POLITECNICO DI MILANO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

ADATTIVITA' IN APPLICAZIONI ORIENTATE AI SERVIZI BASATA SU UN'ANALISI PROATTIVA E GRANULARE DEI DATI CONTESTUALI

Relatore: Prof. Barbara Pernici

Tesi di Laurea di:
Alessandra Sandonini
Matr. 739286

Anno Accademico 2010 / 2011

Ai miei genitori, Luca e Miky

Prefazione

Le applicazioni orientate ai servizi sono sviluppate in un ambiente dinamico e distribuito, mediante la composizione di servizi eterogenei e forniti da diversi service provider. Essi non sono sotto il controllo dello sviluppatore del sistema, ma vengono utilizzati per sfruttarne le funzionalità offerte. Ciò introduce una dipendenza critica tra applicazione e servizio, soggetto a cambiamenti o a improvvise indisponibilità. Le SOA devono quindi prevedere meccanismi adattivi per reagire a cambiamenti in modo dinamico e automatico. L'adattività può avvenire in modo reattivo, dopo la rilevazione di un'anomalia, o proattivo, prima che essa si verifichi, sulla base di stime sul comportamento del sistema e di dati rilevati a runtime riguardo l'applicazione e il contesto associato. Per il monitoraggio di dati contestuali vengono spesso usate reti di sensori. Bisogna determinare la posizione dei sensori, quanto spesso interrogarli e quali dati raccogliere per fornire dati accurati evitando sprechi energetici. Lavorare con dati ad una granularità molto fine potrebbe provocare uno sforzo significativo in termini operazionali, causando problemi nelle prestazioni del sistema. Nella raccolta di dati contestuali si deve anche considerare la granularità di monitoraggio del dato, cioè la frequenza con cui misurare i dati per catturare le caratteristiche dell'ambiente. Nelle emergenze bisogna interrogare i sensori più spesso rispetto alla norma e in altri casi una misurazione più frequente dei dati contestuali non fornisce alcuna informazione aggiuntiva sullo stato dell'ambiente. Si vuole evitare da un lato la raccolta di dati superflui e dall'altro la raccolta di un numero di dati non sufficiente a rilevare le modifiche nel contesto. Nelle applicazioni

adattive, i dati utilizzati per la descrizione del contesto devono essere sempre aggiornati, per evitare adattamenti errati. Lo scopo della tesi è proporre un framework per gestire l'adattività, analizzando dati contestuali in relazione al loro livello di dettaglio e alla frequenza con cui è necessario aggiornarli, adottando un approccio proattivo per determinare le esigenze di adattamento e per l'invocazione dinamica dei servizi.

Abstract

Service-oriented applications are deployed in dynamic and distributed setting by composing different types of services deployed by different service providers. Such services are often not under the control of systems developers, but they are exploited to obtain specific functionalities. This introduces critical dependencies between service-based application and the services that could change without notice or be unavailable. Therefore, service-based applications have to be equipped with adaption capabilities to react dynamically and automatically to critical issues. Adaption can be performed in a reactive way, after a deviation or critical changes has occurred, or in a proactive way, before that changes are monitored, based on predictions about the actions of the applications and on runtime informations about the application and its context. Sensor networks are often used to monitor contextual data. It is necessary to decide where to place sensors, how frequently poll them and which data collect to provide precision, avoiding waste of energy. Another issue about the collection of contextual data is the monitoring granularity, which represents the frequency of monitoring data to catch the significant features of the enviroment. In a state of emergency, it is necessary to poll sensors more often rather the norm and in other situations a more frequent measure of contextual data doesn't provide any additional information about the state of enviroment. Thus it is convinient to avoid on one side the collection of unneccessary data, and on the other side the collection of a number of data that are not suitable to catch changes in the execution context. In context-aware applications, data could always be updated, otherwise the

adaption is performed in a wrong way. The aim of this work is to propose a framework that provides the adaption, analizing contextual data related to the detail level and to the frequency of the update, using a proactive approach to establish the adaption needs and the dynamic invocation of the services.

Ringraziamenti

Vorrei innanzitutto ringraziare la Professoressa Barbara Pernici che mi ha seguito, consigliato e dedicato tempo in questa esperienza di tesi. Grazie alla mia famiglia che mi ha sostenuto in tutti questi anni, in particolare i miei genitori, nonni, Luca e Noemi, anche per la consulenza inglese. Desidero ringraziare inoltre tutte le persone che mi sono state vicine durante questo lavoro di tesi, Miky per avermi supportato e sopportato, gli amici di sempre ed i compagni di corso.

Indice

Elenco delle figure

Capitolo 1

Introduzione

A partire dagli anni Novanta, lo sviluppo sempre più massiccio di Internet e delle tecnologie web ha posto grande interesse sulle architetture orientate ai servizi (SOA). Chiunque disponga di un computer e di una connessione può accedere velocemente ad Internet ed usufruire di molteplici servizi presenti in rete. Per questo motivo le aziende anzichè realizzare programmi ad hoc per ogni esigenza di business scelgono di sfruttare servizi già disponibili in rete che forniscano le funzionalità richieste, selezionandoli, integrandoli e componendoli. Ciò rappresenta un vantaggio significativo in termini di costi e tempi di implementazione e progettazione, ma richiede l'utilizzo di tecnologie standard, e per quanto possibile aperte, per garantire la massima omogeneità tra servizi diversi e la capacità di scambiarsi messaggi.
Inoltre l'utilizzo dei tradizionali middleware aziendali per la comunicazione tra diverse applicazioni funziona egregiamente quando si tratta di applicazioni realizzate dalla stessa organizzazione. Nei sistemi informativi aziendali spesso nasce l'esigenza di far collaborare applicazioni sviluppate da aziende diverse. La maggior parte dei middleware aziendali pongono limitazioni sul sistema operativo, l'architettura hardware e software o il linguaggio di programmazione da utilizzare. L'utilizzo del web come mezzo di comunicazione risolve questa limitazione, garantendo interoperabilità tra applicazioni sviluppate in linguaggi di programmazione diversi, su macchine differenti e

fornite da varie organizzazioni.

I *Web Service* sono definiti come delle applicazioni modulari che forniscano funzionalità di business sul web, consentendo alle applicazioni che vi si collegano di usufruire delle operazioni che mettono a disposizione. I servizi sono caratterizzati da interfacce note e ben definite, attraverso le quali possono cooperare tra loro per raggiungere obiettivi comuni e permettere agli utenti di invocarne le funzionalità.
Un'architettura orientata ai servizi tende a disaccoppiare il legame tra le funzionalità offerte dal servizio e la sua specifica implementazione.
I servizi web disponibili in rete, sono spesso eterogenei e forniti da diverse organizzazioni. Vengono quindi utilizzate tecnologie standard per la loro descrizione, interazione e pubblicazione.
Il *WSDL (Web Service Description Language)* è uno standard basato su XML utilizzato per la descrizione dei servizi. Viene descritta l'interfaccia pubblica di un web service ovvero creata una descrizione, basata su XML, di come è possibile interagire con un determinato servizio. In particolare viene descritto per cosa può essere utilizzato il servizio (le operazioni messe a disposizione), come utilizzarlo (il protocollo di comunicazione da utilizzare per accedervi, il formato dei messaggi accettati in input e restituiti in output ed i dati correlati) e dove utilizzare il servizio (il cosiddetto endpoint del servizio che solitamente corrisponde all'indirizzo - in formato URI - che rende disponibile il Web Service).
Le operazioni supportate dal Web Service ed i messaggi che è possibile scambiare con lo stesso sono descritti in maniera astratta e quindi collegati ad uno specifico protocollo di rete e ad uno specifico formato.
Per lo scambio di messaggi tra web service differenti viene utilizzato il protocollo *SOAP (Simple Object Access Protocol)*, anch'esso basato su XML. La sua struttura segue il paradigma Head-Body, l'Header contiene metainformazioni come quelle che riguardano il routing, la sicurezza, le transazioni e parametri per l'orchestrazione. Il segmento obbligatorio Body trasporta il contenuto informativo e talora viene detto carico utile, o payload. Questo

deve seguire uno schema definito dal linguaggio XML Schema.
Per la pubblicazione di un web service il registro più frequentemente utilizzato è *UDDI (Universal Description Discovery and Integration)*, basato su XML ed indipendente dalla piattaforma hardware.

L'*architettura SOA* descrive il modello dei web service, analizzando le tre entità che entrano in gioco durante il suo ciclo di vita: il fornitore del servizio, il registro web e l'utilizzatore finale del servizio e la loro interazione.
Il *fornitore* crea o semplicemente offre il servizio, lo descrive con un linguaggio basato su XML e si occupa di pubblicarlo in un registro centrale. Il *registro* contiene la descrizione del servizio arricchita di informazioni aggiuntive sul fornitore, come l'indirizzo e i contatti dell'organizzazione a cui appartiene e di dettagli tecnici sul servizio. L'*utilizzatore* accede al registro per la ricerca dei servizi e utilizza la descrizione del servizio fornita per accedervi e per utilizzarlo.
Un'architettura orientata ai servizi mette quindi al centro dell'attenzione il servizio, favorendo l'integrazione tra servizi, l'interoperabilità e la riusabilità dell'applicazione.

Per quanto riguarda l'utilizzo dei web service nei processi di business aziendali, si verifica la necessità di coordinare in modo automatico le attività svolte nei vari servizi allo scopo di raggiungere un obiettivo comune. Le applicazioni software per il supporto di flussi di lavoro, prendono il nome di sistemi di gestione dei workfolw e permettono lo sviluppo e l'utilizzo di processi complessi ottenuti mediante lo svolgimento anche ripetuto di attività, seguendo opportuni criteri di esecuzione. Con il termine *orchestrazione* si indica come i servizi possano interagire tra di loro a livello di messaggi, basandosi sulla logica di business, e individuando l'ordine di esecuzione delle interazioni e le condizioni sotto le quali debba essere svolta una determinata attività.
Il linguaggio maggiormente utilizzato per l'orchestrazione, il coordinamento e l' esecuzione di processi aziendali è il BPEL (Business Process Execution Language), basato sull'XML e costruito per descrivere formalmente i pro-

cessi commerciali ed industriali in modo da permettere una suddivisione dei compiti tra attori diversi.
Sono inoltre stati sviluppati numerosi tool che forniscano un'interfaccia grafica per la configurazione del workflow, detti BPEL designer. Il processo di business è descritto in maniera astratta sottoforma di diagramma delle attività ed è connesso a runtime con gli endpoint degli specifici servizi. La descrizione del servizio è effettuata nel modo più astratto possibile allo scopo di poter generalizzare il comportamento del processo di business anche sostituendo i singoli servizi concreti che lo compongono.
Un'applicazione BPEL viene essa stessa invocata sottoforma di web service ed interpretata mediante un BPEL engine.
Nell'ambito del Service Oriented Computing (SOC) i servizi non sono realizzati per un utente specifico, ma per una categoria di potenziali utenti che tipicamente ricercheranno e comporranno i servizi dinamicamente. I servizi devono quindi essere il più possibile generici in modo da poter essere utilizzati per esigenze differenti da utenti diversi e operanti in contesti specifici.
In questo ambito sono introdotti i concetti di *adattività* e *context-awareness*.
Un'applicazione basata sui servizi è composta da web service forniti da terze parti e non controllati direttamente dall'utente. Ciò significa che i servizi possono evolvere o cambiare improvvisamente e senza la volontà esplicita dell'utente. L'esigenza è quella di lavorare in un ambiente distribuito e fortemente dinamico, fornendo applicazioni flessibili e in grado di adattarsi a cambiamenti contestuali o nei requisiti.

I cambiamenti nei web service possono essere statici e verificarsi a design-time, oppure dinamici ed essere osservati a runtime. Possono verifcarsi *cambiamenti nei requisiti* che costituiscono il principale fattore di cambiamento. I requisiti rappresentano una sorta di benchmark per monitorare il corretto funzionamento del web service implementato. I *cambiamenti nell'interfaccia* descrivono modifiche nelle operazioni fornite dal servizio e nella struttura dei messaggi. In particolare, rappresentano l'aggiunta di nuove funzionalità al servizio oppure l'aggiornamento di quelle esistenti. I *cambiamenti nell'im-*

plementazione del web service sono strettamente correlati ai cambiamenti dell'interfaccia. Sono causati da ottimizzazioni o modifiche del codice o da cambiamenti nelle funzionalità. I *cambiamenti nella QoS* sono gli unici ad essere monitorati unicamente a runtime e dipendono da modifiche delle proprietà dei web service. Sono caratterizzati da parametri come la qualità dei dati, le performance del sistema e da fattori esterni.

Allo scopo di reagire ai cambiamenti, a seconda del momento in cui si verificano, si distingue tra *evoluzione* del sistema, che si realizza quando si presenta la necessità di effettuare modifiche definitive nell'applicazione e legate alle funzionalità del sistema, ed *adattività*, quando si fa fronte a cambiamenti che caratterizzino unicamente la singola istanza di esecuzione dell'applicazione e non il sistema nella sua totalità. Con adattività si intende quindi la capacità delle applicazioni di modificare il proprio comportamento dinamicamente a seguito della rilevazione di cambiamenti a runtime nello svolgimento dell'applicazione stessa o nel contesto di esecuzione. Una strategia di adattamento deve quindi essere progettata e prevista a design-time, ma viene messa in atto a runtime, durante la specifica istanza dell'applicazione in esecuzione. L'adattività è una caratteristica fondamentale nelle applicazioni orientate ai servizi, soggette a cambiamenti improvvisi, operanti in un ambiente distribuito e utilizzate da utenti differenti all'interno di processi di business diversi. Il loro ciclo di vita è quindi raffinato con le caratteristiche riguardanti la progettazione e la messa in atto di meccanismi adattivi. I meccanismi adattivi utilizzati devono essere il più possibile automatici, cercando di ridurre al minimo l'intervento diretto dell'utente. Inoltre, essendo l'adattività un'attività costosa, dev'essere realizzata solo se necessaria, evitando adattamenti inutili o errati.

Un sistema di questo tipo deve quindi monitorare se stesso e il contesto circostante, rilevare i cambiamenti significativi, decidere come reagire e mettere in atto le decisioni prese. Le situazioni che generano un comportamento anomalo sono rilevate da opportuni parametri, denominati *trigger* di adattamento e possono essere correlati al contesto di esecuzione, all'applicazione stessa o

a fattori esterni.
L'adattività può essere realizzata sia a seguito del verificarsi di una situazione anomala ed è detta di tipo *reattivo*, oppure prima che l'anomalia si verifichi, sulla base di previsioni del comportamento futuro dell'applicazione e in questo caso l'adattività si dice *proattiva*.
Essendo l'adattamento un'operazione piuttosto lunga, è preferibile adottare un approccio proattivo, poichè se l'anomalia è prevista in anticipo si comincia da subito ad attuare operazioni correttive evitando l'eccessiva perdita di tempo che si avrebbe reagendo unicamente al verificarsi di cambiamenti. Ovviamente l'adattività proattiva non può sostituire totalmente quella reattiva, in quanto quest'ultima dev'essere sempre presente nel caso in cui nello svolgimento dell'applicazione si verifichino cambiamenti o errori non prevedibili.
In alcune situazioni, le applicazioni orientate ai servizi si comportano in maniera differente a seconda della situazione in cui si trovano. Ad esempio possono necessitare di cambiamenti nell'interfaccia utente nel caso si acceda da device diversi, a cambiamenti nelle funzionalità in caso di accesso da parte di utenti con diversi ruoli all'interno dell'azienda e conseguentemente diversi privilegi, cambiamenti nel livello di qualità richiesta a seconda dello scopo per il quale si accede al web service.
L'applicazione adattiva dovrà quindi modificare la propria interazione con l'utente, interpretando le sue volontà e venendo incontro alle sue esigenze funzionali e di qualità.

Il *contesto* è definito come ogni informazione che può essere utlizzata per rappresentare lo stato di un'entità. Un'entità è una persona, un luogo, o un oggetto che è considerato rilevante nell'interazione tra un utente e un'applicazione, inclusi l'utente e l'applicazione stessi.
E' necessario quindi fornire una rappresentazione formale del contesto e degli elementi che lo compongono. In generale, senza riferimento ad alcuno scenario specifico, le principali entità contestuali da modellare sono: *l'utente* che accede all'applicazione che può avere diversi ruoli e diversi permessi per eseguire operazioni all'interno del sistema (es. amministratore di sistema,

utente semplice ecc.), il *canale*, cioè l'insieme degli strumenti fisici che permettono all'utente di accedere all'applicazione caratterizzato dal device e dal canale per mezzo dei quali avviene la connessione con l'applicazione, *l'applicazione* stessa caratterizzata da un'interfaccia astratta attraverso la quale l'utente può interagire col sistema e connessa a servizi concreti, con specifiche proprietà ed operazioni ed offerti da diversi service provider e *l'ambiente* che rappresenta tutto ciò che non appartiene al sistema vero e proprio, ma ne condiziona il comportamento. A seconda dello specifico scenario di riferimento possono essere considerati altri parametri contestuali, detti *dipendenti dal dominio* significativi solo in relazione alla specifica applicazione.
I parametri contestuali possono essere ottenuti tramite *richiesta esplicita* all'utente mediante l'utilizzo di form, *monitorati* tramite l'utilizzo, il posizionamento e l' interrogazione di opportuni sensori o *derivati* da serie storiche di dati.

Con *granularità dell'informazione*, si indica il livello di dettaglio delle informazioni stesse. E' un problema significativo in particolare per quanto riguarda i dati monitorati, infatti in una rete di sensori è opportuno scegliere accuratamente il posizionamento dei sensori, quanto frequentemente interrogarli e quali dati raccogliere. Ovviamente una rete più densa e composta da sensori con una maggiore precisione fornirà dati migliori, ma a fronte di un costo significativo in termini di efficienza energetica. Per questo motivo è necessario individuare quali sono le informazioni da estrarre e a quale livello di dettaglio. E' necessario quindi estrarre conoscenza significativa da ogni dataset al giusto livello di dettaglio, ed avere delle opportune funzioni che permettano di lavorare con dati con diversa granularità.
Non tutti i dati sorgenti sono significativi, alcuni potrebbero non essere necessari per lo svolgimento dell'applicazione oppure già ottenuti da altre misurazioni precedenti. Anzichè catturare tutti i dati e scartare quelli non necessari è preferibile catturare solo quelli di interesse ed al giusto livello di dettaglio per l'applicazione in relazione allo specifico contesto di esecuzione.
Ogni granularità è suddivisa in granuli che possono essere uniti tra loro per

formare granuli più ampi appartenenti ad una granularità più grossa. In generale, la granularità dell'informazione viene salvata all'interno del modello del contesto di esecuzione dell'applicazione, in relazione ad ogni parametro per cui sia significativo considerarla. La granularità è definita come una classe avente metodi per individuare le granularità più grosse e più fini, un metodo per le trasformazioni tra granularità ed associata a un insieme di granuli e ad un dominio di riferimento.

Un'altra problematica trattata è quella riguardante la *granularità di monitoraggio* che descrive il periodo di tempo con il quale è significativo aggiornare i dati contestuali. Potrebbe accadere che in alcune situazioni, ad esempio di emergenza, sia necessario interrogare i sensori più spesso rispetto alla norma poichè un periodo di monitoraggio troppo lungo non permetterebbe la rilevazione di modifche contestuali significative, mentre in situazioni opposte, un periodo di monitoraggio troppo breve rappresenti unicamente un costo, in quanto non sia necessario allo scopo di monitorare cambiamenti rilevanti nel contesto di esecuzione.

In questo lavoro sarà descritto un *framework granulare e proattivo* per realizzare l'adattività. Il framework si occupa di utilizzare dati contestuali correlati con la loro granularità nell'ambito delle applicazioni adattive legate al contesto. E'descritto in relazione ad uno scenario di riferimento reale allo scopo di fornire un esempio concreto dei concetti descritti a livello teorico e di applicazione del framework. Il caso di studio scelto è quello della gestione delle scorte in un supermercato alimentare, in particolare per quanto riguarda il processo di profilazione granulare del fornitore e della previsione proattiva dell'andamento della domanda con conseguente aggiornamento della frequenza di monitoraggio dei dati.
I parametri relativi al contesto vengono salvati in una base di dati contestuale che descriva tutti i dati del dominio di riferimento in relazione, dove significativo, alla loro granularità, sia informativa che di monitoraggio. In particolare la granularità dell'informazione è rappresentata da una tabella che descriva il parametro ai diversi livelli di dettaglio (campi della tabella),

e quella di monitoraggio è salvata in un campo specifico della tabella del database a cui si riferisce.

La granularità informativa è sfruttata per lo svolgimento di sottoprocessi a diverso livello di dettaglio a seconda di opportuni parametri contestuali, rilevati dinamicamente.
Svolgere un sottoprocesso ad un livello di dettaglio troppo fine dove non necessario, potrebbe comportare uno sforzo operazionale più significativo, che non fornisca alcun valore aggiunto alla qualità o alle funzionalità del processo e fonte di inefficienze temporali ed energetiche.

La granularità di monitoraggio è aggiornata proattivamente a seconda di previsioni sul comportamento dell'applicazione, sulla base dell'analisi di serie di dati storici.
Nello scenario di riferimento ci si riferisce al processo di previsione dell'andamento della domanda e del conseguente aggiornamento della granularità di monitoraggio nel database.

La previsione della domanda viene effettuata utilizzando un tool di *machine learning*, una disciplina caratterizzata dallo studio e lo sviluppo di modelli e algoritmi che rendano i sistemi capaci di migliorare automaticamente le proprie performance durante l'esecuzione.
In particolare, vengono considerati gli *algoritmi di classificazione*, che consistono nella mappatura di un insieme di input in un insieme finito di modelli in output, allo scopo di imparare da serie di dati storici, creando un limite di separazione tra le classi, di modo che a fronte di nuovi input se ne possa determinare la classe di appartenenza.
Nello scenario di previsione proattiva della domanda si utilizza il tool di machine learning WEKA ed un opportuno algoritmo di classificazione, scelto a seguito della comparazione delle performance di diversi algoritmi.
Sulla base di parametri contestuali storici classificati (condizioni atmosferiche, stagione, presenza di vacanze, locazione, trend domanda) si crea un modello dell'applicazione, che è utilizzato per classificare l'istanza che descrive i parametri contestuali attuali. Una volta stimato se la domanda sarà

prevista in calo, crescita o costante sarà aggiornato opportunamente il parametro che descrive la granularità di monitoraggio del magazzino(domanda crescente, monitoraggio magazzino più frequente e viceversa).

Ogni processo è realizzato mediante la realizzazione di web service che svolgano specifiche operazioni, componendoli ed orchestrandoli tra loro utilizzando BPEL.

L'invocazione dinamica dei servizi è effettuata configurando un proxy ESB che permetta la rilevazione di parametri contestuali a runtime e la scelta del servizio idoneo da invocare dinamicamente.
Un ESB (Enterprise Service Bus) implementa una SOA attraverso uno strato software intermedio che descrive l'interazione tra gli attori che partecipano alla comunicazione e il servizio stesso. Offre un approccio dinamico fornendo una soluzione di integrazione e connettività basata su standard che consentano di creare e implementare interazioni tra applicazioni e servizi in modo immediato, riducendo il numero e la complessità delle interfacce. Le principali funzionalità di un ESB sono legate all'utilizzo di un layer di comunicazione tra servizi, che permette di risolvere problematiche legate all'integrazione, quali l'orchestrazione, la pubblicazione di servizi, l'intelligent routing dei messaggi verso diversi endpoint, la trasformazione dei dati, la sicurezza e l'integrazione tra diversi protocolli.
Un proxy service è un componente hardware o software posto solitamente come intermediario tra il browser dell'utente e il server, avente funzioni di filtro. In questo caso è configurato allo scopo di *mediare* le richieste dell'utente: l'utente accede al servizio mediante l'invocazione di uno dei metodi forniti dall'interfaccia del servizio astratto, e sarà il proxy ESB a scegliere, a seconda delle condizioni contestuali rilevate, quale servizio concreto invocare, automaticamente ed in maniera del tutto trasparente all'utente che si connette all'applicazione.
Allo scopo di invocare i servizi dinamicamente tramite un proxy ESB, vengono realizzati: un *servizio astratto* che mostri all'utente le funzionalità astratte offerte dal servizio, e una serie di *servizi concreti*, ognuno dei quali idoneo per

una reale situazione in cui può trovarsi il sistema. Questi ultimi implementano concretamente le funzionalità del servizio e si occupano dello svolgimento vero e proprio delle operazioni.
L'analisi proattiva della domanda è effettuata utilizzando il tool di machine learning WEKA ed un algoritmo di classificazione, allo scopo di prevederne le anomalie.

La tesi è strutturata come descritto in seguito.

Nel Capitolo 2 è descritto lo stato dell'arte, cioè le ricerche effettuate sull'argomento allo stato attuale. In particolare, viene proposta un'analisi e una descrizione delle architetture orientate ai servizi, dell'adattività, dell'analisi del contesto. Ci si riferisce anche al tema della granularità dell'informazione, trattato fino ad ora solo in ambiti differenti, allo scopo di individuarne un modello formale rappresentativo.

