and Multi-Agent Systems for Health Care*, pages 59–75. Springer, 2017.

[33] Angelo Croatti and Alessandro Ricci. Tracking and assisting activities in trauma management: The traumatracker case. 2017.

[34] Michael P Georgeff and Amy L Lansky. Reactive reasoning and planning. In *AAAI*, volume 87, pages 677–682, 1987.

[35] Kevin D Johnston and Mridula R Rai. Conscious sedation for awake fibreoptic intubation: a review of the literature. *Canadian Journal of Anesthesia/Journal canadien d'anesthésie*, 60(6):584–599, 2013.

[36] Nikola Kasabov and Robert Kozma. Introduction: Hybrid intelligent adaptive systems. *International Journal of Intelligent Systems*, 13(6):453–454, 1998.

[37] Pattie Maes. Agents that reduce work and information overload. In *Readings in Human–Computer Interaction*, pages 811–821. Elsevier, 1995.

[38] Donald A Norman. How might people interact with agents. *Communications of the ACM*, 37(7):68–71, 1994.

[39] Anand S Rao. Agentspeak (l): Bdi agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer, 1996.

[40] G Sanson, G Nardi, E De Blasio, S Di Bartolomeo, C Moroni, and C Serantoni. *Prehospital Trauma Care-Approccio e trattamento al traumatizzato in fase preospedaliera e nella prima fase intraospedaliera.* ItalianResuscitationCouncil, 2007.

[41] Andrea Santi, Marco Guidi, and Alessandro Ricci. Jaca-android: An agent-based platform for building smart mobile applications. In *International Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems*, pages 95–114. Springer, 2010.

[42] Michael Wooldridge. *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[43] Michael Wooldridge and Nicholas R. Jennings. Atal: A five year retrospective. `http://mas.cs.umass.edu/atal/workshops/index.html`.

[44] Neil Yorke-Smith, Shahin Saadati, Karen L Myers, and David N Morley. The design of a proactive personal agent for task management. *International Journal on Artificial Intelligence Tools*, 21(01):1250004, 2012.