MC-SHOP

SISTEMA DI PROFILAZIONE CLIENTI

Specifica dei Requisiti

Modena, 15 Agosto 2020

Io sottoscritto Stefano Lugli, matricola numero 118305, ***DICHIARO*** *che questo elaborato è frutto del mio personale lavoro, svolto sostanzialmente in maniera individuale ed autonoma*

ABSTRACT

L'esperienza del cliente in un punto vendita fisico non è fondamentale solo per quanto riguarda la vendita diretta di beni, ma anche per garantire una certa soddisfazione capace di aumentare la buona reputazione del negozio ed esercitare un appeal aggiuntivo sulla potenziale clientela.

Un ambiente di vendita può essere rappresentato come un semplice spazio all'interno del quale i clienti hanno la possibilità di muoversi, interagire ed effettuare acquisti; l'esempio più banale è la tipica boutique di un centro commerciale. Ovviamente, questo progetto mira anche ad ambienti più complessi.

Il comfort può prevenire da diversi fonti, definibili come comfort aspects, adattabili ai gusti dei consumatori, come musica, profumi e colore dei led. Proponiamo quindi un approccio "user-aware" capace di auto configurarsi dinamicamente considerando i comfort-aspects. Introdurremo, inoltre, un modello di contesto personale dell'utente, il quale può essere utilizzato per ottenere informazioni riguardo allo stesso e dunque effettuare le decisioni migliori.

Il sistema di decisione è basato su un'architettura a micro servizi, la quale garantisce modularità e flessibilità.

1. Introduzione

Con la seguente sezione si cercherà di dare una visione completa per quanto riguarda il documento di Specifica dei Requisiti. La struttura è quella suggerita dallo standard IEEE 830-1998, noto come Software *Requirements Specification Standard*.

1.1. OBBIETTIVO

L'obbiettivo di questo documento è la descrizione formale dei requisiti, del funzionamento e della struttura del progetto commissionato dal professore Bicocchi per il corso di Programmazione ad Oggetti (DIEF, UNIMORE). Illustrerà quindi le informazioni sui requisiti raccolte durante la fase di elicitation svolta in presenza del professore stesso, riguardanti l'idea di Stephan Boese (Gk Software, Germania, sboese@gksoftware.it) e Giacomo Cabri (Università di Modena e Reggio Emilia, Modena, Italia, giacomo.cabri@unimore.it).

1.2. CAMPO D'APPLICAZIONE

Il documento tratterà della creazione di un software volto all'automazione di un punto vendita in maniera tale da adattarlo alle preferenze dei clienti, in maniera dinamica, rendendo lo "shopping" più piacevole. Quando le persone entrano in negozio sono spinte ad effettuare acquisti da diversi fattori, in primis i gusti personali e la qualità della merce in esposizione. Ma esclusi questi due punti fondamentali, entrano in gioco altre forze, principalmente basate sul comfort durante la fase di acquisto: la musica, i colori dei led, il profumo, cosa mostrano eventuali monitor e la pubblicità esposta; i cosiddetti *comfort-aspects*. Manipolando questi fattori possiamo prima di tutto incoraggiare i consumatori ad effettuare più acquisti e a frequentare il punto vendita più spesso, in secondo luogo ad attrarne di nuovi grazie alla buona reputazione diffusa dai primi.

Ovviamente, i *comfort aspects* non sono una novità e già ora i negozianti fanno del proprio meglio per renderli più efficaci possibile. Ma ovviamente, come ogni lavoro "manuale", richiede una grande esperienza ed una capacità critica che non tutti possiedono. Come si vide a fine anni '90 e inizio anni 2000 una grande rivoluzione nel mondo degli acquisti online traslando da un modello a "recensioni

d’esperti” ad uno a suggerimenti basato su ciò che gli utenti effettivamente apprezzano, introdotto in primis da Amazon, cercheremo di creare le basi per una rivoluzione nella gestione automatizzata dei punti vendita anche nel mondo reale.

Questo progetto non ha ancora assunto un nome ufficiale, quindi vi si farà riferimento utilizzando il codename scelto in fase di prototipazione, ossia *MC-SHOP*.

È necessario sottolineare fin da subito come non ci si preoccuperà degli aspetti psicologici, come la scelta del brano migliore da riprodurre in una determinata situazione, né ci occuperemo degli algoritmi decisionali, potendo essere scelti tra i tanti già esistenti ed eventualmente essere modificati in base all’implementazione particolare. L’obbiettivo è semplice: creare un’infrastruttura efficace che possa essere facilmente adattata a numerose situazioni.

1.3. DEFINIZIONI

- ARCHITETTURA A SERVIZI/MICROSERVIZI: termine che descrive la pratica di suddividere un’applicazione in una serie di parti più piccole e specializzate, ciascuna delle quali comunica con le altre attraverso interfacce comuni come API e interfacce REST, per creare un’applicazione più grande.
- API REST: un’API REST è un’interfaccia che usa HTTP per gestire dati remoti. L'acronimo REST sta per REpresentational State Transfer
- SENSORE: Unità software/hardware capace di raccogliere dati riguardanti l'ambiente
- ATTUTATORE: Unità software/hardware capace di influenzare l'ambiente causandone una variazione dello stato interno
- DECISORE: Unità software/hardware con la quale il software va interfacciandosi per elaborare i dati raccolti dai sensori
- SPRING: framework per la creazione di architetture a servizi
- SISTEMA/AMBIENTE: per sistema si intende il campo di studio del nostro framework, quindi un punto vendita. Il negozio viene infatti modellato alla stregua di un sistema termodinamico aperto, in cui lo scambio di massa viene sostituito dal passaggio di clienti.
- CONTESTO/STATO: per contesto si intende le proprietà misurabili dell'ambiente (come temperatura, condizione meteo, stagione...), le proprietà misurabili del sistema (temperatura all'interno del negozio, numero di clienti, intensità dei led...) e le proprietà dei singoli clienti (preferenze musicali, età, preferenze sui prodotti...). Lo stato di un sistema rappresenta l'insieme di condizioni che lo influenzano in un dato momento temporale. Di fatto, lo stato riguarda le impostazioni degli attuatori (per esempio la temperatura impostata sul termostato, l'intensità luminosa, la canzone in riproduzione...).
- COMPOSIZIONE: insieme delle unità atomiche che vanno ad influenzare lo stato (l'insieme delle impostazioni dei singoli attuatori). La composizione ottimale viene calcolata in base all'insieme dei contesti e dovrebbe rappresentare lo stato al quale il sistema deve tendere. Ottimamente, è lo stato successivo a quello presente.
- UTENTE: programmatore finale che utilizzerà/adatterà *MC-SHOP* in un'applicazione reale
- CLIENTE: persona che si reca in un ambiente all'interno del quale è stato implementato il progetto. Per esempio, un consumatore all'interno di una boutique di vestiti. Di fatto, colui che subirà passivamente le decisioni dell'implementazione di questo progetto.