Nel Capitolo 3 è descritto il framework proposto, ponendo particolare attenzione sugli aspetti innovativi riguardanti l'utilizzo e la descrizione della granularità, sia informativa che di monitoraggio, nell'ambito delle applicazioni adattive legate al contesto. Viene descritta l'architettura del framework, individuando le principali componenti concettuali che entrano in gioco nella realizzazione di un'applicazione di questo tipo, ossia una componente per la realizzazione e la pubblicazione dei singoli web service, una per la loro orchestrazione, e una per la gestione dei dati a runtime composta da un tool per l'invocazione dinamica dei servizi e uno di machine learning. Viene fornita poi una descrizione formale della granularità sia informativa che di monitoraggio e del loro utilizzo, correlato con il concetto di proattività, all'interno del framework.

Nel Capitolo 4 viene descritto uno scenario reale di applicazione del fra-

mework allo scopo di mostrarne un esempio concreto di utilizzo e di esemplificare i concetti trattati in maniera teorica nel capitolo precedente. Lo scenario scelto è quello della gestione delle scorte a magazzino in un supermercato alimentare.

Nel Capitolo 5 viene descritta l'implementazione del framework, in riferimento allo scenario, mostrando le tecnologie utilizzate per la realizzazione di ogni componente concettuale. In particolare vengono descritte la creazione e la progettazione del database concettuale arricchito delle informazioni sulla granularità, le funzionalità fornite dai web service implementati, la configurazione del proxy ESB per l'invocazione dinamica dei servizi, l'analisi proattiva del contesto mediante l'utilizzo di un algoritmo di classificazione implementato con il supporto del tool WEKA di machine learning, e l'orchestrazione dei web service effettuata con BPEL.

Capitolo 2

Stato dell'arte

Nell'era dell' Internet of Services, le applicazioni orientate ai servizi (SOA) sono considerate una delle tecnologie più promettenti in quanto sono in grado di offrire funzionalità complesse e flessibili operando in ambienti distributi e fortemente dinamici mediante la composizione di servizi eterogenei e forniti da diversi service provider. Spesso i servizi che compongono l'applicazione non sono sotto il controllo diretto dello sviluppatore del sistema, ma vengono semplicemente utilizzati per sfruttarne le funzionalità offerte. Da un lato ciò rappresenta un notevole vantaggio in termini di sforzo implementativo, ma dall'altro introduce una dipendenza critica tra l'applicazione e il servizio stesso. Quest'ultimo, infatti, potrebbe essere soggetto a cambiamenti o diventare non disponibile improvvisamente. Per questo motivo è necessario che le SOA siano dotate di meccanismi di adattività in modo da reagire ad eventi critici ed imprevisti che si possono verificare durante l'esecuzione (ad esempio errori a runtime, cambiamenti inaspettati da parte dei service provider o cambiamenti nel contesto di esecuzione). I sistemi adattivi devono essere in grado di adattarsi al cambiamento dinamicamente e automaticamente, riducendo il più possibile l'intervento umano. Il comportamento di un'applicazione adattiva non è unicamente controllato dagli input dell'utente, ma anche da informazioni riguardanti l'applicazione e il contesto associato. I metodi e gli strumenti utilizzati per la realizzazione di una tale applicazione

devono essere quindi in grado, una volta raccolti i dati dell'applicazione e del suo contesto, di agire e prendere decisioni in accordo.

2.1 Applicazioni orientate ai servizi (SOA)

Un Web Service [8] è un sistema software identificato da un'URL e le cui interfacce pubbliche sono definite e descritte utilizzando XML. La definizione di un Web Service può essere utilizzata all'interno di sistemi software più complessi, con i quali interagisce scambiando messaggi XML trasmessi tramite opportuni protocolli internet (SOAP, HTTP, ecc). I Web Service vengono pubblicati in appositi registri.

Il *web service model* consiste di tre entità principali:

- *service provider:* rappresenta l'azienda che crea o semplicemente fornisce il servizio. Il service provider deve descrivere il servizio in un formato standard, XML, e pubblicarlo in un service registry centrale;
- *service registry:* contiene la descrizione dei servizi e informazioni aggiuntive sul service provider, come l'indirizzo e i contatti dell'azienda, e dettagli tecnici sul servizio fornito;
- *service consumer:* è colui che usufruisce del servizio. Recupera le informazioni dal registro e usa la descrizione del servizio ottenuta per invocare e utilizzare il web service.

L'utilizzo combinato dei Web Services dà luogo ad un'architettura fortemente disaccoppiata e orientata ai servizi.

Gli standard per l'utilizzo e la definizione dei WS sono i seguenti:

- *WSDL (Web Service Description Language):* utilizza il formato XML per la descrizione dei metodi forniti da un Web Service, inclusi i parametri in input e output, i tipi dei dati e il protocollo utilizzato per lo scambio di messaggi;

- *UDDI (Universal Description, Discovery and Integration):* è il principale service registry che permette la pubblicazione e la ricerca dei servizi arricchiti di informazioni sul service provider e sul WS stesso;
- *SOAP (Simple Object Across Protocol):* è utilizzato per lo scambio di informazioni formattate in XML tra le entità coinvolte nel web service model.

2.1.1 Cambiamenti nei Web Service

I web service sono soggetti a numerosi cambiamenti [22] , in particolare quando vengono considerati in relazione all'ambiente in cui operano. L'ambiente consiste di numerosi stakeholder che influenzano e controllano il ciclo di vita del servizio, che risulta esserne l'elemento centrale. Per questo motivo è importante porre particolare attenzione sui fattori che influenzano il comportamento di un servizio. L'evoluzione di un web service comprende: il cambiamento nei requisiti, modifiche nell'implementazione, cambiamenti nella semantica, cambiamenti nell'utilizzo. Queste attività sono provocate da differenti stakeholder, come gli sviluppatori, i fornitori del servizio, l'utente finale o coloro che integrano il sistema.

Stakeholder che influenzano un servizio

In questa fase vengono individuati gli stakeholder che sono interessati nell'utilizzo e sviluppo del servizio e le attività ad essi correlate. Il *provider* è il responsabile dell'ideazione e progettazione del servizio; si occupa della definizione dei requisiti, della negoziazione dell'SLA con il cliente e delle politiche di prezzo. Il *developer* è colui che si occupa dell'implementazione del servizio; si occupa della gestione dei cambiamenti dell'interfaccia, delle diverse versioni dei servizio e di tenere traccia dei cambiamenti. Il *service integrator* ha il compito di integrare servizi esterni in un unico sistema software allo scopo di sfruttare le caratteristiche tecniche dei servizi. Sono interessati in cambiamenti nell'interfaccia, negli attributi di qualità e in quelli semantici.

L'*utente* è colui che utilizza il servizio ed è quindi interessato in cambiamenti nelle funzionalità da un punto di vista non tecnico. Specifica i requisiti funzionali e definisce gli attributi di qualità che il servizio deve soddisfare. Il *broker* gestisce le informazioni di diversi servizi in un registro pubblicamente disponibile per la ricerca e la navigazione. Fornisce le informazioni per la ricerca di un particolare web service all'interno dei registri.

Tipologie di cambiamenti

I cambiamenti possono essere statici e verificarsi a design-time, oppure dinamici ed essere osservati a runtime. Possono verificarsi *cambiamenti nei requisiti* che costituiscono il principale fattore di cambiamento. I requisiti rappresentano una sorta di benchmark per monitorare il corretto funzionamento del web service implementato. I *cambiamenti nell'interfaccia* descrivono modifiche nelle operazioni fornite dal servizio e nella struttura dei messaggi. In particolare, rappresentano l'aggiunta di nuove funzionalità al servizio oppure l'aggiornamento di quelle esistenti. I *cambiamenti nell'implementazione* del web service sono strettamente correlati ai cambiamenti dell'interfaccia. Sono causati da ottimizzazioni o modifiche del codice o da cambiamenti nelle funzionalità. I *cambiamenti nella QoS* sono gli unici ad essere monitorati unicamente a runtime e dipendono da modifiche delle proprietà dei web service. Sono caratterizzati da parametri come la qualità dei dati, le performance del sistema e da fattori esterni.

2.1.2 Composizione di servizi

L'infrastruttura di base dei web service consiste nella semplice interazione tra utente e web service. Se l'implementazione della logica di un'applicazione consiste nell'invocazione di altri web service è necessario combinare le funzionalità di diversi servizi. In questo caso si parla di *servizi composti* e il processo di sviluppo di un servizio composto è detto *composizione di servizi*. I servizi composti sono ricorsivamente definiti dall'aggregazione di servizi elementari e composti (Khalaf and Leymann, 2003).

Questo permette la definizione di applicazioni sempre più complesse, aggregando progressivamente componenti ad un alto livello di astrazione. Un cliente che invoca un servizio composto, può essere considerato a sua volta un web service. Per la composizione ed orchestrazione di web service diventa necessario sviluppare un'applicazione per la loro gestione. Il prodotto software maggiormente usato è BPEL4WS (Business Process for Execution Language for Web Services). I web service sono descritti precisamente nelle loro funzionalità, interfacce e protocolli supportati.

Modello per la composizione di servizi

Vengono individuate sei componenti principali da modellare:

- *modello dei componenti:* assunzioni su cosa un componente è o non è;
- *modello di orchestrazione:* definisce astrazioni e linguaggi per descrivere l'ordine in cui, e le condizioni sotto le quali un web service viene invocato;
- *modello dei dati e dell'accesso ai dati:* definisce come i dati sono descritti e scambiati tra i componenti;
- *modello di selezione del servizio:* si riferisce a come un servizio è selezionato, staticamente a design-time o dinamicamente a runtime;
- *transazioni:* descrive quale semantica transazionale può essere associata alla composizione;
- *modello per la gestione dele eccezioni.*

La composizione di servizi può avvenire manualmente o in maniera automatica, basandosi sulle ontologie.

2.2 Adattività

I sistemi software distribuiti, che hanno a che fare con ambienti soggetti a frequenti cambiamenti, normalmente richiedono la supervisione umana per continuare correttamente lo svolgimento delle operazioni durante queste condizioni di cambiamento. Queste attività di mantenimento sono molto costose e comportano una notevole perdita di tempo durante la fase operativa del sistema. Le applicazioni adattive [21] sono la risposta a questo tipo di problematica: esse hanno la capacità di adattarsi da sole ai cambiamenti durante lo svolgimento delle operazioni. Questi cambiamenti possono verificarsi all'interno dell'applicazione stessa, oppure nel contesto di esecuzione. Un sistema di questo tipo deve quindi *monitorare* se stesso e il contesto circostante, *rilevare* i cambiamenti significativi, *decidere* come reagire e *mettere in atto* le decisioni prese. La definizione di software adattivo fornita in letteratura è la seguente:

> I sistemi self-adattivi modificano il proprio comportamento in risposta a cambiamenti nel proprio ambiente di esecuzione. Con ambiente di esecuzione, intendiamo qualsiasi cosa osservabile dal sistema, come gli input dell'utente finale, i dispositivi hardware esterni e i sensori, o gli strumenti programmativi [Oreizy et al. 1999]

L'adattività è un argomento affrontato dal punto di vista di diverse discipline come la progettazione del software, le architetture orientate ai servizi, la gestione della QoS, l'intelligenza artificiale, il machine learning e il soft computing.

2.2.1 Adattività a design-time

Le strategie di adattamento, come descritto in [3], possono essere programmate a design time, associandole ad eventi che potrebbero verificarsi ed essere rilevati nella fase di esecuzione. E'necessario garantire che il comportamento

dell'applicazione sia sempre allineato con gli eventuali cambiamenti che si verificano nell'ambiente circostante. Una SBA (Service-based Application) adattiva non deve unicamente essere in grado di soddisfare alcuni requisiti funzionali, ma deve anche definirne di nuovi in termini di monitoraggio e adattività.

I *requisiti di monitoraggio* consistono nella necessità di identificare alcune situazioni che devono essere segnalate, in quanto potrebbero determinare un'esigenza di adattamento. Queste situazioni vengono monitorate a runtime da un *monitoring engine* che basandosi su ciò che osserva emetterà alcuni *eventi monitorati*.

I *requisiti di adattività* sono soddisfatti mediante opportune *strategie di adattamento*, che saranno eseguite all'interno del *processo di adattamento* che è effettuato a seguito del verificarsi degli *eventi monitorati* o da stimoli contestuali esterni.

Strategie di adattamento

Per la stessa SBA possono essere adottate diverse strategie di adattamento, poichè ognuna ha differenti funzionalità, caratteristiche e conseguenze ed è utilizzabile per far fronte ad uno specifico cambiamento e può essere strettamente legata al contesto o ai requisiti funzionali e non funzionali. La selezione della strategia di adattamento più idonea può essere un problema complesso a causa delle diverse caratteristiche del sistema da considerare. Quando una SBA è in esecuzione potrebbero verificarsi diversi cambiamenti nell'ambiente che potrebbero causare inefficienze. Per evitare il degrado delle performance dell'applicazione è necessario identificare la strategia di adattamento più idonea, che sia in grado di mantenere allineato il comportamento dell'applicazione con il contesto e i requisiti di sistema. Come descritto in [3] è possibile distinguere tra:

- *strategie indipendenti dal dominio:* applicabili in quasi tutti i contesti esecutivi;

- *stategie legate al dominio:* limitate a uno specifico ambiente di esecuzione.

Principali strategie di adattamento indipendenti dal dominio

- *sostituzione del servizio:* riconfigurazione di un SBA con la sostituzione dinamica di un servizio con un altro;
- *ri-esecuzione:* la possibilità di tornare indietro in un certo punto del processo definito come sicuro per svolgere di nuovo lo stesso set di attività, oppure per svolgerne un set alternativo;
- *ri-negoziazione:* semplice terminazione del servizio in esecuzione e ri-negoziazione del Service Level Agreement (SLA) per la gestione della riconfigurazione delle attività da parte del provider;
- *ri-composizione:* riorganizzazione e riconfigurazione del flusso di esecuzione che lega i diversi servizi componenti nell'applicazione;
- *compensazione:* definizione di attività ad hoc che possono riparare gli effetti di un processo che viene completato in maniera errata;
- *evoluzione:* inserimento di eccezioni nel workflow capaci di attivare l'evoluzione dell'applicazione;
- *salvare e aggiornare le informazioni di adattamento:* salvare tutte le informazioni riguardanti le attività adattive per differenti scopi;
- *fail:* il sistema reagisce al cambiamento salvando lo stato del sistema, causando il fallimento dell'escecuzione del servizio e ri-eseguendolo.

Trigger di adattamento

L'adattamento in una SBA può essere causato da una varietà di fattori, detti *trigger*. I trigger possono essere di due tipologie, legati ai servizi componenti o contestuali [3].

Trigger legati ai servizi componenti dell'SBA

- cambiamenti nelle *funzionalità del servizio:* variazione dell'interfaccia del servizio, variazione del protocollo di interazione, errori a runtime;
- cambiamenti nella *qualità del servizio:* disponibilità del servizio, degradazione dei parametri di QoS, violazioni sell'SLA, calo della service reputation.

Trigger contestuali

- cambiamenti nel *contesto di business:* cambiamento nelle agile service network (ASN), nuove regole e politiche aziendali;
- cambiamenti nel *contesto computazionale:* utilizzo di dispositivi, protocolli, reti differenti;
- cambiamenti nel *contesto dell'utente:* differenti profili utenti con diversi privilegi, ambiente sociale e parametri fisici (locazione, tempo).

Ogni trigger può essere associato ad un insieme di strategie di adattamento idonee per ri-allineare l'applicazione con i cambiamenti del sistema o del contesto.

Per scegliere la strategia di adattamento più idonea è necessario tener conto anche di:

- *scope del cambiamento:* può caratterizzare una sola istanza oppure influenzare l'intero modello;
- *impatto del cambiamento.*

Il design di un'applicazione adattiva deve quindi attraversare le seguenti fasi:

- modellare i trigger di adattamento;
- realizzare le strategie di adattamento;
- associare le strategie di adattamento ai trigger.

2.2.2 Life-cycle

S-Cube (the European network of Excellence on Software Services and Systems) propone una descrizione del ciclo di vita di un'applicazione adattiva [14]. Innanzitutto è necessario distinguere tra i concetti di:

- *evoluzione:* è considerata come la modifica dei requisiti, le specifiche o i modelli del sistema a design-time;
- *adattività:* è considerata come la modifica di una specifica istanza di un sistema orientato ai servizi durante lo svolgimento delle operazioni a run-time.

Il ciclo di vita di una Service Based Application (SBA) proposto da S-Cube è composto da due cicli principali:

- *ciclo di evoluzione e sviluppo:* è composto dalle classiche fasi della progettazione e sviluppo di un'applicazione;
- *ciclo di adattività ed esecuzione delle operazioni:* estende il classico ciclo di vita esplicitando le fasi per la gestione adattiva dei cambiamenti durante la fase operativa della SBA.

I due cicli devono coesistere e non essere in conflitto tra loro e permettono di distinguere i casi in cui è necessaria una nuova riprogettazione del sistema allo scopo di far fronte a cambiamenti, da quelli in cui l'adattamento può essere effettuato a runtime.

Ciclo di evoluzione e sviluppo

- *Analisi dei requisiti:* i requisiti funzionali e di qualità sono individuati e documentati. E'una fase molto importante nelle SBA per la natura fortemente dinamica che le caratterizza;
- *Design:* sono specificate le attività e il flusso di controllo dell'applicazione. Nel caso di SBA questo coincide con la creazione di un workflow

usando linguaggi come BPEL. In questa fase si individuano anche le strategie di adattamento e i meccanismi per permettere all'applicazione di reagire nel caso di un'esigenza di adattamento;

- *Realizzazione:* costruzione dell'applicazione mediante l'integrazione e la coordinazione dei servizi forniti dai diversi service provider;
- *Deployment:* comprende tutte le attività necessarie per rendere la SBA accessibile all'utente.

Ciclo di adattività ed esecuzione delle operazioni

- *Esecuzione delle operazioni e gestione:* in questa fase sono specificate tutte le attività necessarie per svolgere le operazioni e gestire una SBA. In letteratura ciò prende il nome di governance. Nello scenario di riferimento vengono identificati i problemi che si potrebbero riscontrare durante l'esecuzione e i possibili cambiamenti nel contesto;
- *Identificazione delle esigenze di adattamento:* è responsabile di decidere se la SBA ha bisogno di essere adattata per mantenere la qualità e le funzionalità attese o meno. Queste decisioni possono essere prese automaticamente o richiedere l'intervento umano, e in modo reattivo o proattivo;
- *Identificazione della strategia di adattamento:* selezione della strategia di adattamento migliore (SLA re-negotiation, service substitution ecc). L'identificazione della strategia di adattamento più adatta è spesso supportata da meccanismi di reasoning;
- *Attuazione dell'adattamento:* dopo aver scelto la strategia di adattamento, il corrispondente meccanismo è utilizzato per mettere in atto l'adattività.

2.2.3 Selezione del servizio

E' basata sulla distinzione tra *servizi astratti* e *servizi concreti*. I servizi astratti rappresentano un'interfaccia che implementa diversi servizi concreti. L'utente seleziona un servizio astratto e lo invoca; la decisione su quale servizio concreto invocare sarà presa a runtime dall'applicazione adattiva sulla base dell'analisi di parametri contestuali e di esecuzione. Il servizio scelto sarà quello che soddisfa in modo migliore i vincoli imposti dal contesto in cui l'invocazione ha luogo.

2.2.4 Composizione statica e dinamica di Web Service

Le strategie di composizione statica e dinamica si riferiscono all'istante in cui la composizione ha luogo. La *composizione statica* avviene a design-time quando viene progettata l'architettura e il design del sistema software viene pianificato. I componenti da utilizzare sono scelti, connessi tra loro e infine compilati e realizzati. Questo può essere sufficiente quando si sta lavorando in un ambiente non, o molto raramente, soggetto a cambiamenti. La composizione statica risulta quindi essere troppo restrittiva, poichè i componenti dovrebbero essere in grado di adattarsi automaticamente a cambiamenti improvvisi nell'ambiente. In un'architettura orientata ai servizi l'ambiente di riferimento è dinamico: nuovi servizi diventano disponibili/indisponibili giornalmente e il numero di fornitori di servizi cresce costantemente. Idealmente il processo dovrebbe essere in grado di adattarsi a qualunque cambiamento e ai requisiti dell'utente, riducendo al minimo l'intervento umano. Per questo motivo in un ambito di questo tipo ci si focalizza sulla *composizione dinamica*. Essa avviene a runtime e si compone di quattro fasi principali:

- i service provider pubblicano i loro servizi in un registro web;
- il Service Composition Engine decompone i requisiti dell'utente in un servizio astratto e li invia tramite una richiesta SOAP al registro per individuare i servizi adeguati;

- il registro web ritorna una serie di servizi concreti;
- il service composition engine invia una richiesta SOAP ad un servizio concreto.

La composizione di web service richiede una descrizione semantica dei servizi, per permettere ai servizi di interagire tra loro. In particolare, vengono definite due tipologie di regole di composizione: quelle sintattiche e quelle semantiche. Le regole *sintattiche* descrivono le modalità della composizione di servizi. Le regole *semantiche* comprendono la composizione dei messaggi, delle operazioni semantiche e la composizione dei requisiti di qualità.

2.2.5 Composizione di workflow

Nell'articolo [19] sono considerate le *applicazioni legate al workflow* e più in generale le applicazioni sviluppate utilizzando tecniche di composizione. Vengono definite delle *regioni sensibili al contesto*, ossia quelle parti del processo di business che hanno differenti comportamenti a seconda del contesto di esecuzione. Ogni regione è associata a diverse configurazioni, corrispondenti a diversi sottoprocessi, che rappresentano comportamenti differenti. Una volta che una regione è istanziata, una specifica configurazione tra quelle possibili è selezionata sulla base del contesto. Nella progettazione dell'applicazione è necessario definire una condizione di entrata per ogni regione. E'importante determinare quali sono i principali schemi che identificano un comportamento comune seguito dall'utente al quale è fornito un servizio durante il flusso di esecuzione. I principali schemi individuati sono:

Dipendenza dal dispositivo: rappresenta una situazione in cui un utente cambia dispositivo, ma sta ancora completando il processo di business;

Scelta determinata dalla QoS: è possibile avere diverse configurazioni del processo di business a seconda di differenti livelli di QoS;

Scelta determinata dalla locazione: rappresenta una situazione in cui le configurazioni del processo di business sono associati alla locazione dell'utente;

Utente online/offline: a causa di vincoli di consumo delle risorse l'utente potrebbe preferire lavorare offline. Questo scenario è tipico per un utente che si connette attraverso un dispositivo mobile.

2.2.6 Qualità del servizio e Service Level Agreement

La *qualità del servizio (QoS)* [16] gioca un ruolo cruciale nei sistemi orientati ai servizi, ad esempio durante la selezione del servizio da invocare. Quando diversi servizi esterni vengono integrati all'interno di un processo di business, è importante considerare la qualità garantita dal sevice provider.

Il *Service Level Agreement (SLA)* è un contratto esplicito che deriva da un processo di negoziazione tra il fornitore e il consumatore del servizio per definire la qualità, e a volte alcune proprietà funzionali, che devono essere garantite durante l'utilizzo del servizio. In generale, gli attributi di qualità possono essere classificati in deterministici e non deterministici. I primi indicano che gli attributi di QoS sono noti prima che un servizio sia invocato (ad esempio prezzo, sicurezza), mentre i secondi includono tutti gli attributi che vengono rilevati solamente durante l'invocazione del servizio (ad esempio tempo di risposta, disponibilità).

2.2.7 Adattività reattiva vs Adattività proattiva

La maggior parte dei lavori riguardanti l'adattività si concentrano sull'adattività di tipo *reattivo*. Questo significa che il meccanismo adattivo entra in gioco dopo che si verifica una situazione critica o anomala. E' quindi basata sul monitoraggio, allo scopo di verificare se vengono effettuate violazioni nei requisiti. Poichè l'adattività è un'attività lunga, per evitare troppi ritardi nello svolgimento del processo, o interruzioni a runtime, è preferibile che il

rilevamento di una situazione critica e la messa in atto di meccanismi correttivi siano effettuati prima che l'anomalia si verifichi. Si parla in questo caso di *adattività proattiva*, messa in atto per evitare le conseguenze del verificarsi di un potenziale, futuro cambiamento/problema. Ovviamente l'adattività proattiva deve essere affiancata, e non totalmente sostituita, a quella reattiva, che deve essere sempre presente in caso di cambiamenti improvvisi e non prevedibili.

P^2SRD framework

Il framework P^2SRD (Pull and Push Runtime Service Discovery) [26] permette la scelta del servizio da invocare proattivamente e in parallelo all'esecuzione delle operazioni, per le situazioni in cui si verificano dei cambiamenti nelle caratteristiche comportamentali, strutturali, contestuali o di qualità dei servizi, oppure nel contesto di esecuzione o quando un servizio migliore diventa disponibile. Il framework gestisce sia il processo di selezione *pull*, che è eseguito in seguito alla necessità di sostituire il servizio da invocare a seguito di un malfunzionamento o di un cambiamento, che quello *push* eseguito in maniera proattiva. E'dovuto a cambiamenti nel contesto, nel servizio o alla disponibilità di nuovi servizi.

PREvent framework

Per quanto riguarda l'adattività proattiva il framework PREvent [18], una parte del progetto VRESCo, individua tre fasi principali:

- monitoraggio dei dati a runtime;
- predizione di violazioni;
- identificazione di possibili azioni di adattamento e messa in atto.

Nella fase di monitoraggio dei dati a runtime vengono monitorati sia i dati interni al sistema che quelli esterni. Alcuni dati sono già pronti per l'analisi, altri devono essere derivati. Nella fase di predizione delle violazioni vengono

predetti dei valori di SLO (Service Level Objective) in alcuni checkpoint definiti opportunamente. A questo punto vengono fatte delle analisi relative a serie di dati storici e attraverso tool di machine learning per creare un modello del sistema ed effettuare predizioni su opportuni parametri. Alcuni dati non saranno disponibili al checkpoint e per questi se ne fornisce una stima. Le strategie di adattamento più adeguate sono poi selezionate e messe in atto.

2.2.8 VRESCo (Vienna Runtime Enviroment for Service-oriented Computing)

Il framework VRESCo [11] [17] [16] [15] considera molte delle caratteristiche distintive di un'architettura orientata ai servizi, sottolineando l'importanza dell'utilizzo di registri per il supporto alla selezione dinamica e all'invocazione dei servizi. I servizi vengono pubblicati dai fornitori e i clienti possono accedere ai dati salvati all'interno dei registri mediante uno specifico linguaggio di interrogazione VQL (Vresco Query Language). Viene fornita una distinzione tra caratteristiche astratte e concreta implementazione dei servizi, che sono opportunamente raggruppati in accordo con le loro funzionalità. Un formato astratto dei messaggi permette al framework di *mediare* tra i servizi che forniscono le stesse funzionalità, ma utilizzano una sintassi differente. Per ogni concreta implementazione di un servizio viene descritto come gli elementi dell'interfaccia astratta corrispondano agli elementi dell'interfaccia concreta. Anche l'*invocazione* dei servizi avviene dinamicamente, attraverso il componente DAIOS [12]: l'utente invia una richiesta ad un servizio astratto e VRESCo si occuperà trasparentemente di dirigere la richiesta dell'utente verso un endpoint concreto. VRESCo supporta il *service versioning*: i servizi evolvono e cambiano frequentemente, ed è importante che una nuova versione del servizio possa essere pubblicata e facilmente integrata all'interno del processo esistente. Inoltre, la descrizione dei metadati tiene in considerazione, oltre agli attributi funzionali, anche quelli relativi alla *QoS*. La *selezione* del servizio da invocare viene effettuata sulla base degli attributi di qualità.

Modello dei metadati e del servizio

Il modello dei metadati descrive le funzionalità offerte dai web service in maniera astratta. Una *categoria* descrive una funzionalità generale del servizio. Contiene un numero arbitrario di *caratteristiche*, che descrivono un'azione concreta nel sistema. Il modello dei metadati permette inoltre la specifica di opportune *precondizioni e postcondizioni* che devono essere soddisfatte quando viene eseguita una caratteristica. Entrambe le tipologie di condizioni sono associate a *predicati*, caratterizzati a loro volta da *attributi*.

Il modello del servizio costituisce le informazioni riguardo la gestione del servizio concreto. Un servizio è disponibile in una o più *revisioni*. E' fornita una *funzione di mapping* che fornisce le corrispondenze tra servizi concreti e la loro interfaccia astratta. Oltre alle caratteristiche funzionali dei servizi, VRESCo considera anche gli attributi non funzionali, legati alla QoS.

VQL (Vresco Query Language)

VQL costituisce un linguaggio per la richiesta di dati contenuti nel registro. Il linguaggio di interrogazione si riferisce alle entità e relazioni descritte nel modello dei dati del framework. Le query possono essere obbligatorie, ottenute mediante la funzione *Add* oppure facoltative, definite dalla funzione *Match*. Vengono individuate tre strategie di interrogazione:

- *ESATTA*: tutte le query sono considerate come obbligatorie;
- *CON PRIORITA'*: ogni query facoltativa è dotata di un peso numerico che ne definisce la priorità;
- *RILASSATA*: è un caso particolare del meccanismo con priorità, in cui il peso di ogni query facoltativa è 1.

Service versioning

Si occupa della gestione dell'evoluzione del servizio e della documentazione dei cambiamenti nell'interfaccia. Ad ogni servizio sono associate una o più

revisioni. Le relazioni tra le diverse revisioni di un servizio sono descritte nel *grafo delle versioni di un servizio*, dove i nodi costituiscono le revisioni e un arco direzionato punta da una revisione alle successive. Le revisioni possono essere etichettate a seconda delle caratteristiche come iniziali, stabili ecc.

Qualità del servizio ed SLA

In VRESCo viene fornito un meccanismo di monitoraggio dei requisiti di qualità sia lato server che lato client [16], cercando di combinare i vantaggi di entrambi gli approcci. Il monitoraggio *lato server*, fornito in VRESCo dal WPC (Windows Performance Counter), è solitamente più accurato, ma richiede l'accesso all'implementazione del servizio, che non sempre è disponibile. Al contrario, quello *lato client*, fornito nel framework dal tool QUATSCH, è indipendente dalla specifica implementazione del servizio, ma misura valori non sempre aggiornati.

In VRESCo ogni servizio è arricchito di obbligazioni riguardanti l'SLA. Il VRESCo Event Engine si occupa di verificare le corrispondenze tra obbligazioni ed eventi. Se si verificano delle anomalie viene notificata una violazione nel SLA.

2.3 Contesto

I tradizionali sistemi software sono solitamente progettati per soddisfare un insieme di requisiti all'interno di uno specifico contesto di esecuzione. Anche se l'evoluzione dei requisiti e del contesto di esecuzione sono considerati possibili si assume che ciò inciderà sulle versioni successive del sistema software, non sull'istanza corrente che si sta eseguendo.

Questa assunzione non è più valida quando consideriamo un'applicazione orientata ai servizi, dinamica e costruita mediante la composizione di servizi disponibili in rete. In questo tipo di applicazioni l'evoluzione del contesto e dei requisiti rappresenta la norma, non l'eccezione, e per questo motivo è necessario che il sistema sia in grado di evolversi possibilmente durante l'ese-

cuzione e senza necessariamente iniziare un processo completamente nuovo di progettazione e sviluppo. In questa sezione consideriamo il ruolo del *contesto* nelle attività di adattamento. Il contesto è descritto da diverse caratteristiche come la situazione in cui l'utente accede all'applicazione (locazione, privilegi utente, tempo) , e le proprietà di esecuzione dell'applicazione stessa e dei servizi componenti [4]. In letteratura il contesto viene definito come:

> "Il contesto è rappresentato da ogni informazione che può essere utilizzata per caratterizzare lo stato di un'entità. Un'entità è una persona, un luogo, o un oggetto che è considerato rilevante per l'interazione tra un utente e un'applicazione, inclusi l'utente e l'applicazione stessi". [Dey and Abowd (2000)]

Nella fase di progettazione di un'applicazione adattiva è necessario quindi fornire un modello del contesto di riferimento.

2.3.1 Modello del contesto basato su XML

Il modello proposto in [4], è descritto da una rappresentazione XML di tutte le dimensioni contestuali che possono causare un'esigenza di adattamento. Contiene sei dimensioni principali:

- *Tempo:* informazioni riguardanti il momento in cui l'utente accede all'applicazione;
- *Ambiente:* informazioni legate a fattori spaziali;
- *Utente:* informazioni riguardo i privilegi, ruoli, preferenze dell'utente;
- *Servizio:* informazioni riguardo i servizi componenti l'applicazione e il loro stato;
- *Contesto di business:* tiene in considerazione i fattori di business dell'applicazione;

- *Contesto computazionale:* specifica le caratteristiche hardware e software mediante le quali viene effettuato l'accesso da parte dell'utente.

Diverse dimensioni contestuali saranno rilevanti per applicazioni tra loro differenti. I requisiti contestuali di un'applicazione, oltre a considerare le sei dimensioni descritte dovranno essere arricchiti da dimensioni rilevanti solo per lo specifico dominio di riferimento. Il processo di modellazione del contesto è quindi caratterizzato da:

- l'istanziazione del generico modello contestuale relativo all'applicazione;
- la descrizione delle relazioni tra gli elementi della SBA e le dimensioni contestuali.

Processo adattivo legato al contesto

Una volta che il modello del contesto dell'applicazione è stato definito è necessario individuare quando i cambiamenti contestuali risultano essere critici per il corretto funzionamento della SBA e come è possibile reagire per far fronte a queste criticità. Per fare ciò devono essere progettati e realizzati una serie di *monitor* capaci di individuare i cambiamenti nelle dimensioni contestuali e una piattaforma che sulla base dell'analisi del contesto definisca per ogni cambiamento monitorato quale strategia di adattamento adottare.

2.3.2 Modello del contesto semantico

In [6] viene proposto un approccio basato sulle ontologie per la rappresentazione del contesto. Esse sono in grado di descrivere i concetti del dominio e le loro relazioni usando un linguaggio interpretabile da una macchina (OWL, Web Ontology Language).

Ontologia di base

Generalmente un profilo utente descrive una serie di requisiti espressi nella forma
$< attributo, valore >$, dove il valore esprime uno specifico utente o una classe di utenti. Ogni requisito può essere:

- *esplicito:* il valore dell'attributo è specificato dall'utente
- *implicito:* il valore dell'attributo non è specificato direttamente dall'utente, ma è ricavato sulla base di meccanismi di reasoning o dall'analisi di dati storici

La *top ontology* specifica i concetti generali del contesto, indipendentemente dal dominio di riferimento. In particolare, il contesto è composto da tre elementi principali: il profilo utente, l'ambiente e il canale.

Il *profilo utente* descrive le caratteristiche dell'utente che accede all'applicazione e può essere distinto in parametri dipendenti e indipendenti dal dominio.

L'*ambiente* raccoglie le informazioni riguardanti la posizione geografica, le condizioni ambientali, i dettagli temporali e tutte le azioni che caratterizzano l'interazione dell'utente con lo spazio circostante.

Il *canale* descrive gli elementi che descrivono l'interazione dell'utente con la piattaforma usata per accedere all'applicazione: il canale, la rete, l'interfaccia di rete e il protocollo applicativo.

Per ogni sottoclasse inclusa in ognuno dei tre elementi descritti viene fornita un'ontologia che la descriva. Una grande varietà di dati è utilizzata e salvata per ricavare un modello significativo del contesto. Per questo motivo i dati devono essere associati ad opportuni *metadati*, per sottolineare le eterogeneità e per formalizzare il loro utilizzo e la loro gestione. I metadati possono essere distinti in tre classi:

- *build-time:* creati ed utilizzati durante la fase di progettazione e costruzione del sistema. Si distinguono a loro volta tra persistenti e legati al tempo;

- *metadati di controllo:* sono utilizzati nella fase di esecuzione e servono per la gestione delle politiche di sicurezza;
- *metadati di utilizzo:* supportano l'utente nell'utilizzo e nella comprensione dei dati.

Framework per la gestione del contesto

Lo scopo di questo framework è quello di sottolineare le funzionalità essenziali che deve avere un generico sistema informativo legato al contesto, allo scopo di gestire il contesto utilizzando il modello basato sulle ontologie descritto in precedenza. Questo framework deve essere composto da tre componenti principali:

- *gestore del contesto globale:* fornisce al sistema le funzionalità utilizzate per ragionare sul contesto (sia a runtime che sulla base di dati storici) e per gestire la comunicazione tra gli agenti che deve coordinare;
- *repository:* contiene la top ontology, i metadati e le regole per la gestione del contesto;
- *gestori del contesto degli agenti:* fornisce funzionalità lato agente per la gestione del contesto e la comunicazione con il sistema informativo.

2.3.3 MAIS (Multichannel Adaptive Information Systems)

Il progetto MAIS [20, Part I] ha lo scopo di fornire un supporto per l'esecuzione flessibile e adattiva di un'applicazione mobile in un ambiente altamente dinamico, distribuito e multicanale. In un contesto di questo tipo è importante essere in grado di descrivere la continua evoluzione dell'ambiente e delle caratteristiche dell'utente. Il contesto viene modellato come composto dall'aggregazione di quattro componenti principali:

- *il canale:* che definisce il modo con cui l'utente interagisce con l'applicazione;
- *la locazione e il tempo:* definisce dove e quando l'utente accede all'applicazione;
- *il profilo utente:* caratterizza chi sta interagendo con l'applicazione;
- un insieme di *attività* che descrivano le informazioni rilevanti riguardo le attività che l'utente sta svolgendo.

Tutti gli elementi contestuali sono descritti da classi, rappresentate in formato UML. E' posta particolare attenzione sul modello per la descrizione del canale, poichè rappresenta la principale caratteristica di MAIS. Esso viene considerato come composto da:

- *il dispositivo* che l'utente utilizza per accedere all'applicazione, considerando tutte le componenti software e hardware;
- *l'interfaccia di rete* attraverso cui il dispositivo è connesso alla rete, considerando anche in questo caso tutti i parametri riguardanti l'hardware e il software;
- *la rete* utilizzata per trasferire le informazioni e caratterizzata da parametri di rete (copertura,velocità) e di qualità;
- *i protocolli applicativi* usati dal servizio.

MAIS framework

Ha lo scopo di fornire un modello comune per la comprensione dei sistemi informativi in ambito mobile. Il framework è di carattere generale e può essere utilizzato come una base per descrivere una struttura concettuale comune per la descrizione di tutte le tipologie di sistemi informativi mobili, indipendentemente dalla specifica architettura MAIS. E' composto dai seguenti modelli:

- *il modello funzionale* per la descrizione dei servizi e di tutte le entità correlate, considerando la prospettiva sia del richiedente che del fornitore dei servizi;
- *il modello architetturale* per descrivere le componenti del dispositivo usato per accedere al servizio, il comportamento di ogni componente e come esse interagiscono tra loro;
- *il modello contestuale* per la descrizione del contesto in cui i servizi sono richiesti ed utilizzati e descritto nella sezione precedente.

2.4 Granularità

Il framework realizzato nella tesi considera un modello del contesto che rappresenti i dati contestuali correlati con la loro granularità, allo scopo di utilizzarla nello sviluppo dell'applicazione per la realizzazione dell'adattività. La granularità in relazione alle applicazioni legate al contesto non è mai stata considerata in letteratura, sono però stati ricercati degli articoli sull'argomento relativi ad altri ambiti, allo scopo di analizzarne le caratteristiche, le problematiche ed i modelli formali definiti.

Con *granularità dell'informazione*, si indica il livello di dettaglio delle informazioni stesse. E'un problema significativo in particolare per quanto riguarda i dati monitorati [23]. Per il monitoraggio di parametri contestuali vengono utilizzate *reti di sensori*, cioè insiemi di dispositivi distribuiti spazialmente che misurino lo stato di un'entità fisica o di una condizione ambientale. In una rete di sensori, le principali problematiche sono la scelta del posizionamento dei sensori, quanto frequentemente interrogarli e quali dati raccogliere. Ovviamente una rete più densa ci fornirà dati migliori, ma a fronte di un costo significativo in termini di efficienza energetica. Il concetto di *eco-efficienza* consiste nel lavorare sui giusti prodotti, servizi e sistemi. Per questo motivo bisogna individuare quali sono le informazioni da estrarre e a quale livello di dettaglio. Non esite una granularità intrinsecamente mi-

gliore rispetto ad un'altra, ciò dipende sempre dal contesto in cui è rilevata ed utilizzata.

Si deve quindi estrarre conoscenza significativa da ogni dataset al giusto livello di dettaglio, ed avere delle opportune funzioni che permettano di lavorare con dati con diversa granularità (integrazione dei dati [2]). Non tutti i dati sorgenti sono necessari, anzichè catturare tutti i dati e scartare quelli non necessari è più utile catturare solo quelli significativi ed al giusto livello di dettaglio per la specifica applicazione.

2.4.1 Modello per la rappresentazione della granularità con dati temporali

Un modello per la rappresentazione della granularità è stato fornito nell'articolo [9] in relazione però unicamente al parametro tempo. Il modello è fornito allo scopo di estendere il linguaggio SQL per la gestione di dati temporali TSQL. All'interno di un database sono memorizzati dati a differenti granularità, definite dall'utente. C'è bisogno quindi di fornire delle funzioni di trasformazione tra diversi livelli di granularità.

Il dominio temporale è un insieme di punti temporali utilizzati per definire ed interpretare concetti legati al tempo. E'un insieme T totalmente ordinato di punti temporali con relazione d'ordine $\leq$. Le porzioni del dominio temporale sono raggruppate in aggregazioni chiamate *granuli*. La *granularità* è una funzione di mapping G che associa numeri interi a granuli tale che:

- se $i < j$ e $G(i)$ e $G(j)$ sono non vuote, allora ogni elemento di $G(i) \leq G(j)$;
- se $i < k < j$ e $G(i)$ e $G(j)$ sono non vuote, allora $G(k)$ è non vuota e
- $G(0)$, l'origine di G, è non vuota.

Il primo requisito implica che i granuli all'intero di una stessa granularità siano non sovrapposti e totalmente ordinati, con l'ordinamento ereditato dagli interi. Il secondo assicura che il set di interi mappato nei granuli sia continuo;

il terzo rappresenta una convenzione. I granuli descritti con una granularità possono essere aggregati tra loro per formare granuli più ampi provenienti da una granularità più grossa. Le conversioni tra granularità possono essere:

- *regolari:* quando un granulo più grosso è formato sempre dallo stesso numero di granuli più fini (es. giorni → settimana, una settimana è sempre composta da 7 giorni);
- *irregolari:* quando un granulo più grosso è formato da un numero variabile di granuli più fini (es. giorni → mese, un mese può avere 28, 29, 30, 31 giorni).

Le conversioni tra due granularità si effettuano utilizzando la funzione *scale(g,H)* dove g è un dato misurato alla granularità G e sarà trasformato in un dato h con granularità H. Per descrivere le relazioni tra due granularità H e G, vengono introdotti i concetti di *coarser than*, *finer than* e *incompatibilità*. Date due granularità H e G:

- H si dice "coarser than" G e G è detta "finer than" H, se per ogni granulo $h \in H$ esiste un set di granuli $S \subseteq G$ tale che $h = \cup_{g \in S} g$. Se G è finer o coarser than H, allora le due granularità si dicono comparabili.
- altrimenti H è G sono incomparabili.

Nello svolgimento di operazioni tra dati di diversa granularità, gli operandi devono prima essere convertiti alla stessa granularità e poi le operazioni possono essere eseguite. Si assume che $g \in G$ e $h \in H$ siano due elementi alla granularità indicata e $\odot$ un'operazione o un predicato binario tra elementi. F sia una granularità più fine di entrambe le granularità G ed H, e C una granularità più grossa sia di G che di H. L'operazione $h \odot g$ può essere gestita nei seguenti modi:

Incompatibilità: viene segnalato un errore di compatibilità tra le due granularità;

Semantica dell'operatore di sinistra: svolge l'operazione alla granularità del primo operando:

$$g \odot h = g \odot scale(h, G)$$

Semantica dell'operatore di destra: svolge l'operazione alla granularità del secondo operando:

$$g \odot h = scale(g, H) \odot h$$

Semantica più fine: svolge l'operazione alla granularità più fine tra le due. Se le due granularità non sono comparabili la effettua ad una granularità più fine di entrambe:

$$g \odot h =$$

$$\begin{cases} g \odot scale(h, G) & se\ G\ è\ più\ fine\ di\ H \\ scale(g, H) \odot h & se\ H\ è\ più\ fine\ di\ G \\ scale(g, F) \odot scale(h, F) & se\ H\ e\ G\ non\ sono\ comparabili \end{cases}$$

Semantica più grossa: svolge l'operazione alla granularità più grossa tra le due. Se le due granularità non sono comparabili la effettua ad una granularità più grossa di entrambe:

$$g \odot h =$$

$$\begin{cases} g \odot scale(h, G) & se\ G\ è\ più\ grossa\ di\ H \\ scale(g, H) \odot h & se\ H\ è\ più\ grossa\ di\ G \\ scale(g, C) \odot scale(h, C) & se\ H\ e\ G\ non\ sono\ comparabili \end{cases}$$

Grafo delle granularità

La struttura più adatta per la descrizione delle relazioni tra le diverse granularità è l'utilizzo di un DAG (Directed Acyclic Graph) per ogni parametro da rappresentare, in cui i nodi rappresentano le granularità e gli archi le relazioni tra di esse. Le granularità nel grafo sono legate dalla relazione di "finer

than" e "coarser than", ma può anche essere specificata una funzione di conversione tra le granularità legate da ogni arco. Nel grafo deve sempre essere presente una granularità che sia più fine di ogni granularità rappresentata. Questa granularità prende il nome di *chronon*, indicata con $\perp$, e rappresenta la precisione del dato.

Nell'articolo [5] il modello descritto è esteso oltre che ai dati temporali a quelli spaziali. Viene inoltre fornito un modello UML della granularità dei dati, che descriva i concetti di granuli e granularità e le relazioni di coarser than, finer than in riferimento a dati spazio-temporali.

Capitolo 3

Framework granulare e proattivo per l'adattività

Le applicazioni adattive hanno lo scopo di modificare automaticamente e dinamicamente il proprio comportamento sulla base di cambiamenti rilevati da opportuni parametri contestuali. Il contesto è definito come l'insieme delle informazioni che posssono essere utilizzate per rappresentare lo stato di un'entità. Un'entità è una persona, un luogo, o un oggetto che è considerato rilevante nell'interazione tra un utente e un'applicazione, inclusi l'utente e l'applicazione stessi. Il contesto è solitamente descritto da un insieme di parametri, relativi al dominio di esecuzione dell'applicazione e che si riferiscono alle principali entià che entrano in gioco nello svolgimento dell'applicazione stessa. Per utilizzare dati contestuali all'interno di un'applicazione è necessario fornire una descrizione formale del contesto e delle entità che lo compongono. In generale, come mostrato in figura 3.1, le principali entità contestuali da modellizzare sono le seguenti:

- utente: colui che accede all'applicazione. Può avere diversi ruoli e quindi diversi permessi per eseguire operazioni all'interno del sistema (es. amministratore di sistema, utente semplice ecc.);
- canale: è l'insieme degli strumenti fisici che permettono all'utente di accedere all'applicazione ed è quindi caratterizzato dal dispositivo e dal

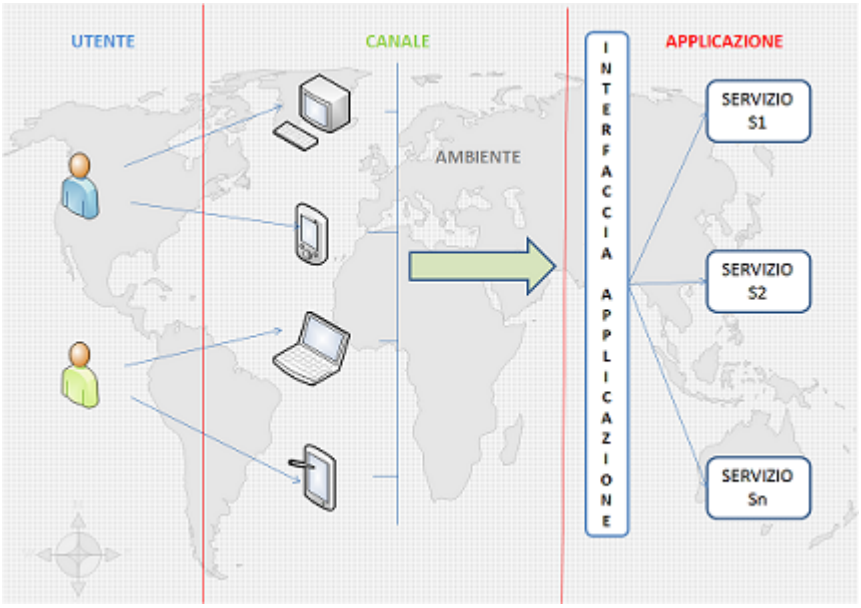

Figura 3.1: Contesto di esecuzione

canale per mezzo dei quali avviene la connessione con l'applicazione;

- applicazione: è caratterizzata da un'interfaccia astratta attraverso la quale l'utente può interagire con il sistema ed è connessa a servizi concreti, con specifiche proprietà ed operazioni, offerti da diversi service provider;

- ambiente: rappresenta tutto ciò che non appartiene al sistema vero e proprio, ma ne condiziona il comportamento.

Vengono forniti in letteratura diversi modelli per la rappresentazione formale del contesto, modellizzando ogni entità rilevante sottoforma di classe UML e descrivendone le relazioni.

La principale limitazione dei modelli per la rappresentazione del contesto esistenti è la mancanza di una correlazione tra i parametri ed il loro il livello di dettaglio. A seconda di specifiche condizioni contestuali potrebbe essere necessario svolgere operazioni a diverso livello di dettaglio, oppure il livello di dettaglio con cui è descritto il dato potrebbe determinare modifiche

nello svolgimento del processo. Inoltre, come descritto nel dettaglio nella sezione successiva, lavorare con dati ad un livello di dettaglio non idoneo per l'applicazione in esecuzione comporta inefficienze temporali, energetiche ed operazionali.

Lo scopo di questo lavoro è quello di realizzare un framework adattivo che descriva e utilizzi i dati del contesto in relazione al loro livello di dettaglio.