1.4. FONTI

Per la documentazione sono state utilizzate le seguenti fonti:

- https://github.com/nbicocchi/ooprogramming
- https://spring.io/guides/gs/rest-service/
- https://www.tutorialspoint.com/maven/index.htm
- https://www.html.it/guide/guida-java-spring/
- https://www.html.it/guide/guida-al-framework-spring-mvc/
- https://it.wikipedia.org/wiki/Framework
- https://en.wikipedia.org/wiki/Microservices
- https://www.garanteprivacy.it/il-testo-del-regolamento

1.5. STRUTTURA DEL DOCUMENTO

- Sezione 1 (questa): Introduzione
- Sezione 2: Descrizione Generale
- Sezione 3: Specifizione dei requisiti
- Appendice con diagrammi UML

2. DESCRIZIONE GENERALE

In questa sezione è riportata la descrizione ad alto livello che espone in generale il prodotto software con le sue maggiori funzioni, caratteristiche degli utenti, vincoli principali e dipendenze.

2.1. PROSPETTIVA SUL PRODOTTO

Per un corretto funzionamento del sistema decisionale sono necessarie alcune astrazioni, le quali verranno salvate come storico all'interno di un database. È necessario creare un modello di preferenze dei clienti al quale adattare il sistema.

Ovviamente, avendo più preferenze manifestate allo stesso tempo, si cercheranno di attribuire valori medi e pesi.

Noi creeremo la composizione dei contesti a partire da 3 fonti d'informazione principali: il contesto **esterno, statistico e personale.**

- Il contesto **esterno** è relativo alle informazioni come meteo atmosferico, l'ora del giorno, il luogo in cui ci si trova e via dicendo. Anche se queste informazioni non coinvolgono direttamente il singolo cliente, sono utili per farsi un'idea delle condizioni generali all'interno delle quali esso si muove (un soleggiato pomeriggio di primavera trasmette emozioni differenti da una piovosa mattina d'inverno). Questo contesto è quello meno influente sulle decisioni poiché estremamente generico. Probabilmente richiederà di collegarsi a fonti di terze parti.
- Il contesto **statistico** riguarda la distribuzione delle presenze nel tempo e nello spazio, dipendendo completamente da altri parametri come l'età media dei clienti. Per esempio, noi potremmo generalizzare come le persone preferiscano musica vivace di mattina e musica lenta di pomeriggio. Si tratta quindi di valutazioni generali a sfondo statistico, utili nel processo decisionale al momento del confronto col dato grezzo fornito dagli altri contesti. Quasi certamente le fonti di queste statistiche saranno di terze parti. Va correlato con la distribuzione delle presenze nel tempo e nello spazio.
- Il contesto **personale** riporta informazioni sul singolo utente, come età, sesso, interessi personali ed abitudini, la propria (recente) storia personale. Si tratta indubbiamente della fonte

di conoscenza più influente ed affidabile, ma anche la più difficile da gestire. È infatti, quasi indubbiamente, necessaria la profilazione della clientela, sia tramite riconoscimento visivo sia tramite collegamenti ad app terze come i social network, i cui utenti sono ben disposti a mettere a nudo la propria anima.

***NOTA**: Per quanto riguarda il contesto personale, come indicato numerose volte lungo l'intero documento, potrebbero sussistere numerosi conflitti con il regolamento europeo per la tutela della privacy.*

2.1.1. INTERFACCIA SISTEMA UTENTE

Il framework sarà distribuito in forma precompilata. Gli utenti potranno immediatamente eseguirne il deployment ed accedervi tramite le api rest messe a disposizione, in maniera da metterlo in comunicazione coi sensori, gli attuatori e gli eventuali decisori. I dati saranno accessibili in tempo reale in formato JSON collegandosi all'interfaccia REST corretta col proprio browser.

2.1.2. ITERFACCIA HARDWARE

La comunicazione con l'hardware sarà presa in carico dalla JAVA Virtual Machine, rendendo il framework completamente indipendente dalla piattaforma di sviluppo. La comunicazione con sensori e attuatori, per contro, avverrà tramite interfacce rest. Ovviamente, questa connessione dovrà essere adeguatamente impostata dall'utente. Sfrutterà l'observer pattern.

2.1.3. INTERFACCIA SOFTWARE

La comunicazione interna ed esterna avviene sempre tramite le interfacce REST.

2.1.4. INTERFACCIA DI COMUNICAZIONE

La comunicazione interna ed esterna avverrà secondo il protocollo http 1.1, tramite un'interfaccia di rete che supporti il protocollo TCP/IP. I dati saranno impacchettati all'interno di oggetti di tipo JSON.

2.1.5. VINCOLI SULL'OCCUPAZIONE DI MEMORIA

Non esistono particolari vincoli sull'occupazione di memoria.

2.2. VINCOLI DI DESIGN

2.2.1. OPERAZIONI (N/A)

2.2.2. VINCOLI DI INSTALLAZIONE

L'installazione è vincolata all'ambiente di sviluppo JAVA, di conseguenza richiede la pre-installazione del Java Runtime Environment 8 o successivi. Gran parte del progetto sarà sviluppato utilizzando la versione di OpenJDK equivalente.

2.3. MACRO Funzionalità DEL SISTEMA

Tramite MC-SHOP ogni utente avrà a propria disposizione le seguenti funzionalità principali:

2.3.1. RECUPERO DEI CONTESTI

- RECUPERO DEL CONTESTO ESTERNO: un servizio che fungerà da observer nei confronti del "mondo reale", ovvero raccoglierà le informazioni riguardanti il mondo esterno.
- RECUPERO DEL CONTESTO STATISTICO: Un servizio capace di raccogliere statistiche generali fornite da terze parti rispetto le quali elaborare decisioni il quanto più precise possibile.
- RECUPERO DEL CONTESTO PERSONALE: Un servizio capace di tenere traccia della profilazione dei clienti attraverso diverse fonti, classificandoli e raggruppandoli secondo parametri principali come età e sesso.