3.1 Granularità dell'informazione

Allo scopo di gestire all'interno dell'applicazione una descrizione del contesto legata al livello di dettaglio dei dati viene introdotto e modellizzato il concetto di *granularità dell'informazione*, ossia il livello di dettaglio delle informazioni stesse.

I dati utilizzati in un'applicazione per la descrizione del contesto di esecuzione possono essere:

- *monitorati:* misurati da sensori in grado di percepire specifiche condizioni ambientali. Sono caratterizzati da una precisione ed una frequenza di interrogazione dei sensori;
- *espliciti:* richiesti esplicitamente all'utente durante l'esecuzione dell'applicazione (ad esempio attraverso l'utilizzo di specifiche form);
- *derivati:* ottenuti mediante elaborazione di dati (es. calcolati tramite operazioni matematiche su altri dati semplici o complessi, stimati sulla base di dati storici ecc). Possono essere costituiti da dati esatti o stimati.

Nelle applicazioni adattive legate al contesto l'utilizzo di dati ad un diverso livello di dettaglio è una caratteristica fondamentale in quanto i dati contestuali sono spesso monitorati mediante reti di sensori. Le principali scelte da valutare in questo ambito sono: il numero di sensori da posizionare, la loro localizzazione, e di quale tipologia. Una rete molto densa e composta

da sensori con un'alta precisione rappresenta la garanzia che i dati raccolti siano molto accurati, a fronte però di un grave sforzo in termini di costo e di efficienza energetica. E' preferibile quindi estrarre i giusti dati ed al giusto livello di dettaglio, a seconda delle condizioni contestuali in cui si opera, per evitare sprechi in termini di energia.

Inoltre, per trattare ed elaborare dati ad una granularità molto fine potrebbe essere necessario uno sforzo più consistente in termini operazionali, di calcolo e di implementazione, causa di inefficienze temporali relative a tutte le tipologie di dati, e problematiche in un ambito come le applicazioni adattive che richiedono altro tempo aggiuntivo per l'adattamento.

D'altro canto un dato espresso ad una granularità troppo grossa potrebbe non essere sufficiente per la rilevazione di caratteristiche contestuali significative, che potrebbero essere necessarie per la determinazione di un'esigenza di adattamento.

Se i sensori non sono sufficientemente precisi, e il dato richiesto è ad una granularità molto fine, potrebbero verificarsi degli errori nella rilevazione dei dati contestuali e provocare l'innesco di meccanismi adattivi non necessari. L'adattamento è un meccanismo costoso, quindi adattarsi ad un contesto errato comporta effetti peggiori sullo svolgimento dell'applicazione rispetto al non adattarsi affatto.

Bisogna quindi trovare un buon compromesso tra livello di dettaglio del dato da rilevare ed utilizzare e efficienza temporale ed energetica, in relazione allo specifico contesto di esecuzione.

Viene quindi scelto di introdurre nel modello contestuale una descrizione formale delle granularità del dato, associandola ad ogni parametro per cui sia significativo considerarla, allo scopo di evitare errori, adattamenti errati o inutili, sprechi di tempo ed energia, ma allo stesso tempo di fornire al processo il dato al livello di dettaglio necessario per il corretto svolgimento delle operazioni.

Non esiste una granularità intrinsecamente migliore di un'altra, ma la scelta del livello di dettaglio con cui estrarre le informazioni è anch'essa stret-

tamente dipendente dal contesto di esecuzione in cui si lavora e determinata a runtime.

3.1.1 Modello per la rappresentazione della granularità

Nel framework realizzato, si presenta quindi la necessità di aggiungere al tipico modello del contesto basato su UML, ad esempio quello proposto in MAIS [20], una classe che descriva la granularità associata ad ogni parametro del dominio per cui sia significativo e funzionale prevedere l'utilizzo di diverse granularità all'interno dello stesso processo. Il modello UML della granularità è descritto in figura 3.2.

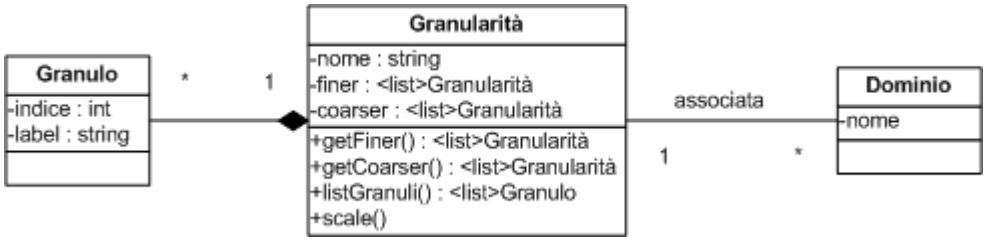

Figura 3.2: Modello UML per la descrizione della granularità

Il dominio di ogni parametro è suddiviso in porzioni, dette granuli, in riferimento ad una granularità. Ogni granulo rappresenta l'unione di tutti i punti del dominio con uno specifico valore del parametro ad un dato livello di granularità. I granuli possono essere uniti tra loro per formare granuli più ampi appartenenti ad una granularità più grossa, come mostrato in figura 3.3. Questa figura mostra il concetto di granuli e granularità in riferimento alla locazione spaziale. La granularità più fine considerata è la Provincia, composta dai granuli rappresentanti la porzione spaziale relativa ad ogni provincia (LC, MI, SO, BG ecc). Unendo un insieme di granuli alla granularità Provincia si ottiene la granularità più grossa Regione, con granuli Lombardia, Piemonte ecc. Il procedimento può essere ripetuto per formare granularità più grosse.

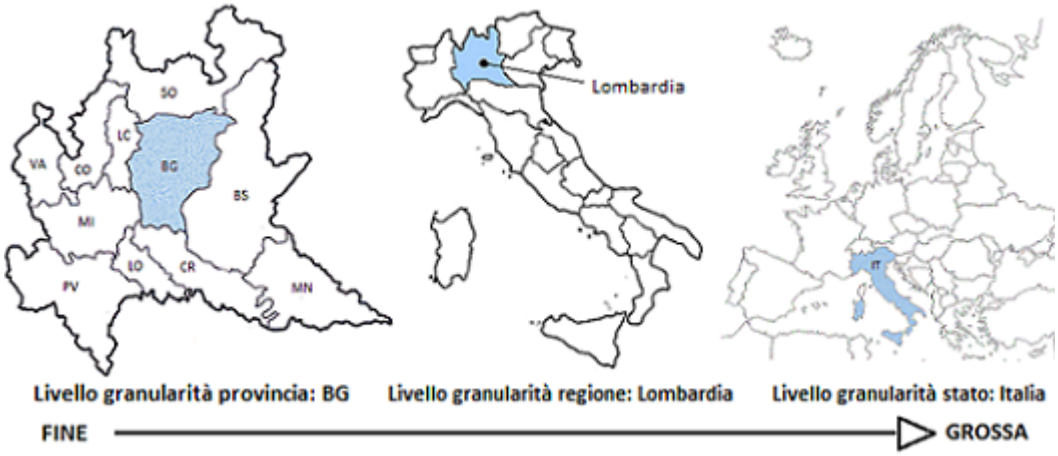

Figura 3.3: Aggregazione di granuli riferiti al parametro locazione spaziale

Relazioni tra granularità

Date due granularità G ed H esse possono essere tra loro nelle seguenti relazioni:

- se per ogni granulo di G esso è costituito dall'unione di granuli di H, allora G si dice più grossa di H, e H più fine di G;
- altrimenti le due granularità sono tra loro incomparabili.

Le operazioni *getFiner()* e *getCoarser()*, presenti nella classe Granularità, definita in figura 3.2, sono introdotte allo scopo di restituire tutte le granularità rispettivamente più fini e più grosse della granularità considerata.

Operazioni tra granularità

Deve essere fornito un metodo che consenta la conversione tra una granularità ed un'altra, poichè le operazioni in generale devono essere svolte tra dati alla stessa granularità. Questo metodo nella rappresentazione UML descritta è fornito da *scale()* che permette la trasformazione tra una granularità ed un'altra comparabile ad essa.

3.1.2 Utilizzo di servizi a granularità diverse nel framework

Nel framework sono considerati due scenari di riferimento che leghino la granularità informativa all'adattività delle applicazioni in relazione al loro contesto di esecuzione:

- svolgimento di sottoprocessi diversi a seconda della granularità con cui è monitorato il dato (Figura 3.4): in questo caso a seconda della granularità del dato il processo svolto sarà differente, in quanto ogni sottoprocesso necessiti di dati ad una diversa precisione, ed i dati ad una precisione molto alta potrebbero non sempre essere disponibili o accurati;

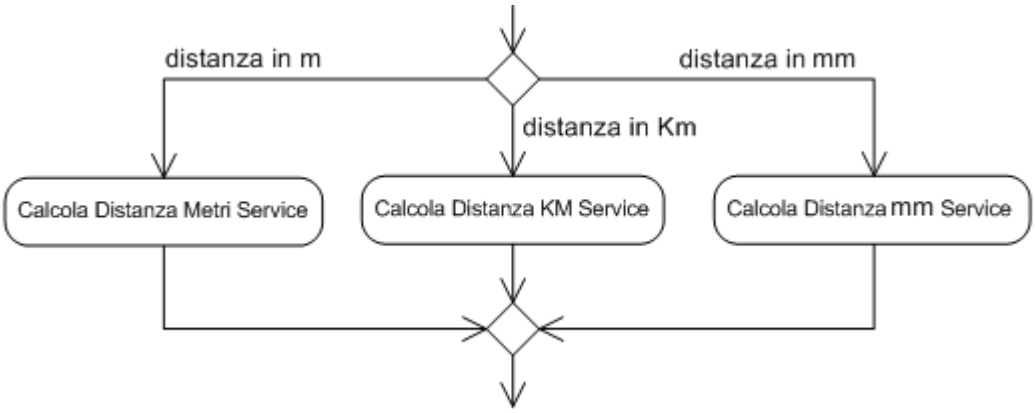

Figura 3.4: Processi diversi a seconda della granularità del dato

- svolgimento di sottoprocessi a diversi livelli di dettaglio a seconda di opportuni parametri contestuali (Figura 3.5): a seconda del contesto di riferimento è necessario svolgere alcune operazioni in maniera più o meno dettagliata. Nell'esempio in figura è descritto un processo di richiesta di informazioni riguardanti un percorso stradale: a seconda della conoscenza del luogo da parte dell'utente saranno fornite informazioni più o meno dettagliate sul tragitto da svolgere.

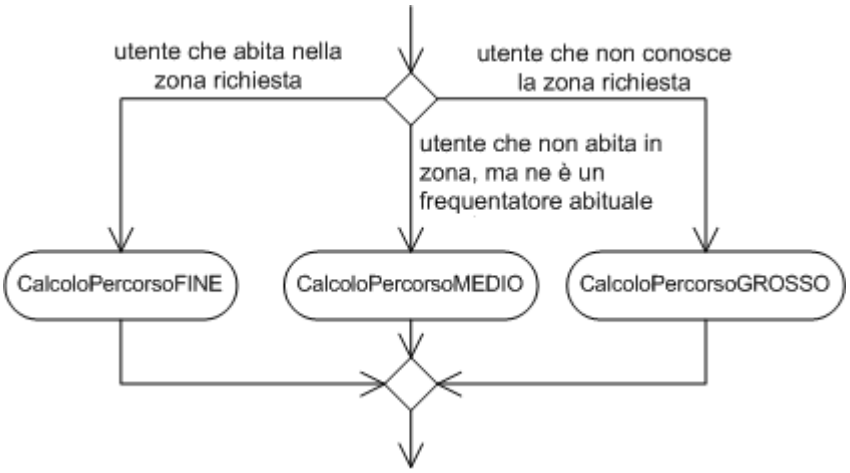

Figura 3.5: Processi a diversa granularità a seconda del contesto

3.2 Granularità di monitoraggio

Viene considerata nel framework un'altra tipologia di granularità, correlata con il periodo di monitoraggio (o di ricalcolo) dei parametri contestuali. Questo tipo di granularità è introdotta poichè nelle applicazioni adattive context-aware, i dati utilizzati per la descrizione del contesto devono essere sempre aggiornati, altrimenti l'adattamento avverrebbe in modo errato.

A questo proposito viene introdotto il concetto di *granularità di monitoraggio*, una grandezza che descriva quanto frequentemente debbano essere misurati i dati rilevanti per l'applicazione. Il problema in questo caso è quello di determinare ogni quanto sia necessario raccogliere i dati per catturare le caratteristiche significative dell'ambiente. E' possibile che in alcune situazioni, ad esempio di emergenza quando i cambiamenti del sistema si verificano più frequentemente, sia necessario interrogare i sensori (o ricalcolare i dati) più spesso rispetto alla norma e in altre, al contrario, una misurazione più frequente dei dati contestuali rappresenterebbe unicamente un costo senza fornire alcuna informazione aggiuntiva sullo stato dell'ambiente.

Si cerca quindi di fornire un meccanismo per distinguere questi scenari, in modo da evitare da un lato la raccolta di dati superflui e dall'altro la rac-

colta di un numero di dati non sufficiente a rilevare le modifiche nel contesto applicativo.

Solitalmente il monitoraggio e la raccolta dei dati contestuali vengono effettuati in maniera continua, o al limite ad intervalli discreti brevi e regolari. Come descritto nella sezione precedente, l'estrazione di dati da reti di sensori o da database specifici rappresenta un costo significativo in termini di efficienza ed un grande sforzo operativo. Per questo motivo si presenta la necessità di monitorare i dati solo quando necessario per la rilevazione di cambiamenti nel contesto.

Nel framework viene proposto di risolvere questa problematica salvando nel modello contestuale un attributo che descriva un intervallo discreto di monitoraggio, fissato per ogni parametro contestuale per il quale sia di interesse, aggiornandolo dinamicamente sulla base del verificarsi di situazioni anomale.

Il valore dell'intervallo viene stabilito sulla base di considerazioni di carattere generale, di dati storici e in relazione allo scenario di riferimento, in modo da garantire la presenza di dati aggiornati in condizioni regolari. Il framework si occuperà di rilevare a runtime eventuali anomalie e provvedere ad aggiornare in accordo il campo relativo alla granularità di monitoraggio (aumentandola o riducendola).

3.3 Composizione di servizi

Una volta modellizzato il contesto di esecuzione si procede allo sviluppo vero e proprio dell'applicazione. I dati contestuali associati alla loro granularità saranno rilevati a runtime ed utilizzati nello svolgimento delle operazioni.

Nel framework si presenta la necessità di coordinare ed orchestrare i web service allo scopo di raggiungere un obiettivo comune. Per fare ciò vengono utilizzati *strumenti di gestione del workflow*, come BPEL (Business Process Execution Language), che permettano di descrivere la sequenza di attività da svolgere mediante una sorta di diagramma di flusso, che determini l'ordine

di esecuzione delle operazioni e le condizioni sotto le quali debbano essere svolte. Nella maggior parte delle applicazioni orientate ai servizi i singoli web service componenti un processo di business non sono realizzati ad hoc per la specifica applicazione, ma sono selezionati dinamicamente. BPEL permette la descrizione delle attività da svolgere in maniera astratta, connettendole concretamente ai servizi specifici a cui si riferiscono a runtime.

3.3.1 Invocazione dinamica dei servizi

Poichè lo scopo del framework è quello di far fronte a cambiamenti contestuali, che si verificano a runtime è opportuno prevedere un meccanismo che si occupi della scelta del servizio da invocare durante l'esecuzione della specifica istanza del programma, anzichè associare un servizio concreto ad ogni attività in fase di progettazione.

Viene quindi configurato un componente che si occupi di invocare *dinamicamente* i servizi a seconda dei parametri rilevati a runtime. Il servizio specifico da invocare è scelto durante l'esecuzione dell'applicazione, poichè le diverse condizioni contestuali non possono essere completamente previste a design-time. Per fare ciò nel framework vengono realizzati servizi di due tipologie:

- *servizi astratti:* rappresentano un'interfaccia astratta del servizio, ma non svolgono alcuna operazione concreta;
- *servizi concreti:* sono associati ad un preciso servizio astratto, con il quale condividono l'XML schema. Si occupano di svolgere concretamente le operazioni fornite dall'interfaccia.

Ad ogni servizio astratto è associato un insieme di servizi concreti, omogenei tra loro in termini di caratteristiche, struttura e funzionalità, con la stessa interfaccia astratta, ma che svolgano le operazioni differentemente. Ogni servizio concreto è associato da un'unica interfaccia astratta, rappresentativa del gruppo di servizi a cui appartiene.

L'idea che sta alla base dell'invocazione dinamica dei servizi è quella di scegliere a design-time un'interfaccia astratta che descriva le operazioni che si vogliono svolgere in ogni fase del processo, configurare un componente che si occupi di rilevare i parametri contestuali di interesse a runtime e connettere l'interfaccia astratta al servizio concreto associato più idoneo per le specifiche esigenze dell'istanza dell'applicazione in esecuzione.

Nel framework l'invocazione dinamica dei servizi sarà implementata mediante la configurazione di un proxy che permetta, a fronte di una richiesta astratta dell'utente, di invocare il servizio concreto più idoneo, effettuando un mapping tra l'XML della richiesta astratta e quello della richiesta concreta e tra la risposta concreta e quella astratta. L'interazione tra utente, applicazione, proxy e servizio concreto è descritta in Figura 3.6.

Il proxy si dovrà quindi occupare di effettuare le trasformazioni in ingresso e in uscita tra XML astratto e concreto e di scegliere sulla base di parametri contestuali monitorati a runtime il servizio concreto più idoneo da invocare.

In questo modo l'adattività sarà realizzata dinamicamente, in modo automatico e trasparente all'utente che accede all'applicazione.

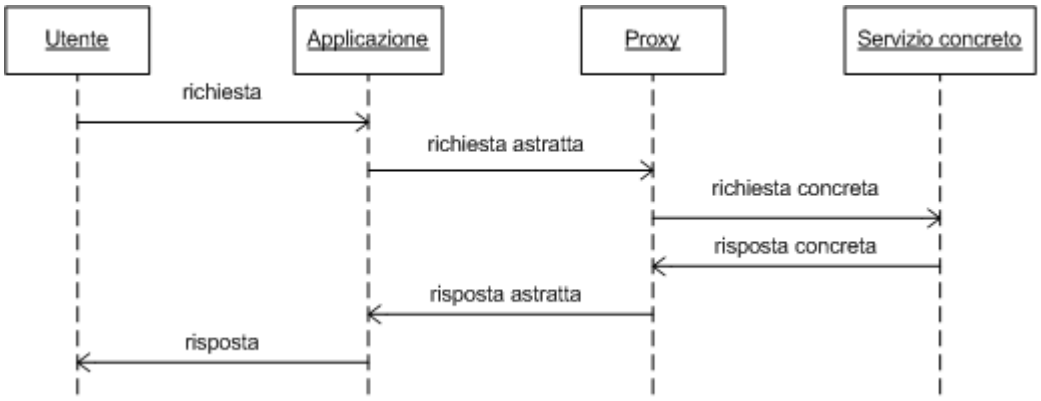

Figura 3.6: Interazione tra utente, applicazione, proxy e servizio concreto.

3.4 Invocazione dinamica di servizi in relazione alla granularità

L'approccio dinamico all'invocazione dei servizi può essere generalizzato allo svolgimento di ogni attività, ma assume particolare interesse nelle porzioni del processo in cui si prevedono e monitorano cambiamenti nel contesto di esecuzione che potrebbero rilevare esigenze adattive.

All'interno del framework si presenta la necessità di rilevare a runtime i valori dei parametri contestuali e sulla base di essi svolgere determinati servizi concreti. In particolare vengono mostrati due esempi di invocazione dinamica dei servizi in riferimento agli scenari legati alla granularità del dato descritti nella sezione 3.1.2, fornendo un meccanismo per la rilevazione a runtime dei dati contestuali e per l'invocazione dinamica del servizio concreto corrispondente.

- invocazione dinamica di sottoprocessi diversi a seconda della granularità rilevata del parametro (Figura 3.7): a runtime viene rilevata la granularità con cui è descritto il dato ed invocato il servizio concreto corrispondente.

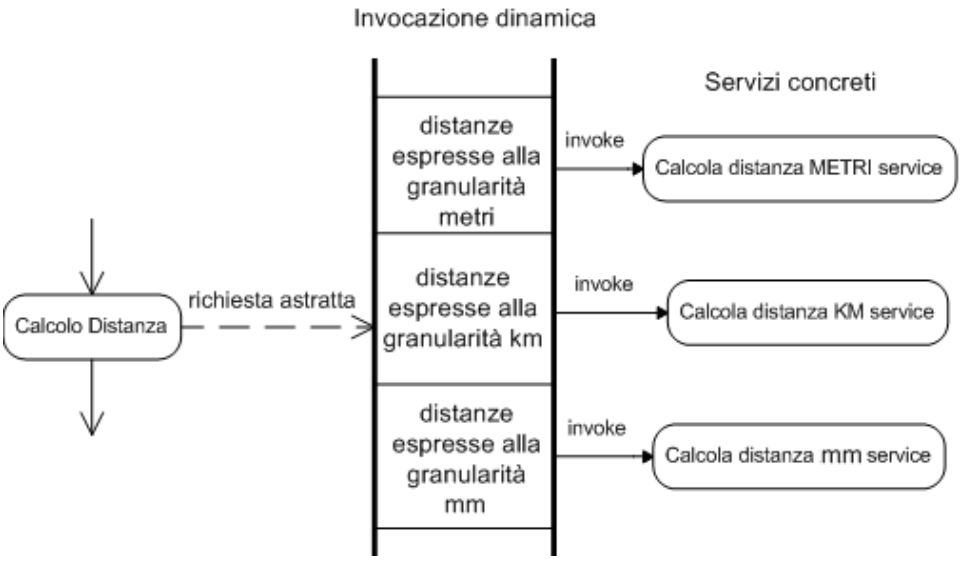

Figura 3.7: Invocazione dinamica di processi diversi a seconda della granularità

- invocazione dinamica di sottoprocessi a diverso livello di dettaglio a seconda di parametri contestuali (Figura 3.8): a runtime vengono rilevati opportuni parametri contestuali, che indichino conoscenza del luogo da parte dell'utente. A seconda di questi dati il sottoprocesso concreto da svolgere sarà effettuato a diverso livello di dettaglio.

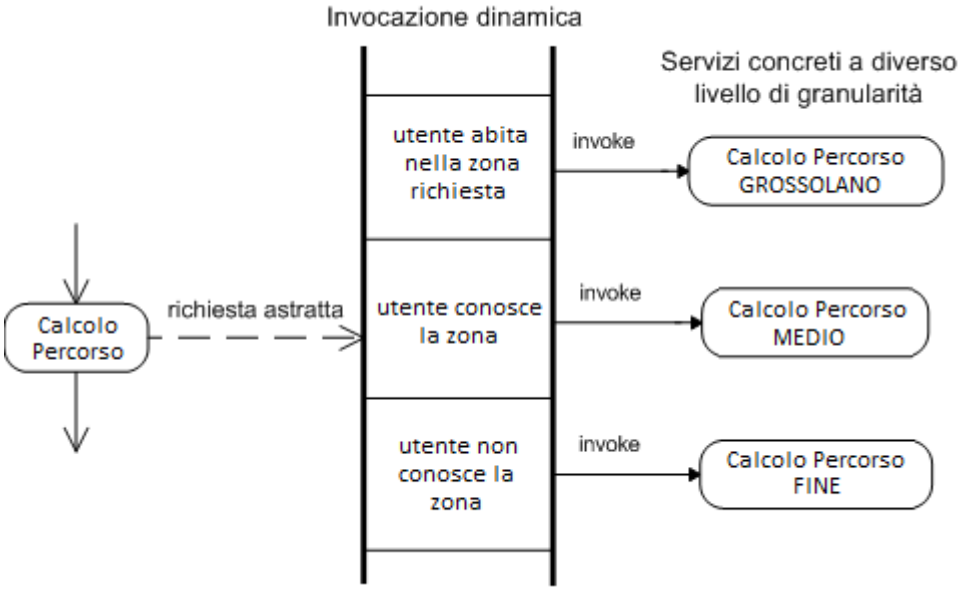

Figura 3.8: Invocazione dinamica di servizi a diversa granularità secondo il contesto

3.5 Adattività

Il framework, riferendosi ad applicazioni orientate ai servizi, deve operare in un ambiente distribuito e soggetto a cambiamenti improvvisi. Per questo motivo è necessario prevedere un comportamento adattivo dell'applicazione. Con adattività si intende la capacità delle applicazioni di modificare il proprio comportamento dinamicamente a seguito della rilevazione di cambiamenti a runtime nello svolgimento dell'applicazione stessa o nel contesto di esecuzione.
L'adattività può essere realizzata secondo due approcci principali:

- *reattivamente:* ossia reagendo ai cambiamenti che si rilevano a runtime;

- *proattivamente:* prevedendo in anticipo i cambiamenti che potrebbero verificarsi a runtime ed effettuando delle modifiche nell'applicazione prima che i cambiamenti veri e propri si rilevino.

La maggior parte dei lavori presenti in letteratura riguardanti l'adattività si concentrano unicamente sull'adattività di tipo reattivo. Questo significa che il meccanismo adattivo entra in gioco dopo che si verifica una situazione critica o anomala. E' quindi basata sul monitoraggio, allo scopo di verificare se vengono effettuate violazioni nei requisiti.