Notare che questi tre servizio potrebbero, nella pratica, corrispondere allo stesso. Per semplificare la struttura del codice, infatti, non è necessaria una chiara differenziazione dei tre, se non tramite l'attribuzione di pesi differenti o semplici flag.

2.3.2. RECUPERO DELLO STATO DEL SISTEMA

L'ottenimento dello stato comporta un'azione simile al recupero dei contesti, ma applicata agli attuatori, osservando su quale valore sono settati. Non vi sono differenze logiche sostanziali tra i diversi attuatori, di conseguenza la funzionalità resta molto generica in confronto a quella di recupero dei contesti.

2.3.3. CREAZIONE DEL GRAFO DELLE COMPOSIZIONI

- RAGGRUPPAMENTO E CLASSIFICAZIONE DEI CONTESTI
- ASSEGNAZIONE DI PESI E PUNTEGGI AI CONTESTI
- CREAZIONE DEL GRAFO DELLE COMPOSIZIONI
- COSTRUZIONE DEL GRAFO DELLE COMPOSIZIONI OTTIMALE

Questa funzionalità è divisa in tre partizioni logiche differenti, le quali compongono la solida base del processo decisionale. Data la molteplicità dei parametri presi in considerazione, un grafo delle composizioni è sempre necessario per poter esplorarne ogni possibile combinazione ed ottenere una decisione efficace e coerente. Se si considerassero i singoli comfort aspects, si potrebbero ottenere strani attuazioni, come musica metal e profumo di fragolina di bosco. Il primo raggruppamento avviene tramite richieste ai servizi di recupero dei contesti.

Una volta ottenuto il grafo delle composizioni, si calcola la composizione ottimale la quale rappresenta lo stato al quale il sistema dovrebbe tendere per ottenere la maggior efficacia.

2.3.4. VARIAZIONE DELLO STATO DEL SISTEMA

- SCELTA DEI TARGET DA INFLUENZARE
  Non tutti gli attuatori possono essere cambiati arbitrariamente: per esempio, selezionata una canzone, si deve tenere conto che essa non potrà essere cambiata per almeno 2 minuti. In altri casi potrebbe essere presa in considerazione una certa latenza, per esempio passare da una temperatura di 27°C a una di 24°C richiederà alcuni minuti. Questa funzione, analizzata la composizione ottimale, dovrà quindi considerare quale siano gli attuatori che possano effettivamente essere variati e quale latenza (considerabile come costo) comporterebbe la relativa variazione. Seguendo una legge simile a quella di Amdahl, si cercherà quindi di ottenere il massimo risultato col minimo sforzo. Se la composizione ottimale rimarrà pressoché invariata per un tempo sufficientemente lungo, tutti gli attuatori si modificheranno di conseguenza facendo combaciare la composizione allo stato effettivo. Ovviamente questa fase è pesantemente influenzata dall'algoritmo decisionale scelto.
- ATTUAZIONE DEL GRAFICO
  Settaggio dello stato virtuale del sistema rispetto al quale gli attuatori dovranno impostarsi. Se lo stato virtuale non combaciasse con quello reale (cioè se il valore da attuare non combaciasse con quello attuato) si avrebbe un sintomo di malfunzionamento del sistema.

2.3.5. SALVATAGGIO DELLO STATO DEL SISTEMA

Creazione di uno storico degli stati reali assunti dal sistema.

2.3.6. SALVATAGGIO DEI CONTESTI

Creazione di uno storico dei contesti misurati dal sistema.

2.4. CARATTERISTICHE DEGLI UTENTI

Gli utenti a cui è rivolto MC-SHOP sono principalmente sviluppatori software sia esperti che inesperti, i quali abbiano un minimo di conoscenza per quanto riguarda il funzionamento di un'API rest, il cui scopo sia utilizzare questo progetto per costruire architetture complesse senza necessitare eccessiva esperienza.

2.5. VINCOLI GENERALI – DA FARE ASSOLUTAMENTE PORCA TROIA

I principali elementi di criticità del framework in questione sono le seguenti:

2.6. ASSUNZIONI E DIPENDENZE –PORCAZZAZOZZA

Le principali ipotesi su cui si basa il documento riguardano:

- La presenza di connessioni di rete che seguano il protocollo TCP/IP dalla latenza contenuta
- La capacità da parte degli utenti di gestire sia sistemi centralizzati che distribuiti
- La capacità di gestire un server remoto da parte degli utenti
- Il consenso da parte dei clienti in merito alla profilazione
- Il rispetto degli utenti delle norme vigenti sulla privacy, come designate dall'unione europea
- L'utilizzo non governativo del prodotto
- L'utilizzo non militare del prodotto
- L'utilizzo di questo prodotto con dispositivi dotati di una JVM.

2.7. REQUISITI DA ANALIZZARE IN FUTURO

Il contesto di un cliente è preso in particolare considerazione nel caso degli e-commerce, i quali lo utilizzano per indirizzare l'individuo verso l'acquisto di prodotti di suo eventuale interesse. In tal senso, non necessitano di eventuali attenzioni per quanto riguarda il comfort, eccetto quello legato alla semplicità delle operazioni (che è ovviamente standard per tutti i clienti). Per gestire efficacemente i consigli, non di meno, è sufficiente avere uno storico degli acquisti degli utenti, riconosciuti tramite un identificativo univoco ma non personale. Per gestire i comfort aspects, per contro, è necessaria una raccolta dati assai approfondita.

In futuro sarà quindi importante spostare la gestione della privacy dalle assunzioni ai requisiti, evitando che il peso di una questione così complessa e "pericolosa" gravi sulle spalle degli utenti. Il regolamento Europeo, infatti, è particolarmente stringente riguardo all'utilizzo di mezzi come il riconoscimento facciale, e in futuro lo sarà sempre di più. Al di fuori della ricerca è permesso a pochissimi software l'utilizzo del video riconoscimento, un esempio è il SARI, utilizzato dalle forze dell'ordine (e permesso solo perché sarebbe un lavoro in alternativa svolto da personale umano, molto meno efficiente). Di applicazioni puramente commerciali capacità di avvalersi di strumenti così sofisticati e precisi, in Italia, non ne esistono, essendo per giunta bandite. I dati biometrici sono i più sensibili e personali, e quindi quelli meglio protetti dal garante della privacy; essi non possono essere cambiati come un nome utente o un'email.