Poichè l'adattività è un'attività lunga, per evitare troppi ritardi nello svolgimento del processo, o interruzioni a runtime, è preferibile che il rilevamento di una situazione critica e la messa in atto di meccanismi correttivi siano effettuati prima che l'anomalia si verifichi, proattivamente, per evitare le conseguenze del verificarsi di un potenziale, futuro cambiamento/problema. L'adattività proattiva deve essere affiancata, e non totalmente sostituita, a quella reattiva, che deve essere sempre presente in caso di cambiamenti improvvisi e non prevedibili.

Nel framework vengono considetati sia un approccio proattivo che uno reattivo per la realizzazione dell'adattività.

3.5.1 Adattività reattiva

Per quanto riguarda l'adattività reattiva, si considerano gli scenari in cui vengono rilevati a runtime opportuni parametri che descrivano il contesto di esecuzione e in accordo ad essi il proxy sceglie di invocare dinamicamente il servizio concreto più idoneo. Reagendo quindi al contesto in cui ci si trova, si sceglie il servizio concreto più idoneo da invocare. Nel framework ciò viene realizzato nella gestione di dati a diverse granularità. A seconda del contesto che viene rilevato si sceglie la granularità del servizio concreto da invocare oppure a seconda della granularità del parametro rilevata si sceglie il servizio più idoneo.

3.5.2 Adattività proattiva

Viene effettuata anche un'analisi proattiva del contesto, allo scopo di mettere in atto meccanismi correttivi prima che si verifichi l'anomalia vera e propria. Le previsioni sul comportamento dell'applicazione sono effettuate analizzando serie di dati storici e i servizi concreti da invocare saranno scelti in accordo. Ovviamente l'adattività proattiva non è basata su dati certi, ma su stime. Nel framework per modellizzare il comportamento dell'applicazione in maniera verosimile è utilizzato un tool di machine learning.

Con *machine learning* si intende un ramo dell'intelligenza artificiale legata alla progettazione e allo sviluppo di algoritmi che permettano alle macchine di evolvere il loro comportamento (migliorando le proprie performance) sulla base di serie di dati storici.

Nel framework viene utilizzato un algoritmo di classificazione per effettuare l'analisi proattiva del contesto allo scopo di rilevare e gestire eventuali anomalie o cambiamenti.

Negli *algoritmi di classificazione* i dati in input consistono in record aventi attributi o caratteristiche multiple. Ogni record appartenente al dataset è etichettato con una classe. L'obiettivo della classificazione è quello di analizzare i dati in input e sviluppare un modello per ogni classe, utilizzando le caratteristiche presenti nei dati. Gli algoritmi di classificazione portano all'identificazione di schemi che definiscono la classe a cui appartiene un dato record. In genere, partendo dall'utilizzo di insiemi di dati esistenti e già classificati, si cercano di definire alcune regolarità che caratterizzano le varie classi. Le descrizioni delle classi vengono usate per classificare nuovi record, di cui non si conosce la classe di appartenenza.

Sulla base di dati storici e utilizzando tecniche di classificazione, viene fornito un modello del sistema ed effettuata una previsione di eventuali anomalie sulla base dei parametri contestuali attuali. A seconda di dati raccolti in esecuzioni precedenti dell'applicazione, viene creato un modello dell'applicazione ed è stimato l'effetto dell'esecuzione del programma con i dati monitorati a runtime. L'analisi è effettuata in punti prestabiliti e signifi-

cativi per l'applicazione, detti checkpoint, utilizzando i dati disponibili al checkpoint e delle stime dei dati non ancora disponibili.

Il processo di analisi proattiva del contesto e di scelta del sottoprocesso da eseguire avviene attraverso le seguenti fasi:

1. Estrazione dati storici dall'apposito DB;
2. Scelta dell'algoritmo di classificazione;
3. Monitoraggio o stima dei parametri contestuali attuali;
4. Previsione di eventuali anomalie/cambiamenti;
5. Scelta del sottoprocesso da invocare.

Estrazione dati storici dal DB

A questo scopo viene configurato un *dataset* che descriva in forma strutturata i dati storici relativi ad esecuzioni precedenti dell'applicazione. In particolare sono determinati gli attributi rilevanti per lo specifico parametro su cui si vogliono effettuare previsioni e vengono salvati in correlazione ad un'etichetta che descriva le caratteristiche del parametro da stimare. Una prima sezione del dataset descrive la struttura del file: quali sono gli attributi da considerare, la classe da stimare e i valori che possono assumere. Nella sezione successiva vengono salvate le specifiche istanze che descrivano i valori dei parametri rilevati in invocazioni precedenti dell'applicazione in relazione alla classe a cui apparteneva il parametro da stimare. Una parte del dataset è utilizzata come *training set* allo scopo di modellizzare il comportamento dell'applicazione, e la restante parte come *test set* per la valutazione delle performance della classificazione.

Scelta dell'algoritmo di classificazione

Nel framework vengono applicati in fase di design alcuni dei più noti algoritmi di classificazione al dataset. L'output dell'esecuzione dell'algoritmo di

classificazione ad uno specifico dataset fornisce alcune misure sulla qualità della classificazione. Queste verranno confrontate ed analizzate allo scopo di individuare il migliore algoritmo per il dataset di riferimento.

Gli algoritmi di classificazione considerati all'interno del framework sono descritti in seguito:

Alberi decisionali: algoritmo J48

Gli alberi di decisione [13] costituiscono il modo più semplice di classificare degli oggetti in un numero finito di classi. Essi vengono costruiti suddividendo ripetutamente i records in sottoinsiemi omogenei rispetto alla variabile risposta. La suddivisione produce una gerarchia ad albero, dove i sottoinsiemi (di records) vengono chiamati nodi e, quelli finali, foglie. In particolare, i nodi sono etichettati con il nome degli attributi, gli archi (i rami dell'albero) sono etichettati con i possibili valori dell'attributo soprastante, mentre le foglie dell'albero sono etichettate con le differenti modalità dell'attributo classe che descrivono le classi di appartenenza. Un oggetto è classificato seguendo un percorso lungo l'albero che porti dalla radice ad una foglia. I percorsi sono rappresentati dai rami dell'albero che forniscono una serie di regole.

La struttura di un albero di decisione può diventare molto complicata, soprattutto nei casi derivati da database contenenti centinaia di attributi ed una variabile risposta con differenti classi. In situazioni del genere, lasciar "crescere" l'albero senza stabilire un limite di qualsiasi natura può far si che l'albero ottenuto diventi non interpretabile e crei un numero elevato di regole, sovraadattando i dati. Esistono, quindi, dei criteri di controllo che limitano la crescita degli alberi, basati o sul massimo numero di regole ottenibili dalla classificazione o sulla massima profondità raggiungibile dall'albero o ancora sul numero minimo di records che devono essere presenti in ogni nodo per poter effettuare la divisione (splitting) in quel nodo. In tale ambito rientra anche la fase di pruning dell'albero che consiste nell'ottenere da un albero il più piccolo sottoalbero, che non comprometta l'accuratezza della classificazione/previsione resa possibile dall'albero madre.

Gli alberi decisionali offrono numerosi *vantaggi*:

- sono facili da capire;
- sono veloci;
- possono essere trasformati in regole;
- hanno dimostrato di ottenere buoni risultati, se messi a confronto con altre tecniche di gestione della conoscenza.

Per quanto riguarda i punti deboli nell'utilizzo degli alberi di decisione la critica sostiene che l'attributo divisore viene scelto quasi sempre da un algoritmo "greedy", che non tiene assolutamente conto dell'influenza che la scelta di un attributo divisore potrebbe avere sui futuri divisori. In altre parole, la decisione dell'attributo (o campo) divisore avviene ad ogni nodo dell'albero, in un preciso momento durante l'esecuzione dell'algoritmo, e non è mai più riconsiderata in seguito. Conseguenza di ciò è che tutti i campi divisori vengono scelti sequenzialmente e ogni campo divisore è dipendente dai predecedenti. Ciò implica che tutti i futuri campi divisori sono dipendenti dal nodo radice dell'albero, a tal punto che una modifica del campo divisore del nodo radice potrebbe portare alla costruzione di un albero completamente differente. Questa critica, di per sè, potrebbe portare ad una forte limitazione nell'uso di tali strumenti, tuttavia la bontà dei risultati spesso ottenuti utilizzando questa tecnica fanno parzialmente dimenticare la correttezza di tale critica.

Reti Bayesiane: approccio Naive Bayes

Una rete bayesiana è un grafo aciclico orientato (DAG) in cui:

- i nodi rappresentano le variabili;
- gli archi rappresentano le relazioni di dipendenza statistica tra le variabile e le distribuzioni locali di probabilità dei nodi foglia rispetto ai valori dei nodi padre.

Si basano sulla regola di Bayes ed esprimono relazioni di dipendenza condizionale (archi) tra le variabili in gioco (nodi). Ad ogni nodo è associata una tabella di probabilità condizionata che quantifica gli effetti che i genitori hanno sul nodo, dove per genitori si intendono tutti quei nodi che hanno archi che puntano al nodo.

Una rete bayesiana rappresenta la distribuzione della probabilità congiunta di un insieme di variabili. Supponiamo di voler classificare un insieme di elementi che sono descritti da un vettore di feature A_1,A_2,..,A_n ovvero dei valori numerici/letterali ricavati da determinate caratteristiche di ognuno di tali elementi, avendo a disposizione un training set con elementi già categorizzati. La probabilità che un certo elemento con determinati valori delle sue feature $A_1 = a_1$; $A_2 = a_2$;.. $A_n = a_n$ appartenga alla classe c è data da:

$$Pr(C = c|A_1 = a_1, A_2 = a_2, .., A_n = a_n)$$

Ora il teorema di Bayes (detto anche regola di Bayes) afferma che l'espressione qui sopra equivale a:

$$\frac{Pr(A_1 = a_1, A_2 = a_2, .., A_n = a_n|C = c)}{Pr(A_1 = a_1, A_2 = a_2, .., A_n = a_n)}.Pr(C = c)$$

$Pr(C = c)$ può essere facilmente calcolato: è la probabilità che un generico elemento di cui non si sa nulla appartenga a c, ricavata dalla percentuale di elementi del training set che appartiene a quella classe.

La probabilità al denominatore $Pr(A_1 = a_1, A_2 = a_2, .., A_n = a_n)$ è irrilevante per la classificazione perchè è la stessa per tutte le classi in cui si prova a inserire l'elemento. Quindi si tratta di calcolare $Pr(A_1 = a_1, A_2 = a_2, .., A_n = a_n|C = c)$.

Ipotesi Naive di Bayes: *Ogni feature è indipendente da ogni altra feature*

$$Pr(A_1 = a_1, .., A_n = a_n|C = c) = Pr(A_1 = a_1|C = c). \cdots .Pr(A_n = a_n|C = c)$$

Le probabilità moltiplicate tra loro alla destra dell'equazione, possono essere calcolate a partire dal training set, calcolando la percentuale di istanze

in cui una certa classe ha quel determinato valore di features rispetto al numero totale di istanze appartenenti a quella classe.

I *vantaggi* dell'utilizzo di una rete Naive Bayes sono:

- la classificazione è veloce;
- considera i valori assunti da molti attributi;
- è robusta rispetto alle alternative irrilevanti;
- i classificatori ottenuti sono facili da interpretare.

Gli *svantaggi* sono i seguenti:

- il dataset deve essere sufficientemente grande;
- gli attributi che descrivono le istanze devono essere condizionalmente indipendenti data la classificazione

NBTree

Rappresenta un approccio ibrido tra le reti Bayesiane di tipo Naive e gli alberi di decisione. E' un albero di decisione che utilizza dei classificatori Naive Bayes sui nodi.
L'algoritmo per la costruzione di un NBTree è il seguente:

Input: un dataset, composto da istanze classificate
Output: un albero di decisione con classificatori Naive Bayes sulle foglie

1. Per ogni attributo X_i se X_i è reale applica una soglia a X_i, poi valuta l'utilità $u(X_i)$ di uno split su X_i.
2. Sia j = $argmax_i(u_i)$ l'attributo con l'utilità maggiore.
3. Se u_j non è significativamente migliore dell'utilità del nodo corrente crea un classificatore Naive Bayes semplice per il nodo corrente e ritorna.

4. Altrimenti partiziona T a seconda dell'esito del test su X_j , se X_j è continuo applica una soglia a X_j . Effettua uno split su X_j per tutti i possibili valori.

5. Per ogni figlio richiama l'algoritmo ricorsivamente sulla porzione di T che corrisponde al test che porta a quel figlio.

I vantaggi dell'utilizzo di questo tipo di tecnica di apprendimento sono:

- le variabili considerate non devono essere necessariamente indipendenti;
- i database possono essere grandi;
- considera i valori assunti da molti attributi.

In breve, gli NBTree sembrano aumentare le performance degli alberi decisionali e dei classificatori bayesiani utilizzati singolarmente.

Ogni algoritmo di classificazione è applicato al dataset utilizzando come tecnica di suddivisione tra training set e test set la *cross-validation*. Essa è una tecnica statistica utilizzabile in presenza di una buona numerosità del training set. In particolare la k-fold cross-validation consiste nella suddivisione del dataset totale in k parti (si chiama anche k-fold validation) e, ad ogni passo, la parte (1/k)-esima del dataset viene ad essere il validation dataset, mentre la restante parte costituisce il training dataset. Così, per ognuna delle k parti (di solito k = 10) si allena il modello, evitando quindi problemi di overfitting, ma anche di campionamento asimmetrico (e quindi affetto da bias) del training dataset, tipico della suddivisione del dataset in due sole parti (ovvero training e validation dataset).

Valutazione dei risultati

In questa sezione sono descritte le principali metriche utilizzate in questo ambito per la valutazione delle performance e la conseguente scelta dell'algoritmo di classificazione con prestazioni migliori in relazione ad uno specifico dataset.

- Percentuale di istanze classificate correttamente

  E' la misura più semplice per la valutazione delle performance della classificazione. E'calcolata come la semplice percentuale di istanze classificate correttamente, rispetto all'intero test set.

- Matrice di confusione

  La matrice di confusione per un modello con classe binaria (matrice 2x2) è:

```
--- Confusion Matrix ---
a       b <-- classified as
VP    FN | a = yes
FP    VN | b = no
```

  Gli elementi della matrice sono di seguito definiti:

  – Veri Positivi: numero di record positivi correttamente classificati come positivi;

  – Falsi Negativi: numero di record positivi erroneamente classificati come negativi;

  – Veri Negativi: numero di record negativi correttamente classificati come negativi;

  – Falsi Positivi: numero di record negativi erroneamente classificati come positivi.

- Curva ROC

  Altra importante misura di prestazione è la curva ROC, che costituisce un metodo grafico di valutazione dei risultati. La curva ha:

  – per ascisse il FPr (False Positive rate), ossia la frazione di casi negativi erroneamente classificati come positivi (Falsi Positivi) rispetto al numero totale di casi negativi presenti nel dataset (Falsi Positivi+Veri Negativi);

  - per ordinate il TPr (True Positive rate), ossia la frazione di casi positivi correttamente classificati come positivi (Veri Positivi) rispetto al numero totale di casi positivi presenti nel dataset (Veri Positivi+Falsi Negativi).

  Quanto più la curva ROC si avvicina all'angolo in alto a sinistra del sistema di riferimento (ossia per alto TPr e basso FPr), migliore è la prestazione del corrispondente modello. Altra interessante misura di prestazione è l'AUC (Area Under Curve), ossia la superficie sottesa dalla curva ROC.

- Precision e Recall

  Precisione e Recall sono due classificazioni statistiche molto usate. La precisione può essere vista come una misura di correttezza o fedeltà, mentre la recall è una misura di completezza. In un processo di classificazione, i termini vero positivo, vero negativo, falso positivo e falso negativo sono usati per confrontare la classificazione di un oggetto (l'etichetta di classe assegnata all'oggetto da un classificatore) con la corretta classificazione desiderata (la classe a cui in realtà appartiene l'oggetto). Precisione e recall sono definite come:

  $$\textit{Precisione}: \frac{VeriPositivi}{VeriPositivi+FalsiPositivi} \text{ , } \textit{Recall:} \frac{VeriPositivi}{VeriPositivi+FasiNegativi}$$

Previsione di eventuali anomalie/cambiamenti

Sulla base dell'applicazione al dataset dell'algoritmo di classificazione scelto nella fase precedente viene modellizzato il comportamento dell'applicazione in relazione ai parametri contestuali considerati. Vengono poi monitorati o stimati a runtime i dati contestuali attuali ed effettuata una previsione sul comportamento del sistema rilevando eventuali anomalie o cambiamenti e utilizzando il modello ottenuto dall'applicazione dell'algoritmo di classificazione scelto.

Scelta del sottoprocesso da eseguire

A seconda di ciò che è stato predetto nella fase precedente viene invocato dinamicamente il sottoprocesso più idoneo.

Riassumendo, i dati storici estratti dal database vengono utilizzati come dataset, una parte come training set, per modellizzare il comportamento della rete ed una parte come test set, per verificare la bontà del modello creato. La scelta dell'algoritmo di classificazione viene effettuata sulla base del confronto tra le misure di qualità del modello ottenute utilizzando diversi meccanismi di classificazione. Dopo aver scelto l'algoritmo più idoneo vengono monitorati o stimati i parametri contestuali attuali, effettuata una previsione sul comportamento dell'applicazione in questa situazione e scelto in accordo il sottoprocesso da svolgere.

3.6 Architettura generale del framework

Lo scopo del framework è quello di fornire un meccanismo di invocazione dinamica ed automatica dei servizi considerando in particolare un'analisi proattiva del contesto in cui è sviluppata ed eseguita l'applicazione in riferimento alla granularità dei dati e alla frequenza con cui essi devono essere raccolti.

In questa sezione ci si occuperà di descrivere le componenti di cui è composto il framework e le problematiche per cui sono utilizzate.
L'architettura generale del framework è mostrata in figura 3.9.

Il framework si compone di tre componenti principali, ognuna delle quali è responsabile della gestione dell'applicazione in una specifica fase del suo ciclo di vita. Nella prima fase ci si occupa della realizzazione e pubblicazione dei web service necessari allo svolgimento delle operazioni, oppure, in alternativa, alla ricerca di web service idonei preesistenti pubblicati nei registri web. Solitamente i servizi sono strettamente correlati ai dati contestuali dello scenario di riferimento, perciò sarà necessaria la progettazione, realizzazione e connessione ad un database contestuale. In esso vengono descritti tutti

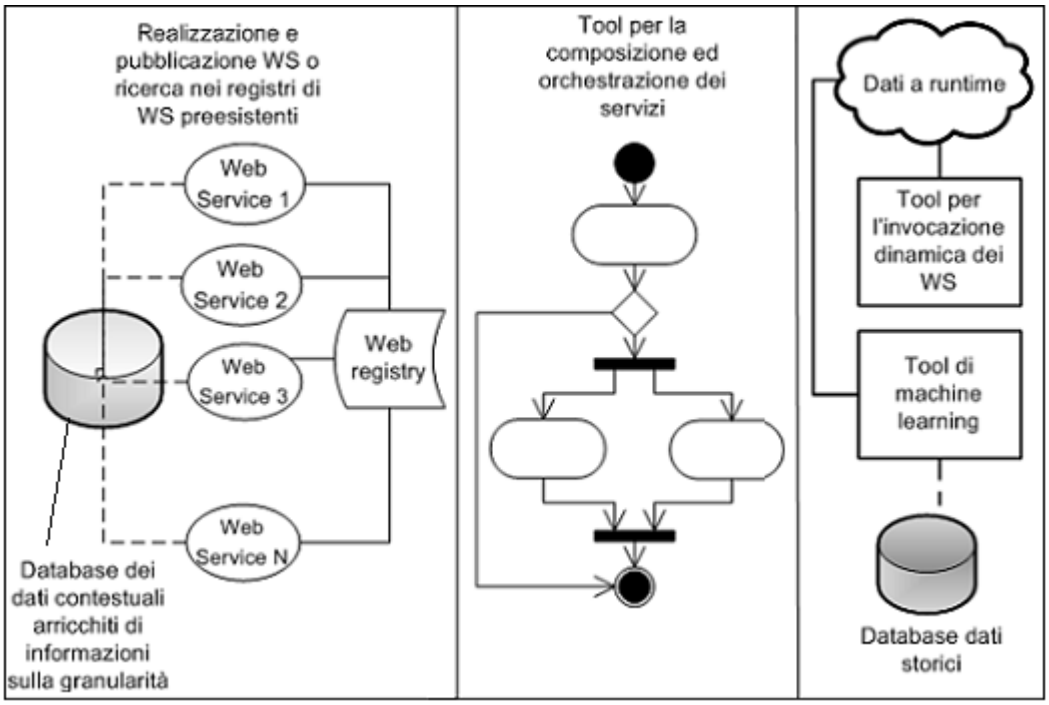

Figura 3.9: Architettura del framework

i dati significativi per l'applicazione e, dove significativo, associati alla loro *granularità*: per ogni parametro granulare viene realizzata una tabella che ne descriva i vari livelli di dettaglio. Nella figura 3.10 è mostrato un esempio di rappresentazione della granularità nel database in relazione a dati spaziali:

GRANULARITA' LOCAZIONE					
IdLocazione	Coordinate	Paese	Provincia	Regione	Stato
1	coord1	Dervio	LC	Lombardia	Italia
2	coord2	Milano	MI	Lombardia	Italia
3	coord3	Milano	MI	Lombardia	Italia

COORDINATE		
IdCoordinate	Latitudine	Longitudine
coord1	46,51	9,18
coord2	45,28	9,12
coord3	45,46	9,17

Figura 3.10: Granularità dei dati nel DB: dati spaziali

Ad alcuni dati è associata nel database anche una *frequenza di monitoraggio*, che sta ad indicare ogni quanto è significativo monitorare o ricalcolare un dato del contesto. Viene definita sulla base di considerazioni matematiche, storiche e contestuali e salvata in un opportuno campo della tabella corrispondente al dato, come mostrato in figura 3.11. Il valore della frequenza (o equivalentemente del periodo) è inizialmente calcolato in situazioni normali e verrà eventualmente aggiornato se saranno previste anomalie o cambiamenti.

PUNTO VENDITA			
CodicePV	Nome	Locazione	PeriodoMonitoring
1	Punto Vendita Milano	città	25
2	Punto Vendita Bormio	montagna	100
3	Punto Vendita Bellagio	mare	20

Figura 3.11: Granularità di monitoraggio nel DB

Una volta realizzati e pubblcati o scelti i web service è necessario un meccanismo che si occupi della composizione ed orchestrazione dei servizi. Esso si occupa di stabilire l'ordine di svolgimento delle operazioni e le condizioni sotto le quali alcuni servizi debbano essere svolti. Entrambe queste fasi si svolgono a design-time.

La terza fase è quella che si occupa della gestione dell'applicazione a runtime, cioè mette in atto meccanismi che entrano in gioco in relazione alla specifica istanza in esecuzione dell'applicazione. In particolare, per lo scopo che si vuole raggiungere con l'utilizzo di questo framework individuiamo:

- *Tool per l'invocazione dinamica dei servizi* che permetta in modo automatico e trasparente l'invocazione di un servizio concreto specifico a seconda di opportuni parametri a fronte di una stessa richiesta astratta dell'utente. Il tool viene configurato a design-time ma agisce a runtime in relazione ai valori dei parametri nella specifica istanza. In questo framework l'invocazione adattiva dei servizi sarà effettuata per la gestione, l'aggiornamento e l'utilizzo di granularità specifiche;

- *Tool di machine learning*, viene utilizzato allo scopo di modellizzare il comportamento dell'applicazione in relazione ad alcuni parametri contestuali mediante l'utilizzo di specifici algoritmi che permettano al sistema di evolvere migliorando le proprie performance. Una volta modellizzato il sistema sulla base di serie di dati storici il modello verrà utilizzato per prevedere il comportamento dell'applicazione con i valori attuali dei parametri del contesto. E'opportuno prevedere anticipatamente (proattivamente) anomalie o cambiamenti nel contesto di esecuzione in modo da risolvere il problema prima che si verifichi anzichè cercare di rimediarvi una volta che si è già riscontrato.

Capitolo 4

Scenario di riferimento: gestione delle scorte

Lo scopo di questo capitolo è quello di mostrare un esempio di applicazione del framework ad un caso di studio, a supporto dei concetti introdotti nel capitolo precedente a livello teorico. Lo scenario scelto è quello della gestione delle scorte in un supermercato alimentare.

4.1 Caso di studio: gestione delle scorte

La gestione delle scorte [25], [7] ha l'obiettivo di minimizzare i costi di mantenimento a magazzino delle scorte, pur garantendo una corretta alimentazione dei flussi produttivi. Le scorte vengono definite come una serie di materiali, semilavorati e prodotti che in un determinato momento sono in attesa di partecipare ad un processo di lavorazione o distribuzione.

La gestione delle scorte è importante per tutte le tipologie di imprese: sia in quelle *manifatturiere* per regolare correttamente lo svolgimento delle attività produttive che in quelle *commerciali* per soddisfare il cliente nei tempi e nelle quantità richieste.

Provenienza delle scorte

Per quanto riguarda la provenienza, le scorte possono essere distinte in *interne* quando risultano da processi di produzione interni all'impresa ed *esterne* quando la provvista viene effettuata presso fornitori esterni all'impresa.