Partendo da quello precedente, un ulteriore requisito futuro potrebbe essere l'utilizzo di dispositivi tecnologici capaci di riconoscere l'ingresso del cliente all'interno di uno spazio senza bisogno dei suoi dati biometrici. Abbiamo già alcune idee, ma tutte ferme allo stadio di studio teorico.

3. SPECIFICA DEI REQUISITI

3.1. REQUISITI FUNZIONALI

- RECUPERO DEI CONTESTI

RF01	
INPUT	Tipo di contesto, tipo di dato, dato, id sensore
PROCESS	Registra il sensore nel caso fosse la prima volta che si connette, screma il rumore, ordina i dati
OUTPUT	I dati ordinati e scremati dal rumore. Ottenimento composizione sistema attuale.

- RECUPERO DELLO STATO DEL SISTEMA

RF02	
INPUT	Tipo di dato trattato, dato, id attuatore
PROCESS	Registra l'attuatore la prima volta, screma il rumore sempre, ordina i dati in ingresso
OUTPUT	I dati riordinati e puliti dal rumore. Ottenimento dello stato attuale del sistema.

- CRAZIONE DEL GRAFO DELLE COMPOSIZIONI

RF03	
INPUT	Composizione attuale del sistema.
PROCESS	Assegnazione dei punteggi ai componenti della composizione. Creazione connessioni accurate tra comfort aspects. Creazione grafo delle composizioni dei comfort aspects.
OUTPUT	Grafo delle composizioni possibili

- VARIAZIONE DELLO STATO DEL SISTEMA

RF04	
INPUT	Grafo delle composizioni possibili
PROCESS	Identificazione del miglior percorso all'interno del grafo, il quale possiede la maggior probabilità di creare la composizione ottimale. Identificazione degli attuatori sui quali è possibile intervenire (attraverso l'analisi di svantaggi in caso di variazione errata)
OUTPUT	(Eventuale) variazione del valore assegnato agli attuatori

- SALVATAGGIO DELLO STATO DEL SISTEMA

RF05	
INPUT	Stato del sistema
PROCESS	Accesso ad un database e salvataggio dello stato
OUTPUT	Storico degli stati del sistema

- SALVATAGGIO DEL CONTESTO DEL SISTEMA

RF06	
INPUT	Composizione attuale dei contesti del sistema
PROCESS	Accesso ad un database e salvataggio delle composizione dei contesti
OUTPUT	Storico dei contesti

3.2. REQUISITI NON FUNZIONALI

RN01	RQUISITI PRESTAZIONALI | REQUISITI PER LE MIGLIORI PERFORMANCE DEL SISTEMA
Descrizione	Per poter ottenere le migliori performance da parte del sistema, è necessaria una connessione internet performante che segua lo standard TCP/IP. È consigliato eseguire i servizi decisionali su macchine dedicate con ottime prestazioni in termini di capacità di calcolo e throughput, in maniera tale da minimizzare la latenza. Si consiglia di utilizzare una rete cablata almeno Cat5e.

RN02	RQUISITI PRESTAZIONALI | SISTEMA LOCALE
Descrizione	Gli attuatori e i sensori vengono automaticamente raggruppati via software, ma sarebbe meglio dividerli anche fisicamente in maniera analoga, per esempio dedicando un servizio di raccolta dei contesti/degli stati ed un controller per ogni

	stanza. In questa maniera diventerà più semplice la manutenzione e il sistema sarà più performante dato il minor carico sui singoli servizi.

RN03	RQUISITI PRESTAZIONALI | CALCOLATORI MULTIPROCESSORE (O MULTICORE)
Descrizione	Nel caso si eseguissero più servizi sul medesimo calcolatore, è suggerito l'utilizzo di macchine costituite da più processori/core, possibilmente uno per servizio, o almeno capaci di sostenere l'hyperthreading (ed equivalenti).

RN04	RQUISITI PRESTAZIONALI | CONSUMO ENERGETICO
Descrizione	Data la probabile estensione di un progetto implementante MC-SHOP, data inoltre la necessità di mantenerlo attivo quasi 24/7, si suggerisce l'adozione di dispositivi a basso consumo energetico e si sconsiglia l'implementazione su dispositivi datati. In caso contrario, si rischia che il costo di utilizzo sia maggiore dei benefici tratti.

RN05	DATABASE | SPECIFICHE DATABASE
Descrizione	Si richiede che il database utilizzi un motore supportante lo standard SQL. Preferibilmente, il database deve essere eseguito localmente e dev'essere inaccessibile da terze parti, in maniera che non si rischino furti di dati sensibili.

RN06	ATTRIBUTI DEL SISTEMA | SICUREZZA DATI UTENTI
Descrizione	Data la sensibilità dei dati trattati, si preferisce un sistema con accesso all'esterno controllato, limitando l'esposizione delle API rest solo alla rete locale, possibilmente con l'utilizzo di sotto reti fisiche/virtuali.

RN07	ATTRIBUTI DEL SISTEMA | Portabilità
Descrizione	Il framework, venendo eseguito tramite la JVM, è indipendente dal sistema operativo e dall'hardware di implementazione. Il lavoro dell'utente, però, potrebbe non essere altrettanto portabile.

RN08	ATTRIBUTI DEL SISTEMA | Modularità
Descrizione	Utilizzando un'architettura a micro servizi, si garantisce un'estrema modularità da parte del software, inoltre si suggerisce di mantenere un'alta modularità anche nell'implementazione, evitando di accorpare troppi sensori/attuatori su di un unico servizio.

APPENDICE

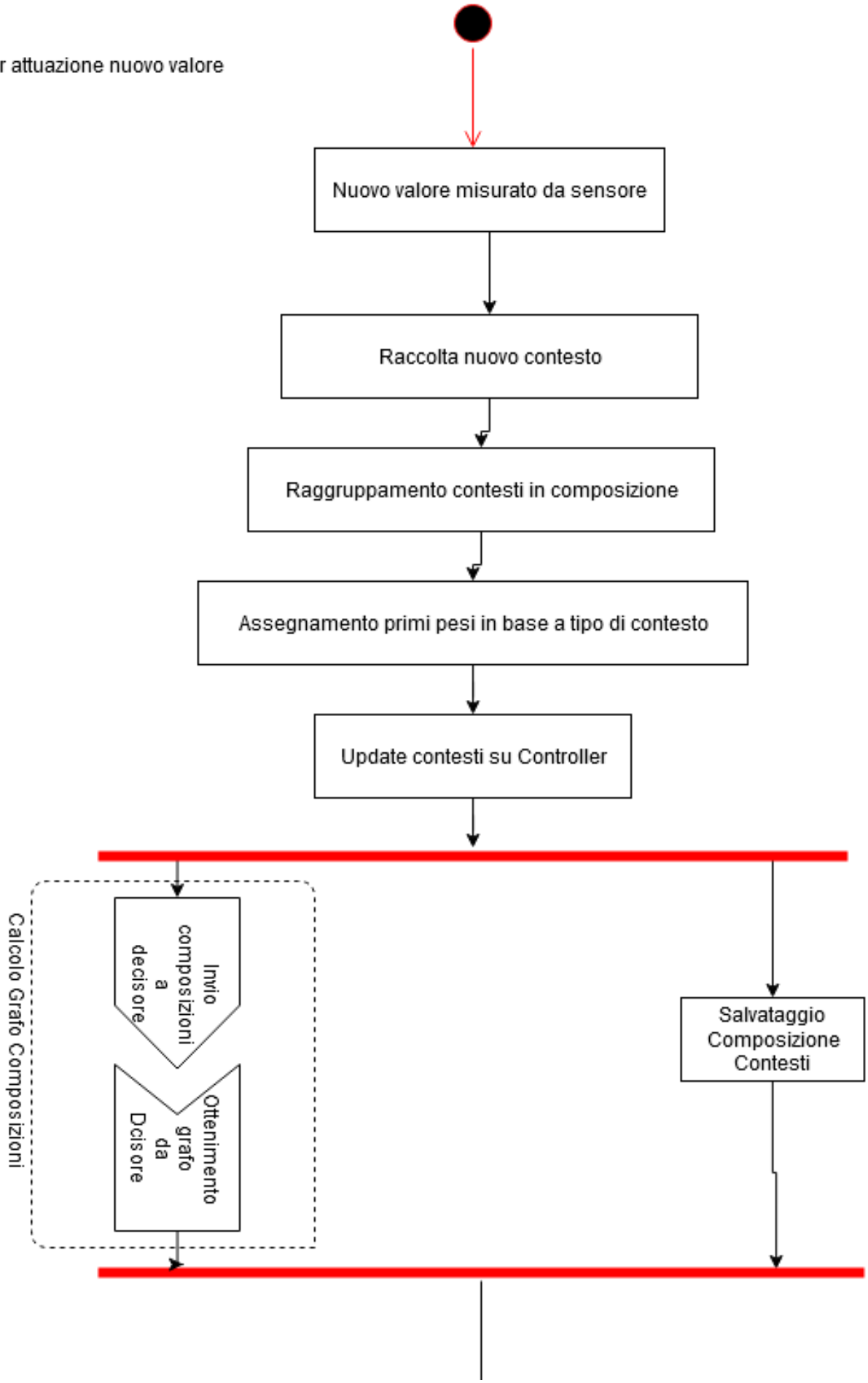
In questa sezione finale saranno riportati i diagrammi rappresentati i principali requisiti e il rapporto tra le principali funzionalità svolte da questo progetto software. I digrammi seguiranno lo standard UML, e devono essere considerati a puro fine illustrativo. Non sarà riportata alcuna riga di codice, essendo al di fuori dell'ambito di interesse di questo documento.

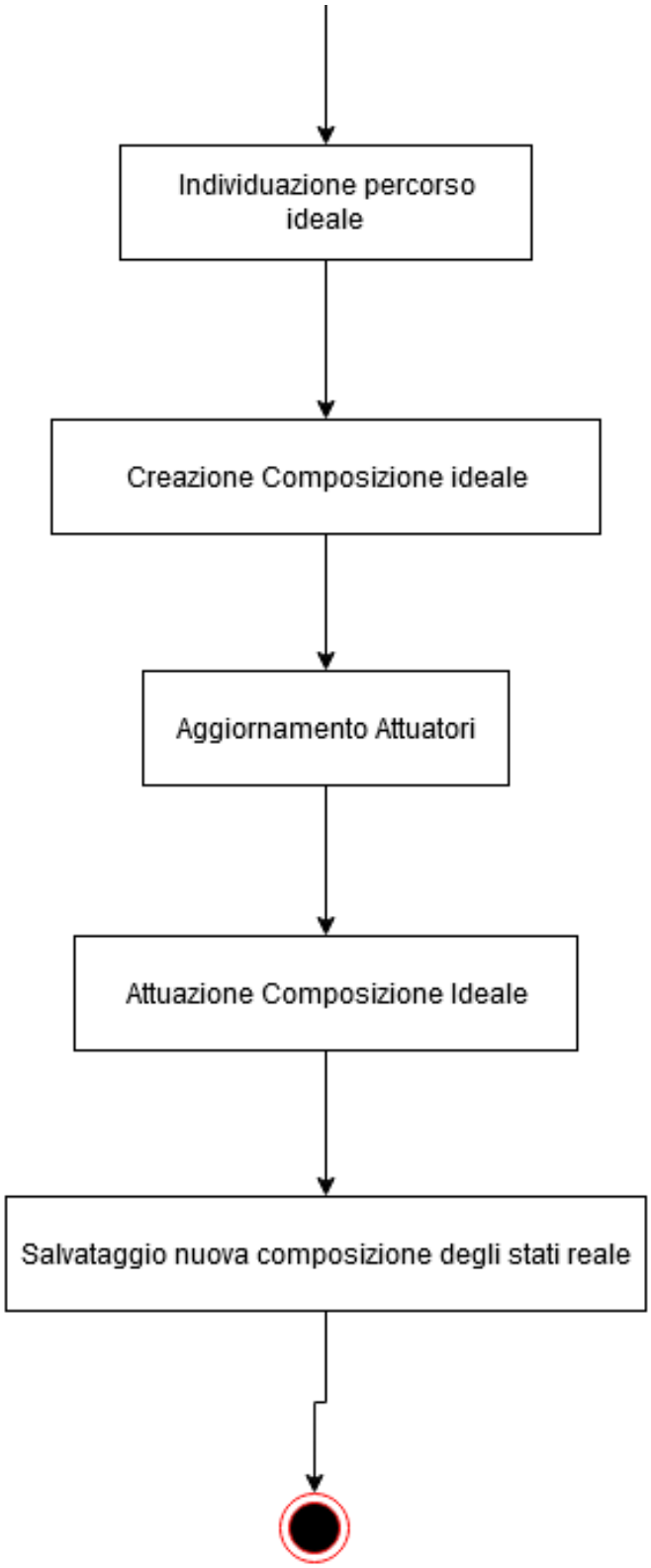
Sarà inoltre evidente come la complessità espressa nei paragrafi precedenti fosse puramente concettuale, ma in un'implementazione reale le varie differenziazioni, come quelle sulla tipologia del contesto, saranno puramente formali e liquidate con l'affibbiazione di un semplice peso differente.