Destinazione funzionale delle scorte

Le scorte possono essere di *materie prime* che servono per lo sviluppo dei flussi produttivi in entrata, destinati alla trasformazione, di *semilavorati*, cioè materiali che hanno subito qualche trasformazione ma non sono ancora pronti per la vendita oppure di *prodotti finiti*, ossia pronti per essere venduti. Il *livello di scorte* da detenere a magazzino deve tener conto dei seguenti fattori:

- investimento di capitali: le scorte assorbono capitali, per cui devono essere tenute a magazzino solo se necessarie;
- pericolo di obsolescenza/decadimento: deterioramento tecnologico e fisico dell'oggetto;
- spazio di magazzino;
- tempo di approvviginamento;
- tasso di vendita del prodotto;
- costi di conservazione/movimentazione;
- posizione finanziaria: passività a breve termine, debiti con i fornitori;
- svalutazione e investimento: analisi delle tendenze del mercato;
- sconti di quantità: il prezzo della scorta è legato al quantitativo acquistato;
- livello di servizio.

Riassumendo, gestire ottimamente le scorte di magazzino significa essere in grado di rispondere adeguatamente alle seguenti problematiche:

- *quanto* ordinare e conservare in magazzino;
- *quando* emettere un ordine di approvvigionamento.

4.1.1 Il problema del quanto ordinare

Minimizzazione dei costi implicati nella gestione delle scorte:

- costi di approvvigionamento (o di emissione dell'ordine);
- costi di mantenimento;
- costi di stock-out;
- costi di stock-over.

In *condizioni di certezza*, la quantità da ordinare è calcolata con il Modello di Wilson o del Lotto Economico di Acquisto, secondo cui il problema della gestione delle scorte è connesso unicamente ai costi di approvvigionamento e di mantenimento, e la quantità è calcolata profilando un trade-off tra i due costi. Secondo questo metodo (detto anche metodo delle scorte separate o Two Bin System), ogni volta che la giacenza di magazzino sta per scendere al di sotto del quantitativo predeterminato (livello di riordino) va spiccato un ordine per una quantità corrispondente al Lotto Economico d'Acquisto. In sintesi, le due grandezze da determinare per l'applicazione del modello di Wilson, sono appunto il livello di riordino e il lotto economico d'acquisto.

4.1.2 Il problema del quando ordinare

Il problema del quando ordinare si rifeisce alla determinazione dell'istante in cui emettere l'ordine di approvvigionamento. Per far fronte a questa

problematica si introduce il livello di riordino che coincide con la giacenza sufficiente a far fronte al consumo o alla vendita durante il periodo di approvvigionamento.

$$LR = T_A.t_c$$

dove,

T_A = tempo di approvvigionamento

t_c= tasso di consumo

Condizioni di incertezza

Non esiste un sistema di fornitura perfetto, affidabile e sicuro e nemmeno un esatto metodo per valutare l'andamento della domanda. Le merci potrebbero quindi essere consegnate in anticipo o in ritardo rispetto alle attese e la domanda potrebbe essere superiore o inferiore rispetto alle previsioni. Introduciamo a questo scopo la scorta di sicurezza (SS) cioè la quantità di scorta a cui è delegato il compito di assorbire le variazioni nel tempo della domanda e degli approvvigionamenti. L'ammontare della SS si lega alla variabilità della domanda e alla variabilità del periodo di approvvigionamento.

$$LR = T_A.t_c + SS$$

4.1.3 Gestione delle scorte in un supermercato alimentare

Le scorte sono di prodotti finiti, esterne e relative ad un'impresa commerciale.

Le forniture possono essere effettuate secondo due meccanismi distinti, come mostrato in figura 4.1, la logistica integrata e quella non integrata.

Con *logistica integrata* si intende una situazione in cui sono presenti alcuni magazzini centralizzati connessi a diversi punti vendita. Ogni punto vendita non ha quindi contatto diretto con i fornitori, ma si rifornisce attraverso i magazzini centrali a cui è collegato. Al contrario nella *logistica non integrata* i punti vendita si riforniscono attraverso un contatto diretto con i fornitori di prodotti.

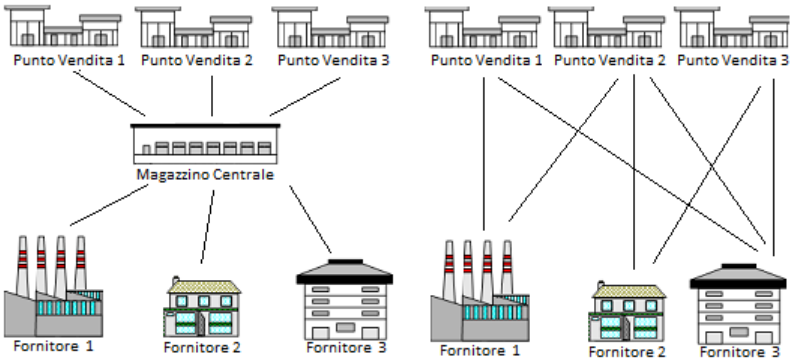

Figura 4.1: Logistica integrata e Logistica non integrata

Problema del quanto e quando ordinare

Il "quanto" ordinare e il "quando" ordinare, all'interno dei punti vendita, in situazioni di certezza sono determinati sulla base di misure quantitative, calcolando il livello di riordino sulla base della valutazione di parametri come quelli descritti nella sezione precedente. Quando il volume di merce nel magazzino scende sotto il *punto di riordino*, viene effettuato un ordine al magazzino centrale, di un quantitativo di merce calcolato a priori. Il magazzino centrale sarà sempre fornito delle scorte necessarie a fornire tutti i punti vendita connessi in caso di richieste "regolari", come mostrato in figura 4.2.

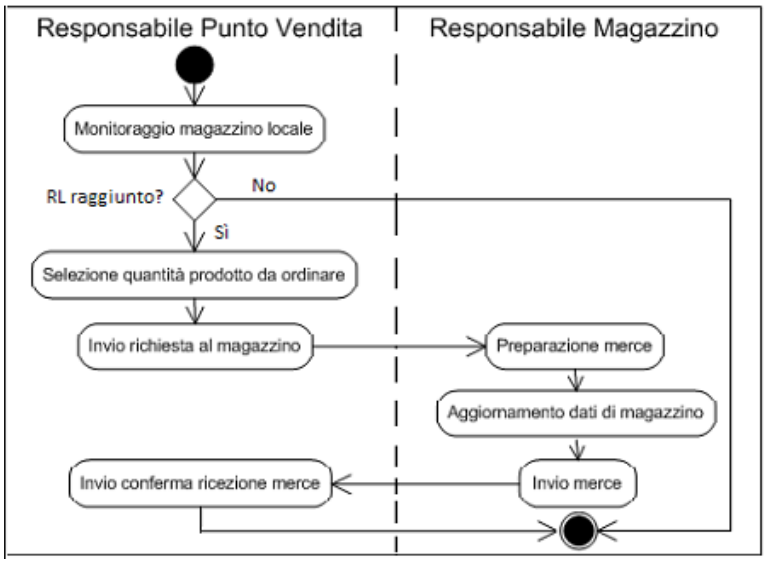

Figura 4.2: Magazzino in situazioni regolari

Questa strategia non è ottimale in *situazioni di incertezza*. Nei punti vendita potrebbero verificarsi anomalie nella domanda e di conseguenza la quantità di prodotto da ordinare e la frequenza con cui sarà richiesta merce al magazzino saranno differenti da quelle standard. Per questo motivo il magazzino centrale dovrà a sua volta analizzare le richieste provenienti dai punti vendita a cui è connesso per prevenire situazioni di stock-out o stock-over. Il diagramma seguente mostra l'interazione tra punto vendita, magazzino centrale e fornitore nel caso in cui si verifichi una situazione per la quale il magazzino centrale non sia fornito di un numero necessario di prodotti per far fronte alla richiesta del cliente:

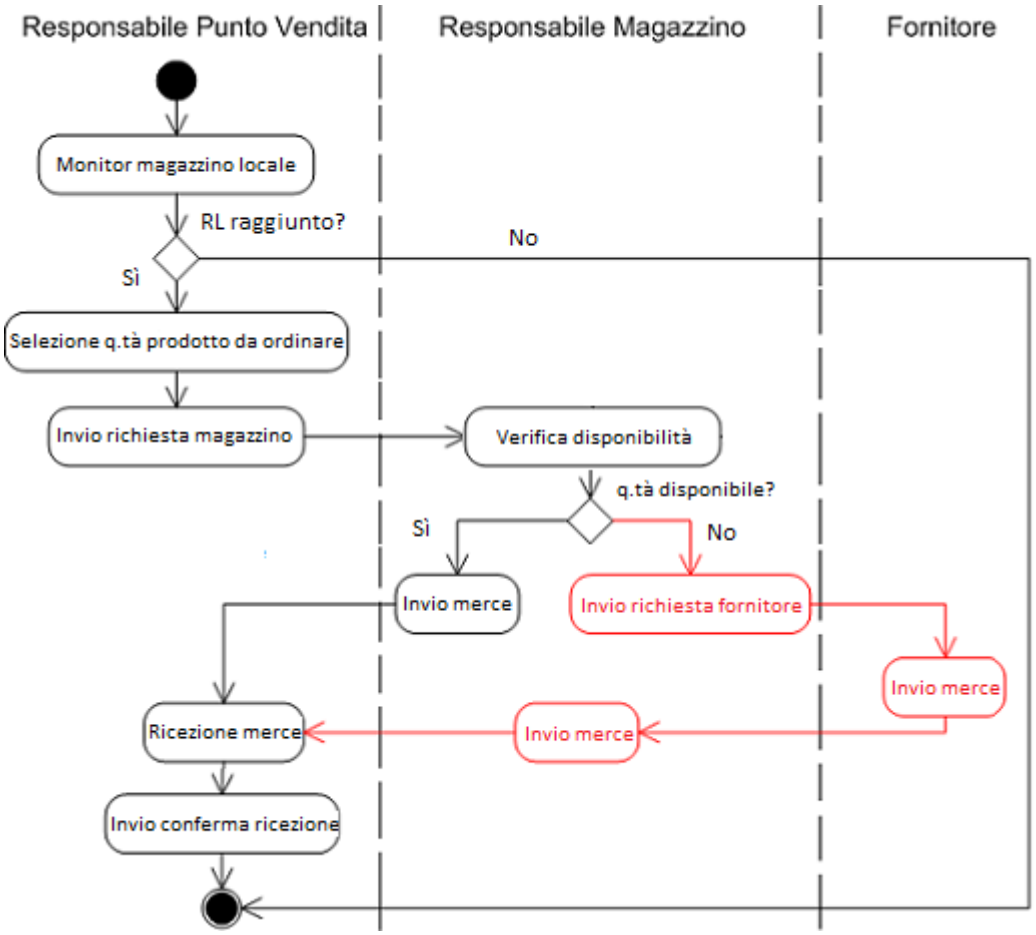

Figura 4.3: Magazzino in situazioni non regolari: stock-out

Come mostrato in figura 4.3, il tempo aggiuntivo per effettuare ulteriori richieste al fornitore è significativo. Sarebbe quindi preferibile evitare l'intervento diretto del fornitore, mantenendo sempre il magazzino centrale aggiornato anche in situazioni anomale.

4.2 Profilazione granulare del fornitore

In questo scenario, si verifica l'esigenza di scegliere il fornitore più idoneo per la fornitura di ogni prodotto. Per fare ciò viene eseguito un processo granulare di selezione del fornitore.
A seconda del contesto di esecuzione, descritto in questo caso dall'importanza del prodotto in termini di fatturato, verrà scelto dinamicamente un servizio concreto da invocare, al giusto livello di dettaglio. I dati utilizzati nel servizio scelto avranno un diverso peso e un diverso livello di dettaglio a seconda dell'importanza del prodotto a cui ci si riferisce. In particolare si considera come granulare il parametro relativo alla localizzazione, sia del punto vendita che del fornitore.
Il processo di selezione granulare del fornitore è stato scelto allo scopo di mostrare l'applicazione pratica dei concetti di:

- svolgimento di sottoprocessi a diverse granularità a seconda di opportuni parametri contestuali;
- invocazione dinamica dei servizi, attraverso la configurazione di un proxy;
- adattività di tipo reattivo.

4.2.1 Descrizione del problema della selezione granulare del foritore

Il livello di attenzione rivolto dal management alla gestione delle giacenze di magazzino può essere differenziato in funzione del grado di "importanza" che

le stesse rivestono all'interno dell'organizzazione produttiva. La definizione della metodologia di gestione delle scorte può essere ottenuta attraverso l'analisi ABC, che consente di misurare il peso relativo, ad esempio in termini di fatturato, dei diversi cluster di materiali da gestire.

La *tecnica ABC* [1] si basa sul principio di Pareto, detto anche Legge 80/20. Secondo tale teorema, la maggior parte degli effetti dipende da un numero limitato di cause (approssimando, risulta che l'80% degli effetti dipende dal 20% delle cause). Tale analisi permette di definire quali sono gli articoli su cui focalizzare la propria attenzione. Si procede alla divisione degli articoli in 3 classi (A, B, C), facendo ricadere nella classe A gli articoli che nella cumulata danno origine a un valore approssimativo dell'80% (secondo la legge di Pareto). Nella classe B ricadono gli articoli che nella cumulata sono presenti nella fascia immediatamente successiva, dall'80% al 90%. Nella classe C si trovano, invece gli articoli che occupano la fascia complementare per arrivare al 100%.

Il comportamento nei confronti degli articoli è differente a seconda della classe in cui ognuno di essi ricade. La classe A richiede particolare attenzione in quanto si tratta della classe che genera il maggiore fatturato ed è particolarmente richiesta. Di conseguenza è buona norma prevederne un'adeguata scorta in modo da evitare situazioni di stock out che sarebbero particolarmente gravi, visto che si tratta di articoli molto richiesti e che generano ampia quota del fatturato. È necessario prestare particolare attenzione al fine di evitare costi gestione eccessivamente elevati. La classe B denota una minore criticità, vista la minore influenza sul fatturato dell'impresa. La classe C, invece, è un settore a bassa criticità che ha impatto ridotto sul fatturato aziendale e ad essa può essere dedicata minore attenzione in fase operativa.

La selezione del fornitore viene effettuata periodicamente all'interno del magazzino centrale (logistica integrata). Per ogni prodotto viene determinata l'esigenza di scorte e viene profilato il fornitore più idoneo, analizzandolo a diversi livelli di dettaglio a seconda dell'importanza del prodotto, salvata all'interno del database che descrive il contesto di esecuzione. Il parametro che

in questo scenario è utilizzato in relazione alla sua granularità è la locazione sia del fornitore che del punto vendita.

Viene realizzato un servizio astratto *SelezioneFornitoreAbstract* che descriva tramite un'interfaccia astratta l'operazione di selezione del fornitore. A questo servizio sono associati tre servizi concreti, ognuno dei quali descriva la selezione del fornitore relativa ad una delle classi di importanza del prodotto. I metodi si differenzieranno nel peso attribuito ai parametri in base a cui valutare i fornitori e sulla granularità dei dati utilizzata. In particolare, saranno progettati come segue:

1. PROFILAZIONE MOLTO DETTAGLIATA: svolta per i prodotti di tipo A che rappresentano l'80% del fatturato nei punti vendita.

   (a) *Locazione del fornitore* (granularità massima): è importante che il fornitore si trovi il più vicino possibile in quanto i prodotti di tipo A non possono mai mancare all'interno del magazzino.Gli ordini di questo tipo di prodotti saranno quindi frequenti e consistenti; la distanza di alcuni km tra un fornitore e l'altro potrebbe quindi essere determinante. Peso parametro: 0,3

   (b) *Qualità della fornitura:* massima (scala 1-10: $qualitaFornitore > 8$. Peso parametro: 0,3

   (c) *Feedback del fornitore:* massima (scala 1-10: $qualitaFornitore > 90$. Peso parametro: 0,3

   (d) *Costi fornitura:* minimizzati. Peso parametro: 0,1 (per prodotti di questo livello di importanza per il punto vendita non è un parametro così rilevante)

2. PROFILAZIONE MEDIAMENTE DETTAGLIATA: svolta per gli articoli della fascia B che nella cumulata sono presenti nella fascia immediatamente successiva, dall'80% al 90%.

   (a) *Locazione del fornitore* (granularità intermedia). I fornitori devono trovarsi nella stessa provincia rispetto al punto vendita.

(b) *Qualità del fornitore:* medio-alta (scala 1-10: $qualitaFornitore > 7$. Peso parametro: 0,3

(c) *Feedback del fornitore:* massima (scala 1-10: $qualitaFornitore > 85$. Peso parametro: 0,3

(d) *Costi fornitura:* minimizzati. Peso parametro: 0,4

3. PROFILAZIONE POCO DETTAGLIATA: svolta per gli articoli di tipo C che occupano la fascia complementare per arrivare al 100%.

   (a) *Locazione del fornitore* (granularità bassa): ordini poco frequenti, tasso di vendita prodotti molto basso, impatto minimo sul fatturato e sull'immagine del supermercato in caso di mancanza dell'articolo. E' sufficiente che i fornitori si trovino nella stessa regione rispetto al punto vendita.

   (b) *Qualità del fornitore:* media (scala 1-10: $qualitaFornitore > 6$. Peso parametro: 0,25

   (c) *Feedback del fornitore:* massima (scala 1-10: $qualitaFornitore > 80$. Peso parametro: 0,25

   (d) *Costi fornitura:* minimizzati. Peso parametro: 0,5

Lo svolgimento di una profilazione molto dettagliata del fornitore comporta uno sforzo più significativo in termini operazionali rispetto agli altri due servizi. Il dato della locazione del fornitore e del punto vendita alla granularità più fine (coordinate) devono essere estratti molto accuratamente per evitare errori, attraverso un procedimento costoso in termini energetici. Inoltre il calcolo delle distanze tra coordinate è molto più dispendioso in termini operazionali rispetto ad un semplice confronto sulla provincia o sulla regione di appartenenza. E' preferibile quindi prevedere lo svolgimento di un processo così dispendioso solo nel caso in cui sia strettamente necessario per il buon funzionamento del supermercato, in riferimento quindi ai prodotti di tipo A, con un forte impatto sul fatturato.

Nello scenario considerato, viene infatti effettuata una profilazione granulare del fornitore sulla base della classe del prodotto che si intende ordinare, come descritto nel diagramma in figura 4.4. L'attività di selezione del fornitore è descritta dal servizio astratto *SelezioneFornitoreAbstract* e a runtime viene invocato dinamicamente il servizio concreto più idoneo a seconda dell'importanza del prodotto.

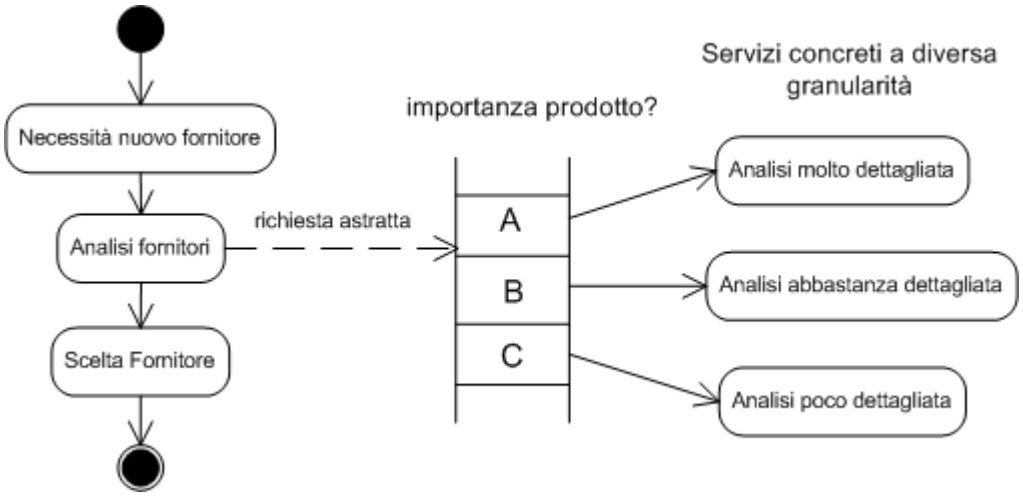

Figura 4.4: Selezione fornitore granulare

La granularità della locazione del fornitore è salvata all'interno del database che descrive il contesto esecutivo (Figura 4.5). E' realizzata una tabella, associata al fornitore, in cui sia descritta la sua locazione ad ogni livello di dettaglio.

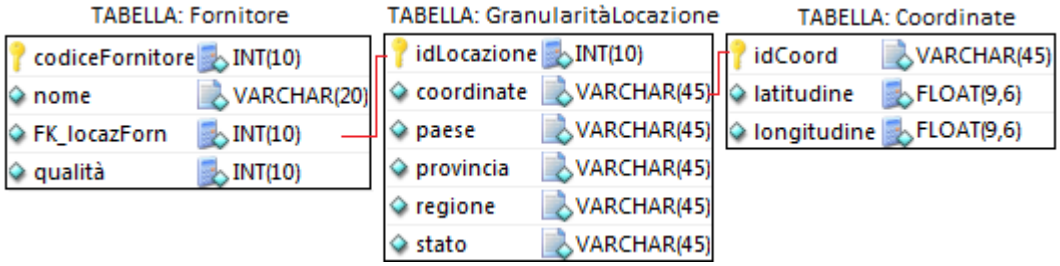

Figura 4.5: Tabelle del Database per il salvataggio della granularità informativa

4.3 Previsione proattiva della domanda

Una corretta previsione della domanda rappresenta l'input fondamentale per pianificare e coordinare le principali attività di ogni impresa, in quanto costituisce il punto di partenza per sviluppare il piano di produzione, per stabilire quale politica di approvvigionamento adottare, per pianificare le attività di distribuzione e per definire le future strategie di marketing.

In particolare, per quanto riguarda lo scenario di gestione delle scorte a magazzino, è opportuno prevedere proattivamente anomalie nella domanda per evitare situazioni di stock-out e stock-over prima che si verifichino, rivalutando l'entità della merce da ordinare e la frequenza di monitoraggio del magazzino mediante l'analisi di serie di dati storici, correlate ad opportuni parametri contestuali.
Questo processo è considerato allo scopo di fornire un esempio di adattività proattiva.
In particolare è necessario realizzare un dataset che tenga in considerazione i parametri contestuali che influenzino la domanda. Per quanto riguarda lo scenario considerato vengono individuati i seguenti:

- condizioni atmosferiche (sole, nuvole, pioggia, neve);
- presenza di vacanze (rapporto tra giorni di vacanza nel mese e giorni di cui è composto il mese);
- stagione (estate, autunno, inverno, primavera);
- locazione (mare, montagna, città).

Nel database di dati storici sono salvate alcune istanze che descrivano il comportamento della domanda in relazione a dati contestuali passati e che verranno utilizzate per effettuare previsioni sul comportamento attuale.
Il dataset è salvato come una serie di istanze classificate costituite da record di attributi contestuali (es. sole, 3.1, estate, mare) etichettati con l'andamento della classe domanda (in calo, costante, in crescita) nella specifica situazione.

```
@data
neve,3,inverno,montagna,crescita
pioggia,5.1,primavera,mare,calo
                        ....
sole,6.2,estate,città,calo
nuvole,6,estate,montagna,costante
```

Queste istanze vengono utilizzate per creare un modello che descriva l'andamento della domanda in relazione alle condizioni contestuali e per verificarne la bontà.
Vengono eseguiti diversi algoritmi di classificazione sul dataset considerato allo scopo di valutarne le performance e scegliere in accordo il più idoneo, sulla base di opportune metriche di qualità.
Una volta scelto l'algoritmo e realizzato il modello che descriva il comportamento della domanda in relazione ai parametri contestuali, questo viene utilizzato per la classificazione dell'istanza attuale dei parametri contestuali misurati, in modo da ottenere l'andamento previsto della domanda attuale.

Esempio:

Nel caso di studio vengono considerati tre punti vendita, ognuno di quali si trova in una tipologia di locazione differente (mare, montagna, città). Per lo scenario specifico, si considera che le località lacustri abbiano le stesse caratteristiche in termini di andamento della domanda di quelle marine. Per questo motivo vengono indicate anch'esse con la locazione mare.
Punto vendita 1: Bellagio

- stagione: maggior afflusso di turisti nella stagione estiva;
- condizioni atmosferiche: località marina, maggior afflusso di turisti nelle giornate soleggiate;
- presenza di festività/vacanze: spesso presenti seconde case, sfruttate in particolare nei weekend e nei periodi di vacanza (estive);

- classe da stimare: andamento domanda (in calo, costante, in crescita).

Punto vendita 2: Milano

- stagione: maggior afflusso nella stagione lavorativa;
- condizioni atmosferiche: in caso di giornate soleggiate spesso anche i residenti si allontanano dalla città;
- presenza di festività/vacanze: zona lavorativa, minor afflusso clienti durante le vacanze;
- classe da stimare: andamento domanda (in calo, costante, in crescita).