Come sottolineato al paragrafo 1.2, infatti, il punto di forza di *MC-SHOP* deve essere la semplicità e l'adattabilità. Queste due qualità si possono riconoscere, spesso, nei progetti la cui soluzione al problema è assai più banale della complessità concettuale necessaria ad esporre il problema stesso.

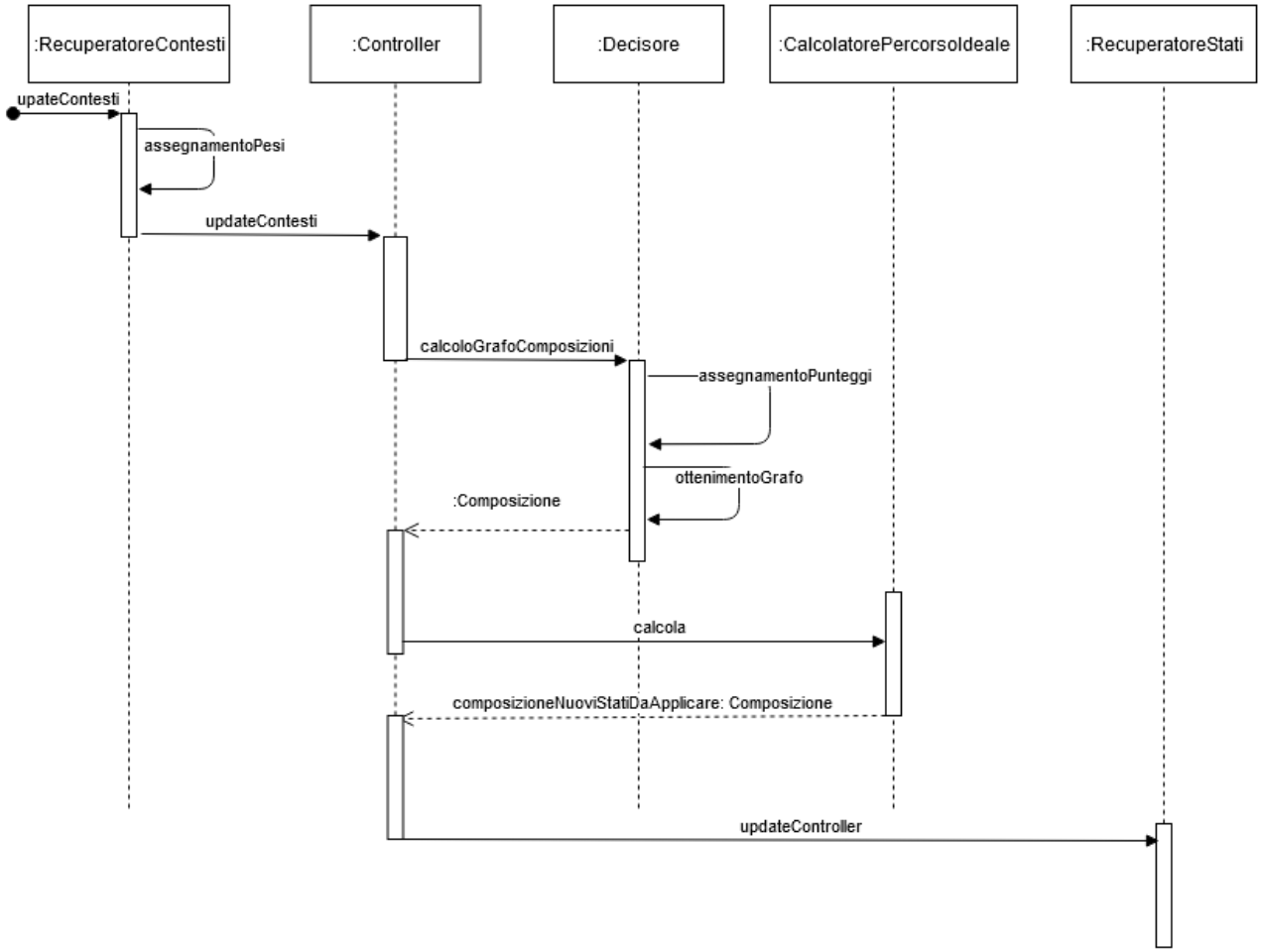
La mancata differenziazione tra contesto esterno, statistico e personale è quindi voluta. Utilizzando l'attribuzione di pesi iniziali al posto di tipi ben definiti ci permette di espandere queste tre tipologie ad un numero infinito di possibilità in futuro.

Activity diagram, percorso per attuazione nuovo valore

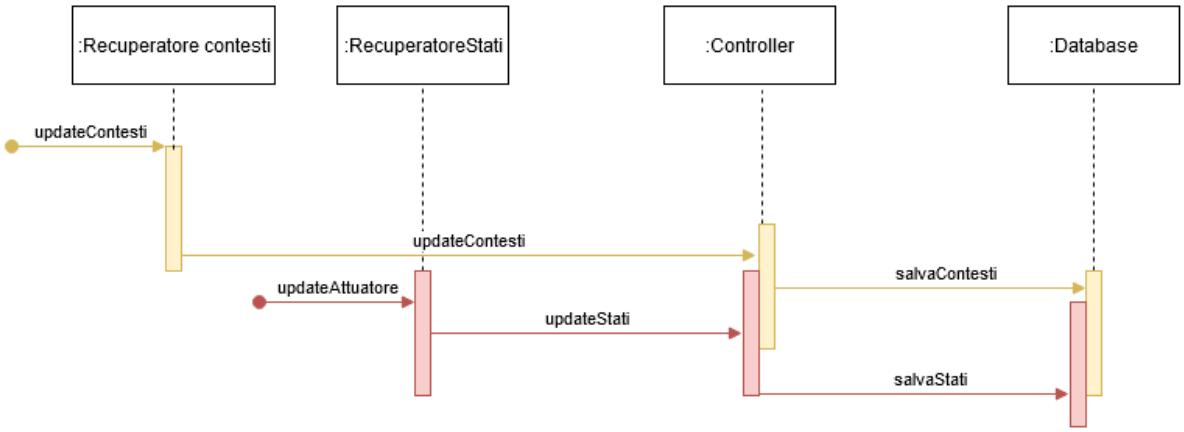



Individuazione percorso ideale
Creazione Composizione ideale
Aggiornamento Attuatori
Attuazione Composizione Ideale
Salvataggio nuova composizione degli stati reale