Punto vendita 3: Bormio

- stagione: maggior afflusso di turisti nella stagione invernale;
- condizioni atmosferiche: presenza di piste da sci, frequentate soprattutto in giornate soleggiate, quando è garantita una buona visibilità sulle piste ed in presenza di neve;
- presenza di festività/vacanze: spesso presenti seconde case, affittate terzi o sfruttate dai proprietari in particolare nei weekend e nei periodi di vacanza (invernali);
- classe da stimare: andamento domanda (in calo, costante, in crescita).

Questa analisi proattiva dell'andamento della domanda viene effettuata in ogni punto vendita. Il responsabile del magazzino centrale, raccoglie i dati ricevuti da ogni punto vendita, li analizza e aggiorna la richiesta da effettuare al fornitore, come mostrato in figura 4.6.

L'analisi dell'andamento della domanda viene inserita nel processo di aggiornamento della granularità di monitoraggio, allo scopo di prevedere in anticipo e far fronte ad eventuali anomalie nell'andamento della domanda, evitando il verificarsi di situazioni di stock-out o stock-over. L'analisi è effettuata a periodi regolari di tempo (es. mensilmente, al termine del mese) e si riferisce al periodo di tempo successivo.

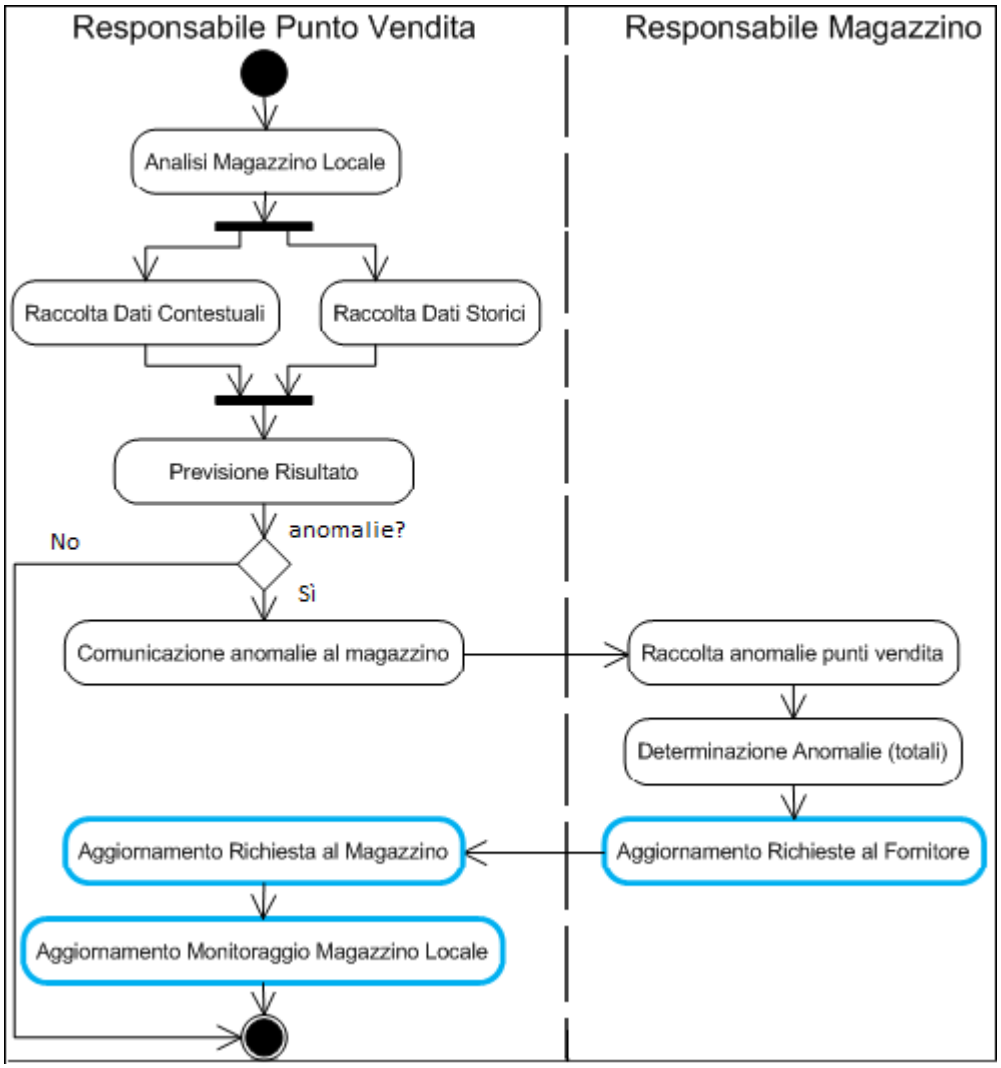

Figura 4.6: Previsione proattiva della domanda

4.4 Aggiornamento periodo monitoraggio del magazzino

L'incertezza ed il rischio sono da sempre connaturati al mondo degli affari e sono accentuati dalla crescente complessità degli scenari di mercato. La conoscenza della domanda futura rappresenta l'informazione fondamentale per la gestione delle scorte sia all'interno di una singola azienda che all'interno

della supply chain.

In questo processo viene descritto l'aggiornamento di alcuni dati contestuali (tra cui la granularità di monitoraggio) a seguito della previsione proattiva delle anomalie nell'andamento della domanda. Si è scelto di descrivere questo processo allo scopo di esemplificare i concetti di:

- granularità di monitoraggio;
- adattività proattiva;
- invocazione dinamica dei servizi.

La previsione sull'andamento della domanda viene effettuata allo scopo di aggiornare in accordo la granularità di monitoraggio del magazzino e la quantità di prodotto da ordinare per evitare il verificarsi di situazioni di stock-out e stock-over.

Nello scenario considerato, si assume che:

- il monitoraggio del magazzino venga effettuato ad intervalli discreti, regolari e periodici;
- le richieste al fornitore vengano effettuate quando il livello di scorta a magazzino scende sotto il livello di riordino, e siano costituite da un quantitativo fisso di merce, calcolato in relazione ad opportuni parametri, tale da garantire la corretta alimentazione dei flussi produttivi in un periodo pari al tempo di approvvigionamento;
- sia il dato che rappresenta la granularità di monitoraggio del magazzino che quello che rappresenta la quantità di riordino siano salvati all'interno del database contestuale.

Si suppone che il monitoraggio del magazzino in situazioni normali venga effettutato ad intervalli di tempo discreti con una cadenza T0, che il quantitativo fisso di merce ordinata al magazzino centrale per il prodotto P sia K0 e il quantitativo richiesto ai fornitori dal magazzino centrale sia F0.

I valori di T0, K0 e F0 sono definiti sulla base di considerazioni matematiche ed economiche, in modo da garantire la corretta alimentazione dei flussi distributivi in condizioni non anomale.

Nel singolo punto vendita: Il comportamento dell'applicazione nel singolo punto vendita è mostrato nella figura 4.7.

- Situazione normale: nessuna anomalia nell'andamento della domanda rilevata da WEKA. Non viene aggiornato alcun parametro nel database contestuale, essendo i parametri di base definiti sulla base della corretta alimentazione dei flussi distributivi in condizioni normali.
  - monitoraggio magazzino locale: T=T0
  - richieste al magazzino centrale: K =K0
- Domanda crescente: viene rilevato che la domanda sarà crescente nel prossimo periodo in un dato punto vendita. E'quindi necessario aggiornare i valori dei parametri contestuali in questo modo:
  - T=T-T1 il monitoraggio del magazzino viene effettuato più spesso.
  - K=K+K1 viene proposto all'utente dell'applicazione di richiedere un quantitativo maggiore di merce al magazzino, per evitare situazioni di stock-out. La scelta finale della quantità da ordinare sarà comunque effettuata dal compratore, tenendo conto anche delle condizioni contrattuali.
- Domanda in calo: viene rilevato che la domanda sarà in calo nel prossimo periodo in un dato punto vendita. E'quindi necessario aggiornare i valori dei parametri contestuali in questo modo:
  - T=min(T0,T+T1) il monitoraggio del magazzino viene effettuato meno spesso, mantenendo però un periodo minimo di T0.
  - K=K-K1 viene proposto all'utente dell'applicazione di richiedere un quantitativo minore di merce al magazzino, per evitare situazioni di

stock-over. La scelta finale della quantità da ordinare sarà comunque effettuata dal compratore, tenendo conto anche delle condizioni contrattuali.

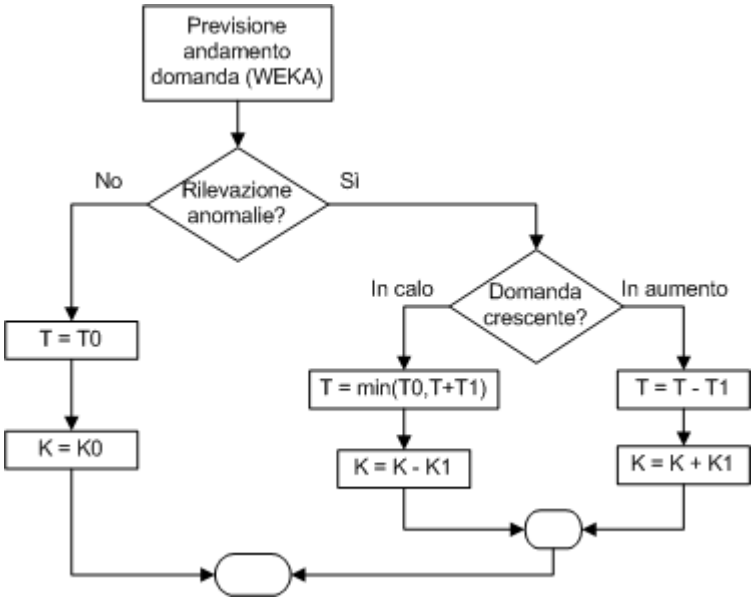

Figura 4.7: Granularità monitoraggio singolo punto vendita

Nel magazzino centrale: Il comportamento dell'applicazione nel magazzino centrale è mostrato nella figura 4.8.

- Domanda TOTALE non anomala: non vengono riscontrate anomalie nell'andamento della domanda totale dei punti vendita connessi al magazzino. Non è necessario effettuare aggiornamenti nei dati contestuali di magazzino.

  – F=F0

- Domanda TOTALE crescente: il responsabile del magazzino centrale rileva una domanda crescente, riferendosi ai dati ricevuti da tutti i punti vendita connessi. E'necessario effettuare il seguente aggiornamento nel database contestuale del magazzino centrale:

– F=F0+F1 viene proposto all'utente dell'applicazione di richiedere un quantitativo maggiore di merce al fornitore, per evitare situazioni di stock-out. La scelta finale della quantità da ordinare sarà comunque effettuata dal compratore, tenendo conto anche delle condizioni contrattuali stipulate con il fornitore.

- Domanda TOTALE calante: il responsabile del magazzino centrale rileva una domanda calante, riferendosi ai dati ricevuti da tutti i punti vendita connessi. E'necessario effettuare il seguente aggiornamento nel database contestuale del magazzino centrale:

  – F=F0-F1 viene proposto all'utente dell'applicazione di richiedere un quantitativo minore di merce al fornitore, per evitare situazioni di stock-over. La scelta finale della quantità da ordinare sarà comunque effettuata dal compratore, tenendo conto anche delle condizioni contrattuali stipulate con il fornitore.

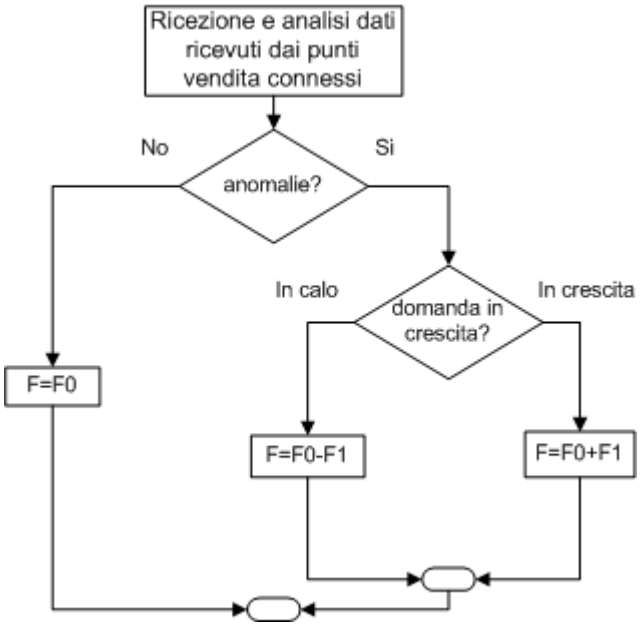

Figura 4.8: Granularità monitoraggio magazzino centrale

Il processo sarà realizzato mediante la creazione di un servizio che si occupi, mediante tecniche di machine learning e utilizzando algoritmi di classificazione, di prevedere l'andamento della domanda. Sulla base dell'andamento rilevato a runtime, sarà invocato un servizio astratto *AggiornamentoMagazzinoAbstract* che si occuperà di invocare dinamicamente il servizio concreto idoneo per effettuare il corretto aggiornamento dei parametri di magazzino, come mostrato in figura 4.9. In particolare:

- AggiornamentoMagazzinoCrescita: viene diminuito il periodo di monitoraggio del 20% e aumentata la quantità di prodotto da ordinare della stessa percentuale;
- AggiornamentoMagazzinoCalo: viene aumentato il periodo di monitoraggio del 20%, mantenendo un periodo minimo di monitoraggio, e diminuita la quantità di prodotto da ordinare;
- AggiornamentoMagazzinoCostante: non viene effettuata alcuna modifica ai dati di magazzino.

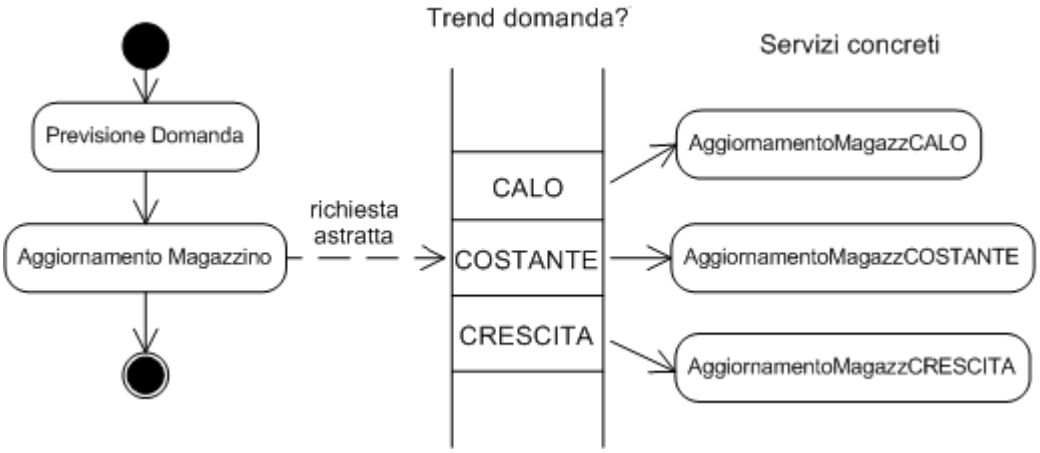

Figura 4.9: Processo di aggiornamento proattivo della granularità di monitoraggio

Capitolo 5

Implementazione

In questo capitolo sono mostrate le tecnologie utilizzate per la realizzazione del framework. Ci si riferisce in particolare allo scenario della gestione delle scorte, ma si cerca di generalizzare il più possibile, allo scopo di fornire una panoramica sugli strumenti da utilizzare nella realizzazione del framework a supporto di un qualunque senario di esecuzione.

5.1 Progettazione e realizzazione database

Il database viene realizzato allo scopo di contenere tutte le informazioni e i dati riguardanti lo scenario di esecuzione.

5.1.1 Progettazione: diagrammi ER

Il diagramma ER (Entità-Relazione) è realizzato allo scopo di rappresentare concettualmente i dati del dominio dell'applicazione ad un alto livello di astrazione. Lo schema concettuale del database considerato è descritto nella figura 5.1.

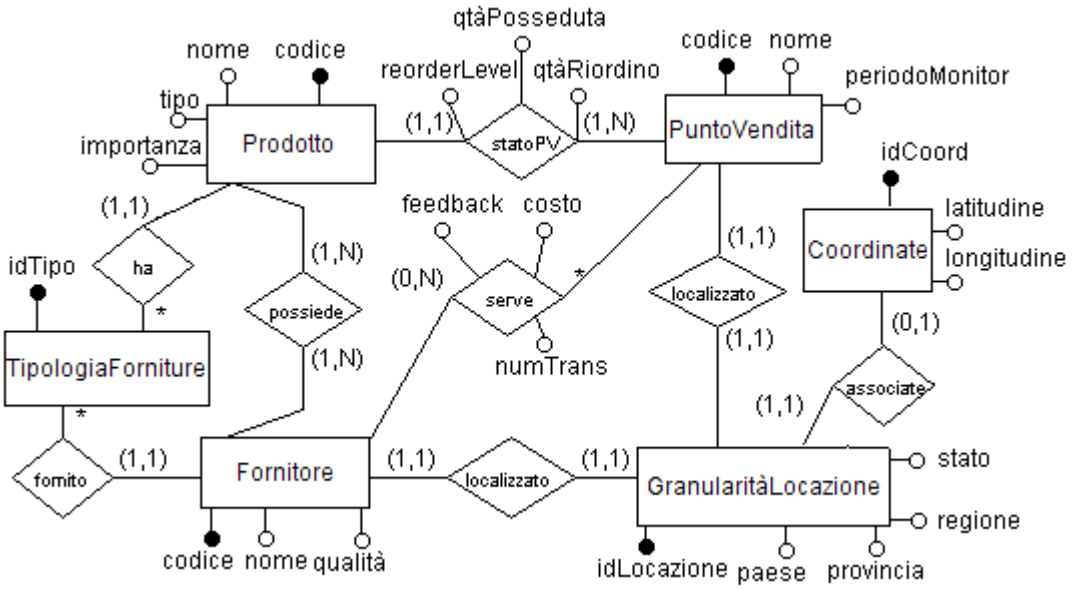

Figura 5.1: Diagramma ER dell'applicazione: modello concettuale

Entità

Rappresentano le classi di oggetti che descrivono il dominio applicativo. Ogni entità è caratterizzata da una serie di attributi che la caratterizzano.

Per quanto riguarda lo scenario considerato, viene trattato il caso di logistica non integrata, eliminando il concetto di magazzino centrale. Questa semplificazione facilita la progettazione e realizzazione dell'applicazione e non provoca alcuna perdita di generalità, in quanto il magazzino centrale si comporta a turno come punto vendita o come fornitore. Vengono individuate le seguenti entità:

- PRODOTTO: rappresenta il prodotto presente in alcuni punti vendita e fornito da specifici fornitori
  - *codice:* è la chiave primaria identificativa del prodotto. E'descritto dal suo codice a barre;
  - *nome:* nome specifico del prodotto;
  - *tipo:* rappresenta la categoria merceologica a cui appartiene il prodotto;

  - *importanza:* rappresenta il livello di incidenza del prodotto considerato sul fatturato, secondo la legge di Pareto descritta precedentemente. L'attributo importanza può assumere i valori A, B, C;
  - *FK fornitore:* chiave esterna relativa alla tabella fornitore per indicare quale sia il fornitore preassegnato ad ogni prodotto.

- TIPOLOGIA FORNITURE: indica le tipologie di prodotti che fornisce un determinato fornitore
  - *id Fornitura:* è la chiave primaria identificativa della tipologia della fornitura. E' descritta da un valore intero numerico;
  - *FK fornitore:* chiave esterna relativa al codice del fornitore;
  - *FK tipoProdotto:* chiave esterna relativa alla tipologia dei prodotti.

- PUNTO VENDITA: rappresenta i punti vendita in cui sono venduti i prodotti
  - *codice:* è la chiave primaria identificativa del punto vendita. E' descritta da un valore intero numerico;
  - *nome:* nome specifico del punto vendita;
  - *periodo monitoring:* indica la granularità di monitoraggio del punto vendita;
  - *FK locazione:* chiave esterna riferita alla tabella granularità locazione.

- FORNITORE: descrive coloro che forniscono i punti vendita con opportuni prodotti
  - *codice:* è la chiave primaria identificativa del fornitore. E' descritta da un valore intero numerico;
  - *nome:* nome specifico del fornitore;

  - *qualità:* indica la qualità della fornitura su una scala da 1-10;
  - *FK locazione:* chiave esterna riferita alla tabella granularità locazione.

- GRANULARITA'LOCAZIONE: indica in maniera granulare la locazione di un'entità
  - *id Locazione:* è la chiave primaria identificativa della locazione. E' descritta da un valore intero numerico;
  - *FK coordinate:* chiave esterna riferita alla tabella coordinate. Indica la locazione a livello di granularità coordinate;
  - *paese:* indica la locazione a livello di granularità paese;
  - *provincia:* indica la locazione a livello di granularità provincia;
  - *regione:* indica la locazione a livello di granularità regione;
  - *stato:* indica la locazione a livello di granularità stato.

- COORDINATE: sono associate ad ogni locazione e rappresentano i valori della locazione in coordinate geografiche
  - *id Coordinate:* è la chiave primaria identificativa delle coordinate. E' descritta da una stringa;
  - *latitudine:* indica la latitudine di un punto. E'descritta da un FLOAT(9,6) ;
  - *longitudine:* indica la longitudine di un punto. E'descritta da un FLOAT(9,6).

Relazioni

Descrivono le relazioni in cui si trovano le entità tra loro. Possono anch'esse essere dotate di attributi che le descrivano. Nello scenario vengono individuate:

- FORNISCE: relazione tra Fornitore e Prodotto. Indica il fornitore preassegnato per la fornitura di uno specifico prodotto.
- POSSIEDE: relazione tra Fornitore e Prodotto. Indica le categorie di prodotti fornite da uno specifico fornitore.
- STATO PV: relazione tra Punto Vendita e Prodotto. Indica tutti prodotti presenti in ogni punto vendita.
  - *reorder Level:* livello di riordino dello specifico prodotto all'interno del punto vendita considerato;
  - *quantità posseduta:* indica la quantità disponibile dello specifico prodotto all'interno del punto vendita considerato ;
  - *quantità riordino:* indica la quantità di riordino in condizioni normale dello specifico prodotto all'interno del punto vendita considerato.
- SERVE: relazione tra Fornitore e Punto Vendita. Indica i punti vendita forniti da ogni fornitore.
  - *feedback:* indica un valore di qualità delle transazioni tra fornitore e punto vendita;
  - *numero transazioni:* numero di transazioni effettuate tra fornitore e punto vendita;
  - *costo:* parametro di costo relativo alle transazioni.
- LOCALIZZATO: relazione tra Fornitore e Granularità Locazione e tra Punto Vendita e Granularità Locazione. Individua la locazione (espressa in maniera granulare) di ogni entità.
- ASSOCIATE: relazione tra Coordinate e Granularità Locazione. Essendo le coordinate geografiche un valore doppio, formato da latitudine e longitudine è necessario creare una tabella per l'espressione di questo tipo di granularità.

- HA: relazione tra Prodotto e Tipologia Forniture. Individua le tipologie di prodotti posseduti da un dato fornitore.
- FORNITO: relazione tra Fornitore e Tipologia Forniture. Individua le tipologie di prodotti che un fornitore è in grado di fornire.

Nel modello logico si crea una tabella per ogni entità e per ogni relazione molti a molti (con l'aggiunta degli attributi sulla relazione) e un nuovo attributo per ogni chiave esterna generata da una relazione uno a molti. Lo schema logico è mostrato in figura 5.2.

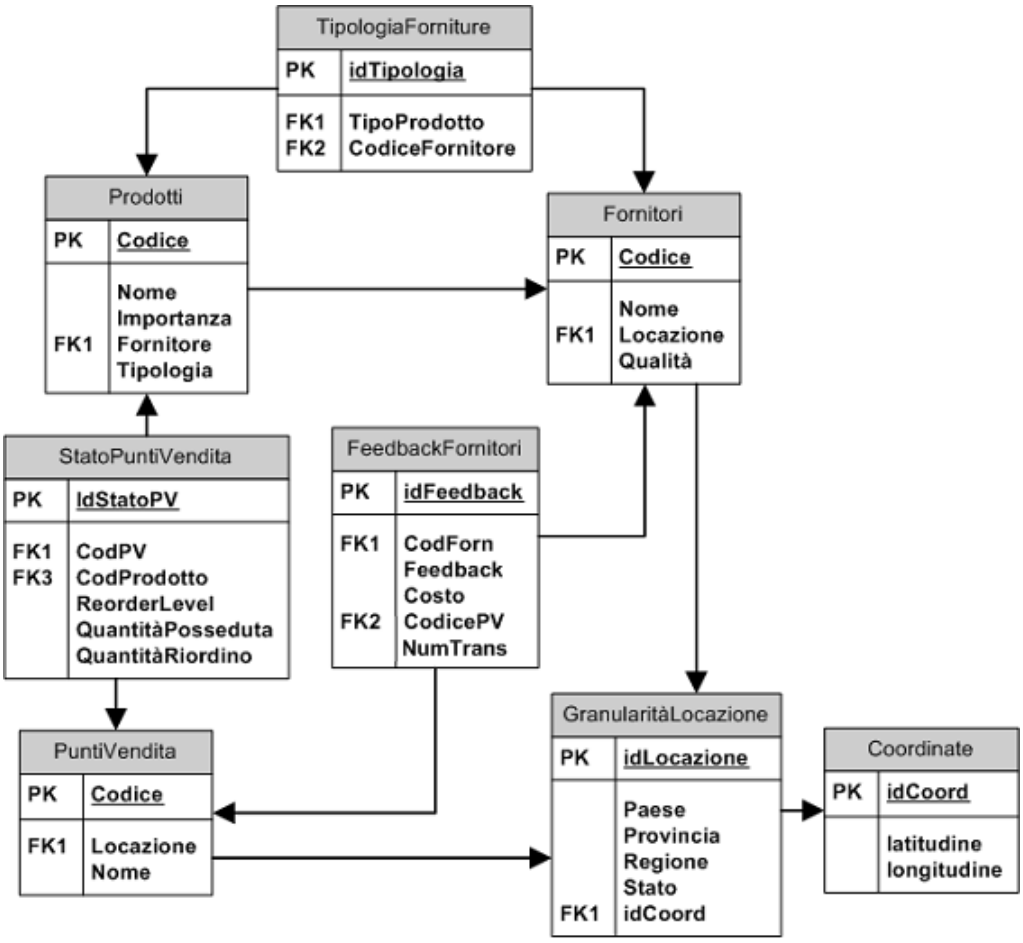

Figura 5.2: Diagramma ER dell'applicazione: modello logico

5.1.2 Realizzazione

La base di dati è realizzata utilizzando MySql 5.1.49, e il tool grafico MySqlAdministrator 1.2.17. Viene implementata una tabella per ogni entità del modello logico e con i campi indicati.