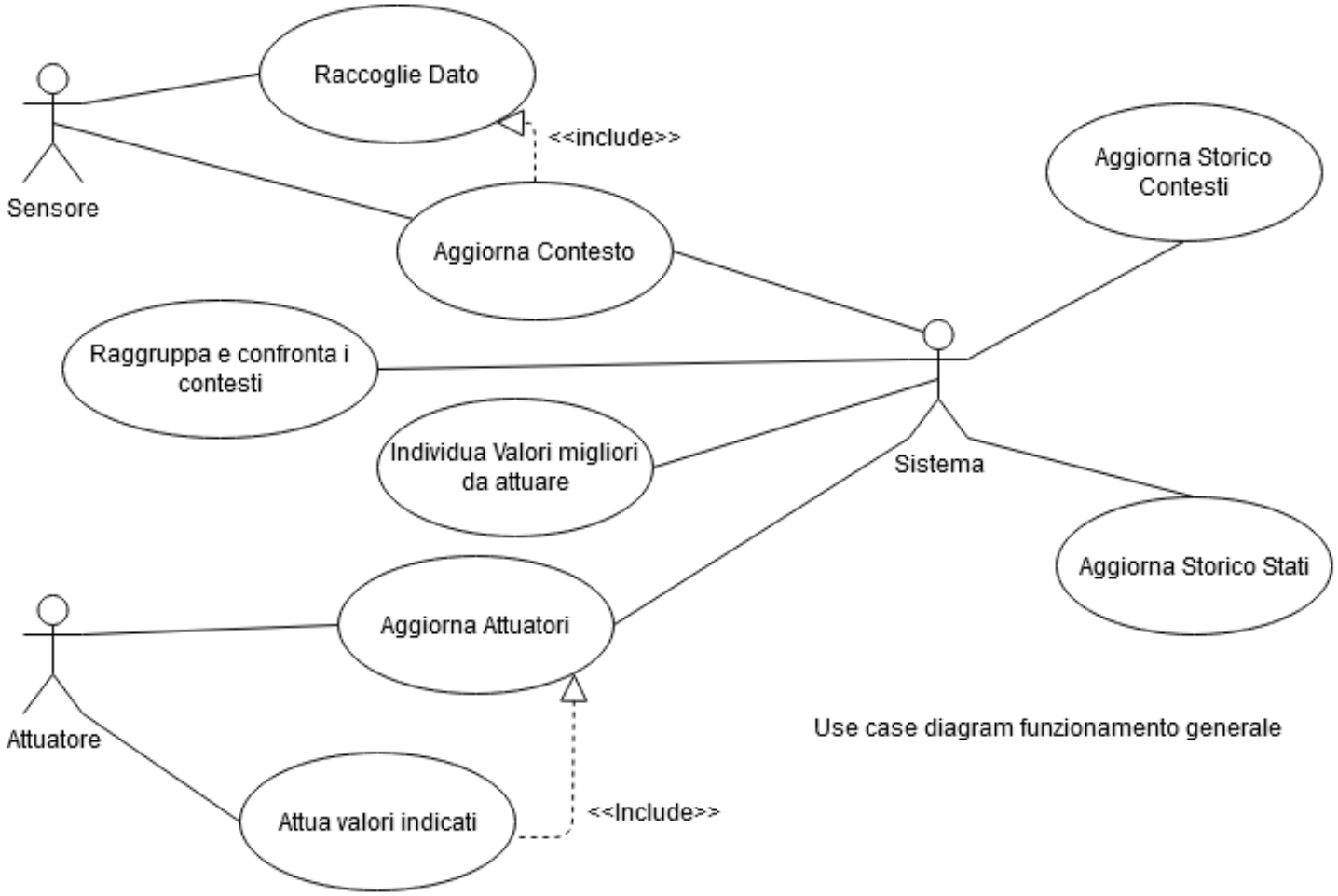
Sequence Diagram dell'individuazione di un nuovo stato ideale



Sequence diagram della creazione di uno storico contesti/stati

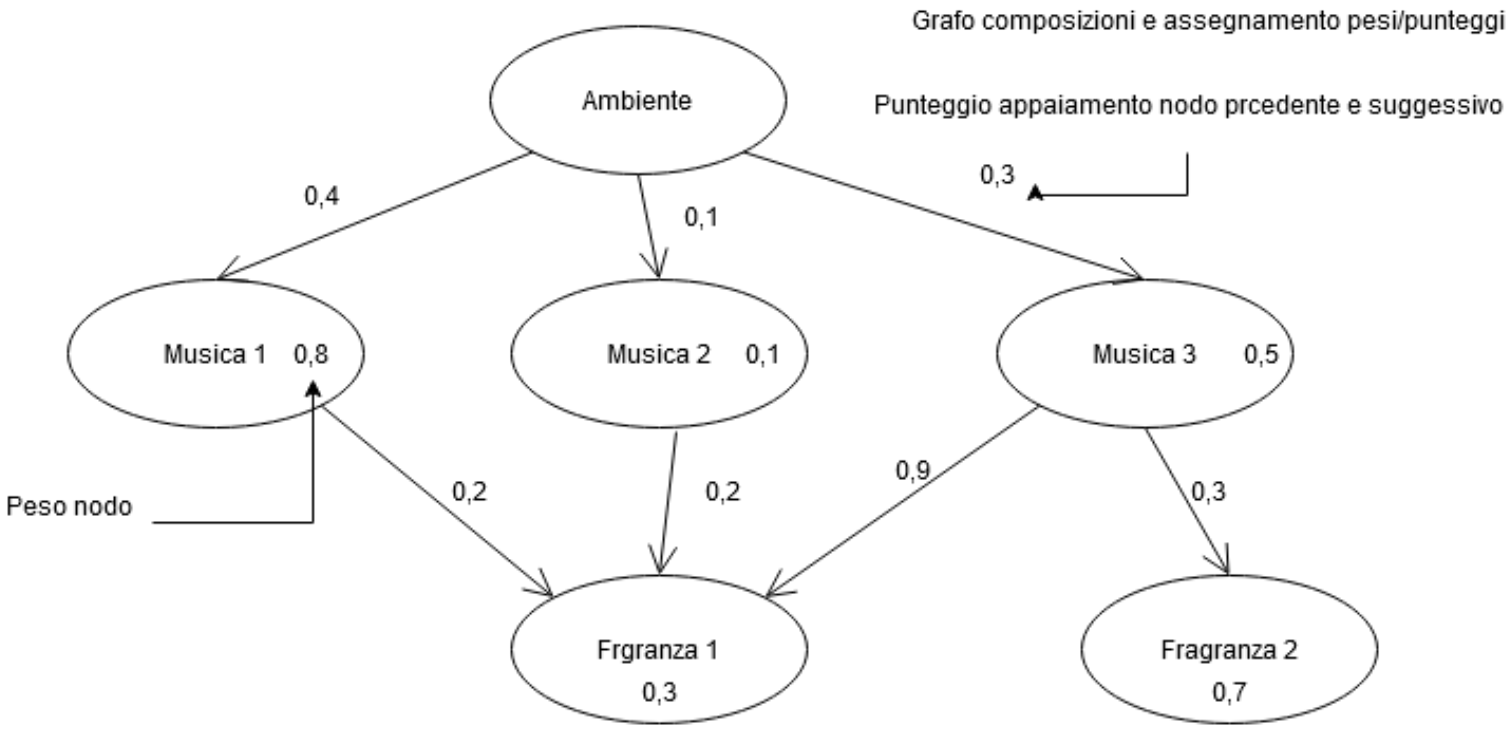


Il salvataggio di nuovi contesti e di stati attivi può avvenire in maniera del tutto indipendente e contemporanea.
Il ciclo di funzionamento previsto del software è illustrato nell'activity diagram, esso mostra come il salvataggio dei contesti debba avvenire prima del salvataggio degli stati, essendo questi ultimi registrati al momento della loro variazione.
Ma ciò non significa che, durante il termine del ciclo naturale, non ne abbia già avuto inizio un secondo, che andrà a lavorare in parallelo.
Thread rosso: salvataggio degli stati
Thread giallo: salvataggio dei contesti



Use case diagram funzionamento generale

Grafo composizioni e assegnamento pesi/punteggi



Si può notare l'utilizzo del design pattern "observer" per quanto riguarda il rapporto tra i sensori/attuatori e il reuperatori di stati/contesti, nonché nella comunicazione tra i recuperatori e i controller

Inoltre, è presente l'utilizzo del "facade" pattern per quanto riguarda il Controller e il template pattern all'interno del decisore.
Quest'ultimo, infatti, necessiterà di essere implementato dall'utente, ma sarà costretto ad eseguire due operazioni fondomantali che portino alla creazione di un grafo riportante dei pesi/punteggi, in maniera tale che il controller possa individuare il percorso ideale utilizzando la relativa classe.

Negli UML successivi non sarà owiamente riportato alcun funzionamento dei pattern essendo questi ultimi standard.

Il decisore può essere esteso con una classe rale che funga da api per collegarsi ad un decisore reale a piacimento implementato dall'utente

RecuperatoreContesti

+ip:String

+updateContesti(contesto:Contesto):void
-assegnamentoPesi():void
+notifyControllers():void
+addController(controller:controller):void
+removeController(controller:Controller):void
+mostraDatiRaccolti():JSONObject

CalcoloGrafoComposizioni chiama assegnamentoPunteggi e ottenimentoGrafo

<<Abstract>>
Decisore

+calcoloGrafoComposizioni(contesti:Coposizione):composizione
-assegnamentoPunteggi():Composizione
-ottenimentoGrafo():Composizione

Use

RecuperatoreStati

+ip:String

+updateAttuatore(stato:Stato):void
+removeAttuatore(attuatore:Attuatore):void
+addAttuatore(attuatore:Attuatore):void
+notifyAttuatori():void
+updateController(composizione:COmposizione):void
+notifyControllers():void
+addController(controller:Controller):void
+removeController(controller:COntroller):void
+mostraDatiRaccolti():JSONObject

Controller

+ip:String

+updateContesti(composizione:Composizione, ipProvenienza:String):void
+updateStati(composizione:composizione, ipProvenienza:String):void
+addAttuatori(attuatore:RecuperoStati):void
+removeAttuatori(attuatore:RecuperoStati):void
-notifyAttuatori():void
+mostraStoricoContesti():JSONObject
+mostraStoricoStati():JSONObject

Use

<<Interface>>
Database

+salvaStati(stati:Composizione, ipProvenienza:String)
+salvaContesti(contesti:Composizione, ipProvenienza:String)
+ottieniStoricoContesti():Composizione
+ottieniStoricoStati():Composizione

<<Interface>>
Sensore

+add()
+remove()
+notify()

<<Interface>>
Attuatore

+update()
+remove()
+add()
+notify()

updateContesti chiama Decisore.calcoloGrafoComposizioni(), CalcolatorePercorsoIdeale.calcola(), Database.salvaContesti() e self.notifyAttuatori();

updateStati() chiama database.salvaStati()

mostraStoricocontesti chiama Database.ottieniStoricoContesti

mostraStoricoStati chiama Database.ottieniStoricoStati

Il Controller si occuperà di tenere in comunicazione servizi di recupero contesti con quelli di recupero degli stati. Ogni controller sarà considerato come "servizio locale", cioè ogni servizio recuperatore di contesti ed ogni servizio recuperatore di stati che si collega al medesimo controller verrà considerato come parte del medesimo raggruppamento

CalcolatorePercorsoIdeale

+calcola(grafo:Composizione):Composizione

Use

Static

Contesto

+tipo:String
+idSensore:int

Stato

+idAttuatore:int

<<Abstract>>
Composizione

+tipo:String
+dato:Object
- peso:Integer

+assegnaPeso(peso:Integer):Integer
+aggiungi(comp:Composizione):void
+rimuovi(comp:Composizione):void
+getChild():Composizione

1
1
1
Extends
Extends
Extends
0..n

Dato

+ assegnaPeso(peso:Integer):Integer

Composizione

+assegnaPeso(peso:Integer):Integer
+aggiungi(comp:Composizione):void
+rimuovi(comp:Composizione):void
+getChild():Composizione

CLASS DIAGRAM completo