5.2 Realizzazione dei Web Service

I web service sono realizzati utilizzando un approccio bottom up a partire dalla classe Java. Viene utilizzato come ambiente di sviluppo Eclipse Helios arricchito dei driver WTP(Web Tools Platform Project) e delle funzionalità di Axis2 versione 1.5.4. Il server usato è Apache Tomcat 7.0.11, che lavora alla porta 8081.

Per quanto riguarda la realizzazione dei processi relativi allo scenario di riferimento vengono realizzati i seguenti servizi:

- *InvioOrdine(int pv)*: si occupa di verificare quali prodotti del punto vendita hanno raggiunto il reorder level, effettuando una connessione al database e una query allo scopo di estrarre e confrontare i campi specifici. Per i prodotti che hanno raggiunto il reorder level simula l'invio di un ordine del quantitativo preassegnato.

- *RaccoltaFeedback(int pv)*: viene richiesto all'utente di inserire alcuni dati (numero prodotti non conformi, giorni di ritardo) allo scopo di calcolare mediante opportuni indici di demerito la qualità della transazione con il fornitore preassegnato. Una volta misurato l'indice di qualità della transazione sarà pesato secondo il numero di transazioni effettuate con quello presente nel database ed aggiornato il valore.

- *SelezioneFornitore(int pv, String importanza, int codiceProdotto)*: è un servizio astratto utilizzato per l'invocazione dinamica dei servizi concreti:

  - *SelezioneFornitoreA(int pv, String importanza, int codiceProdotto)*: effettua una scelta molto dettagliata del fornitore migliore per il prodotto. Vengono selezionati solo i fornitori la cui qualità è superiore a 8, il cui feedback è maggiore di 90 e che abbiano la minima distanza (calcolata alla granularità più fine, cioè le coordinate) con il punto vendita. E'considerato anche il parametro di costo ma gli si dà un peso minimo dello 0,1 sul totale.
  - *SelezioneFornitoreB(int pv, String importanza, int codiceProdotto)*: effettua una scelta abbastanza dettagliata del fornitore migliore per il prodotto. Vengono rilassati i vincoli precedenti, selezionando i fornitori la cui qualità è superiore a 7, il cui feedback è maggiore di 85 e il confronto sulla locazione si basa unicamente sul fatto che i fornitori si trovino nella stessa provincia. Il parametro di costo è considerato con un peso dello 0,2 sul totale.
  - *SelezioneFornitoreC(int pv, String importanza, int codiceProdotto)*: effettua una scelta poco dettagliata del fornitore migliore per il prodotto. Si selezionano i fornitori coni qualità superiore alla sufficienza, il cui feedback è maggiore di 80 e il confronto sulla locazione si basa unicamente sul fatto che i fornitori si trovino nella stessa regione. Il parametro di costo è considerato fondamentale in questo caso, e gli si attribuisce un peso dello 0,4 sul totale.

- *AnalisiWeka(String condAtm, int vacanze, String stagione, String locazione)*: si occupa della creazione del modello dall'applicazione sulla base dell'algoritmo di classificazione NBtree e di un dataset di dati storici e effettua una previsione sull'andamento della domanda a seconda dei parametri contestuali inseriti.

- *AggiornamentoMagazzino(String trendDomanda, String locazione)*: è un servizio astratto utilizzato per l'invocazione dinamica dei servizi concreti:

- *AggiornamentoMagazzinoInCalo(String trendDomanda, String locazione):* effettua un aggiornamento della granularità di monitoraggio salvata nel database nei punti vendita con la locazione inserita, aumentando il periodo di monitoraggio del punto vendita. Inoltre vengono diminuiti del 20% anche il reorder level e la quantità di riordino.
- *AggiornamentoMagazzinoCostante(String trendDomanda, String locazione)*: non effettua alcun aggiornamento nel database.
- *AggiornamentoMagazzinoInCrescita(String trendDomanda, String locazione)*: effettua un aggiornamento della granularità di monitoring salvata nel database nei punti vendita con la locazione inserita, diminuendo il periodo di monitoraggio del punto vendita. Inoltre vengono aumentati del 20% anche il reorder level e la quantità di riordino.

5.3 Invocazione dinamica dei servizi

Per invocare dinamicamente i servizi all'interno dei processi è stata utilizzata la piattaforma open source ESB (Enterprise Service Bus) versione 3.0.1, basata su Apache Synapse e fornita dal consorzio WSO2. Viene configurato un proxy service, ossia un componente software posto come intermediario tra il browser dell'utente e il server, avente funzioni di filtro. E' configurato allo scopo di mediare le richieste dell'utente: l'utente accede al servizio mediante l'invocazione di uno dei metodi forniti dall'interfaccia del servizio astratto, e sarà il proxy ESB a scegliere, a seconda delle condizioni contestuali rilevate, quale servizio concreto invocare, automaticamente ed in maniera del tutto trasparente all'utente che si connette all'applicazione.

A questo scopo vengono implementati e pubblicati su localhost alla porta 8081 alcuni web service, con lo stesso XML-schema:

- un servizio astratto che non svolga alcuna operazione specifica, ma fornisca un'interfaccia per un gruppo di servizi concreti omogenei con

caratteristiche, funzionalità e struttura molto simili tra loro. L'utente invocherà sempre un servizio astratto e si aspetterà una risposta da esso;

- alcuni servizi concreti che si occupino dello svolgimento vero e proprio delle operazioni. Sono raggruppati tra loro secondo gruppi omogenei, ognuno dei quali è descritto da un'interfaccia di riferimento.

Per accedere alla piattaforma ESB ci si connette all'indirizzo http://localhost:9443/carbon/wso2 e si configura un proxy service che a fronte di una richiesta astratta dell'utente a seconda di parametri contestuali rilevati a runtime invochi il servizio concreto più idoneo.

Per quanto riguarda lo scenario della gestione delle scorte, l'invocazione dinamica dei servizi si realizza due volte nell'applicazione:

- nella selezione granulare del fornitore a seconda dell'importanza del prodotto, mediante l'invocazione del servizio astratto *selezioneFornitore* e in riferimento ai servizi concreti *selezioneFornitoreA, selezioneFornitoreB, selezioneFornitoreC*;

- nell'aggiornamento della granularità di monitoraggio del magazzino a seconda del trend della domanda, mediante l'invocazione del servizio astratto *aggiornamentoMagazzino* e in riferimento ai servizi concreti *aggiornamentoMagazzinoCalo, aggiornamentoMagazzinoCostante, aggiornamentoMagazzinoCrescita*.

La struttura del proxy configurato è pressochè identica nei due casi e viene descritta in seguito cercando di generalizzare il più possibile e fornendo esempi concreti dove necessario riguardo il processo di selezione granulare del fornitore. Il meccanismo di invocazione dinamica descritto può essere facilmente utilizzato, con delle piccole modifiche, per ogni caso di studio a cui ci si riferisca.

Vengono definite due sequenze, una in ingresso e una in uscita al proxy, come mostrato in figura 5.3, che indichino quale deve essere il comportamento

del proxy nei confronti della richiesta astratta ricevuta dall'utente e della risposta concreta ottenuta dall'invocazione del servizio concreto:

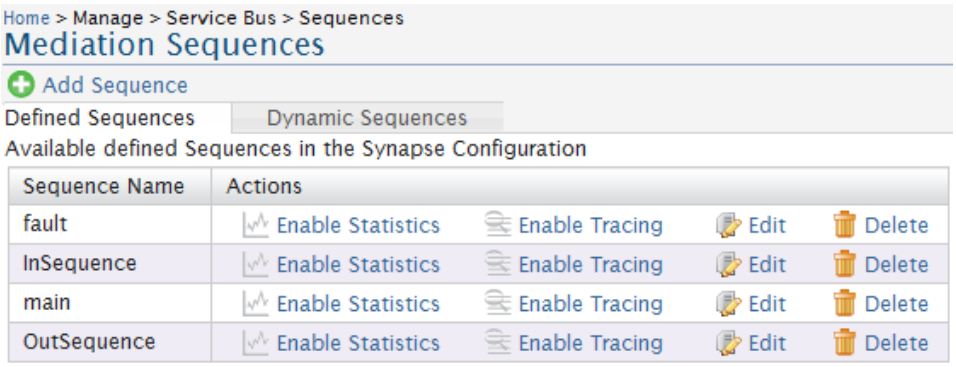

Figura 5.3: Sequenze del Proxy service

La sequenza in ingresso ha il compito di ricevere la richiesta astratta dell'utente e indirizzarla, a seconda dei parametri contestuali rilevati a runtime, verso il servizio concreto più idoneo effettuando delle trasformazioni sulla richiesta in modo da renderla compatibile con quella del servizio concreto scelto. Sarà quindi realizzata come mostrato in figura 5.4.

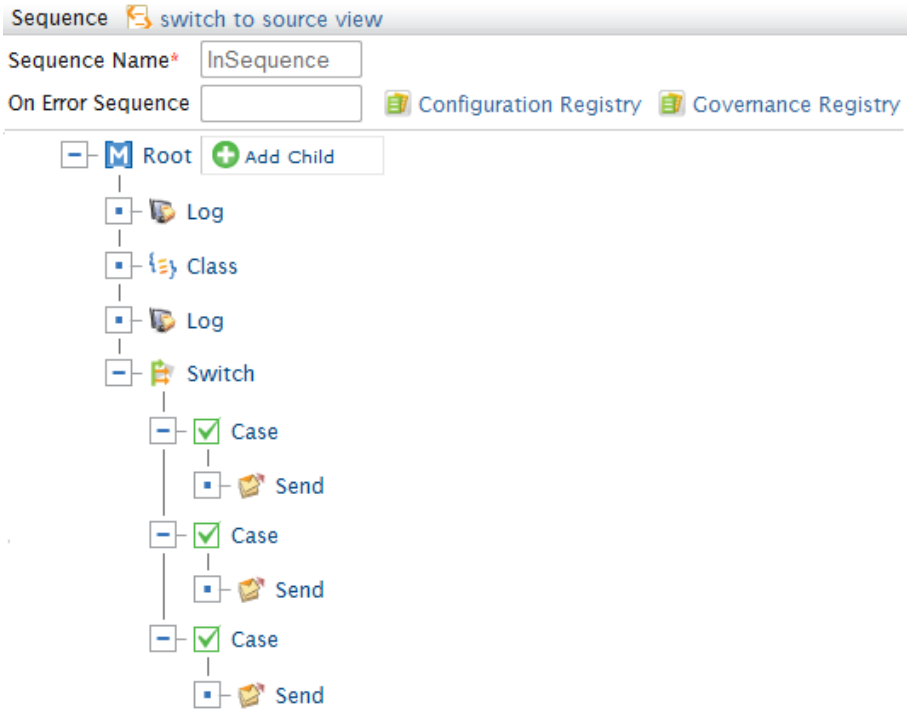

Figura 5.4: Sequenza in ingresso

I *log mediator* hanno il solo scopo di mostrare nel prompt il messaggio che passa attraverso il proxy.

ll *class mediator* è una classe Java realizzata in Eclipse mediante l'aggiunta della libreria synapse-core1-1-1.jar ed inserita nell'ESB come file jar. Si occupa di decidere, sulla base di parametri contestuali rilevati a runtime, quale servizio concreto invocare, settando il campo SoapAction a seconda dei parametri contestuali, con il metodo del web service idoneo da invocare.

A scopo esemplificativo viene mostrata in figura 5.5 la classe realizzata per il processo di profilazione granulare del fornitore. Viene rilevata a runtime, mediante il parsing della richiesta dell'utente, l'importanza del prodotto (A, B, C) e settato il campo SoapAction con il metodo del servizio concreto da invocare (selezioneFornitoreA, selezioneFornitoreB, selezioneFornitoreC).

```
package mediatore;

import javax.xml.namespace.QName;
import org.apache.axiom.om.OMElement;
import org.apache.synapse.ManagedLifecycle;
import org.apache.synapse.MessageContext;
import org.apache.synapse.core.SynapseEnvironment;
import org.apache.synapse.mediators.AbstractMediator;

public class Mediatore extends AbstractMediator implements ManagedLifecycle{
public boolean mediate(MessageContext synCtx){
 System.out.println("mediate");
 String ns="http://fornitori";
 OMElement servizio=synCtx.getEnvelope().getBody().getFirstElement();
 OMElement importanza=servizio.getFirstChildWithName(new QName(ns,"importanza"));
 String imp=importanza.getText();
 System.out.println(imp);
 if (imp.equals("a")) {
     synCtx.setSoapAction("selezioneFornitoreA");
     System.out.println("selezioneA"); }
 if (imp.equals("b")){
     synCtx.setSoapAction("selezioneFornitoreB");
     System.out.println("selezioneB"); }
 if (imp.equals("c")){
     synCtx.setSoapAction("selezioneFornitoreC");
     System.out.println("selezioneC"); }
 return true;  }
public void destroy(){ }
public void init(SynapseEnvironment arg0) {
    // TODO Auto-generated method stub  }}
```

Figura 5.5: Class Mediator: nel caso di profilazione granulare del fornitore

Nel caso dell'aggiornamento del magazzino quello che cambierà nella classe sono unicamente i parametri contestuali rilevanti e il settaggio del campo SoapAction.

Lo *switch mediator* rileva come è stato settato il campo SoapAction ed è associato ad un *send mediator* che si occupa dell'invio della richiesta all'endpoint concreto scelto. La configurazione dello switch mediator è mostrata in figura 5.6 e in figura 5.7, in riferimento allo scenario di profilazione granulare del fornitore.

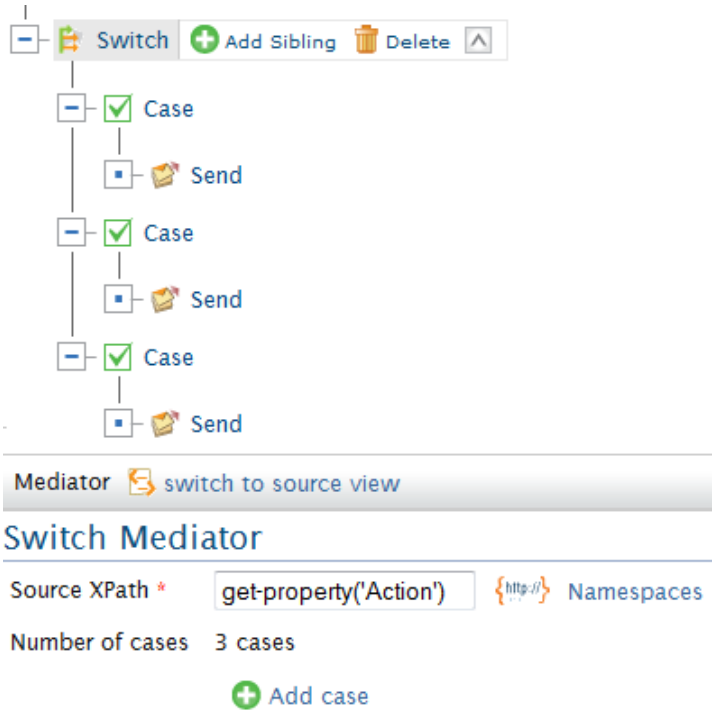

Figura 5.6: Switch Mediator: nel caso di profilazione granulare del fornitore

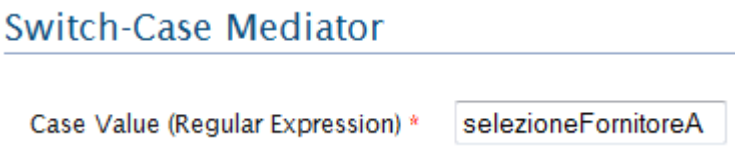

Figura 5.7: Switch Mediator: Caso A

La sequenza in uscita si occupa di inviare la risposta all'utente. La risposta ricevuta dal servizio concreto, avrà una struttura specifica a seconda del servizio invocato. L'utente deve ricevere una risposta avente la forma del servizio astratto invocato.

Per questo motivo, la sequenza in uscita si occupa di effettuare modifiche alla risposta ottenuta dal servizio concreto in modo da renderla analoga a quella che si otterrebbe eseguendo il servizio astratto.
La sequenza in uscita è descritta in figura 5.8.

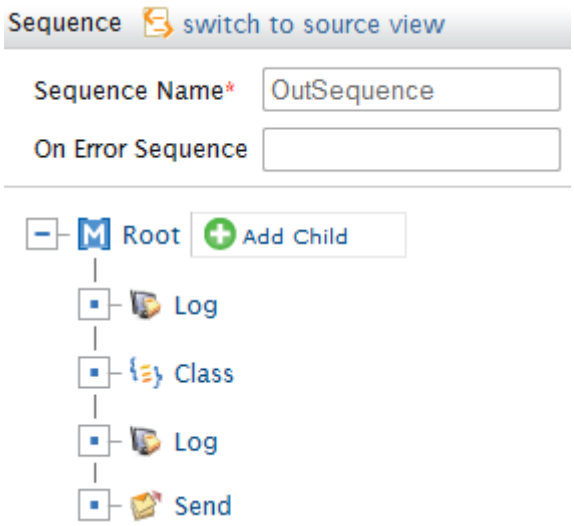

Figura 5.8: Sequenza in uscita

Essa è composta unicamente da un *class mediator*, che si occupi della modifica della risposta e da un *send mediator* vuoto per l'invio della risposta al client.

La struttura del class mediator, in riferimento allo scenario, è mostrata in figura 5.9, e si occupa di modificare la risposta ottenuta dal servizio concreto in modo da renderla conforme al servizio astratto richiesto dall'utente.

```
package mediatoreOut;

import org.apache.synapse.ManagedLifecycle;
import org.apache.synapse.MessageContext;
import org.apache.synapse.core.SynapseEnvironment;
import org.apache.synapse.mediators.AbstractMediator;

public class MediatoreOut extends AbstractMediator implements ManagedLifecycle{
public boolean mediate(MessageContext synCtx){
 synCtx.getEnvelope().getBody().getFirstElement().setLocalName("selezioneFornitoreResponse");
 return true;
 }
public void destroy(){
   }
public void init(SynapseEnvironment arg0) {
       // TODO Auto-generated method stub
   }}
```

Figura 5.9: Class mediator in uscita

A questo punto viene configurato un Custom Proxy Service che abbia come sequenze di ingresso e di uscita quelle definite e come WSDL della richiesta quello del servizio astratto di riferimento.

5.4 Proattività: analisi della domanda con WEKA

Nel framework, come descritto nei capitoli precedenti, viene fornito un approccio proattivo per la realizzazione dell'adattività, affiancato al tipico approccio reattivo. Essendo l'adattamento un'operazione piuttosto lunga, è preferibile prevedere eventuali anomalie prima che si verifichino in modo da cominciare da subito ad attuare operazioni correttive evitando l'eccessiva perdita di tempo che si avrebbe reagendo unicamente al verificarsi di cambiamenti.

L'adattività proattiva è basata su stime riguardanti il comportamento dell'applicazione in determinate situazioni ed è quindi necessario fornire un meccanismo per prevedere in maniera verosimile l'andamento del sistema. Nel framework realizzato saranno utilizzate tecniche di machine learning, utilizzando algoritmi di classificazione che permettano all'applicazione di pre-

vedere il proprio comportamento sulla base dell'analisi di serie di dati storici relative ad esecuzioni precedenti del programma.

Viene descritta in seguito questa metodologia, in riferimento allo specifico scenario di riferimento relativo alla previsione dell'andamento della domanda, in modo da aggiornare parametri quali la frequenza di monitoraggio del magazzino, il reorder level e la quantità di riordino per evitare il verificarsi di situazioni di stock-out e stock-over.

Come tool di machine learning viene utilizzato Weka 3.6.4, uno strumento open source di data mining, realizzato dall'università di Waikato [10].

5.4.1 Dataset

Nel caso di studio, relativo alla previsione della domanda all'interno di un punto vendita, si definisce un attributo per ogni parametro contestuale da cui dipende la domanda. Vengono individuati i seguenti attributi:

- condizioni atmosferiche;
- presenza di vacanze;
- stagione;
- locazione.

Il dataset viene salvato in un file con uno specifico formato .arff ed è composto da una prima sezione che descriva la struttura del file, quali sono gli attributi da considerare, la classe da stimare e i valori che possono assumere:

```
@relation domanda

@attribute condAtm {sole,nuvole,pioggia,neve}
@attribute vacanze numeric real
@attribute stagione {autunno,inverno,primavera,estate}
@attribute locazione {mare,montagna,città}
@attribute trendDomanda {calo,costante,crescita}
```

Nella sezione @data vengono poi salvate le specifiche istanze che descrivano i valori dei parametri rilevati in invocazioni precedenti dell'applicazione in relazione alla classe a cui apparteneva il parametro da stimare.

Il dataset considerato nello scenario di riferimento è composto da 76 istanze classificate, nel seguente formato:

```
@data
neve,3,inverno,montagna,crescita
pioggia,5.1,primavera,mare,calo
                        ....
sole,6.2,estate,città,calo
nuvole,6,estate,montagna,costante
```

Alcune istanze del dataset utilizzato per la descrizione dell'andamento della domanda in relazione ai parametri contestuali sono mostrate in forma tabellare in figura 5.10.

Relation: domanda

No.	condAtm Nominal	vacanze Numeric	stagione Nominal	locazione Nominal	**trendDomanda** Nominal
1	sole	3.0	autunno	mare	crescita
2	sole	3.44	autunno	montagna	crescita
3	sole	5.0	autunno	mare	costante
4	sole	7.75	autunno	montagna	costante
5	nuvole	3.1	autunno	mare	costante
6	nuvole	3.8	autunno	città	costante
7	nuvole	7.5	autunno	mare	calo
8	pioggia	3.75	autunno	montagna	calo
9	pioggia	6.2	autunno	mare	calo
10	pioggia	6.0	autunno	montagna	calo
11	neve	3.44	autunno	montagna	crescita
12	neve	3.0	autunno	città	costante
13	neve	5.1	autunno	mare	calo
14	neve	7.75	autunno	montagna	crescita
15	sole	3.1	inverno	mare	costante

Figura 5.10: Dataset arff

5.4.2 Algoritmi di classificazione

Vengono applicati al dataset considerato tutti gli algoritmi di classificazione descritti nel capitolo 1. Vengono mostrate in seguito le funzioni Weka che permettano di effettuare tali classificazioni:

Alberi decisionali: algoritmo J48

Il sistema WEKA implementa diversi algoritmi per la classificazione utilizzando DT. Tra questi è stato scelto l'algoritmo *J48* che contiene opzioni di pruning e che gestisce sia dati a valori numerici che a valori nominali.

Reti Bayesiane: approccio Naive Bayes

Con WEKA si possono applicare vari algoritmi presenti nella scheda Classify per la classificazione mediante BN, ad esempio l'algoritmo BayesNet in Bayes (il cui percorso è WEKA.classifiers.bayes.BayesNet), in cui si possono selezionare delle opzioni che permettono la scelta dell'algoritmo per la costruzione della topologia della rete e per il calcolo delle tabelle di probabilità condizionate associate ai nodi della rete. Per poter applicare l'algoritmo BayesNet, è necessario raffinare la preparazione dei dati per ottenere un dataset adeguato allo scopo. Infatti i dati devono essere discretizzati. Il filtro che WEKA utilizza si chiama Discretize e si può recuperare tra i filtri unsupervized.

Vengono considerate le reti Bayesiane di tipo Naive, con WEKA si applica l'algoritmo NaiveBayes (il cui percorso è WEKA.classifiers.bayes.NaiveBayes) presente anch'esso nella scheda Classify per la classificazione mediante BN.

NBTree

Con WEKA viene applicato l'algorimto NBTree presente nella scheda Classify per la classificazione mediante alberi di decisione.

5.4.3 Confronto tra i risultati ottenuti

E' necessario scegliere l'algoritmo più idoneo per la modellizzazione dell'andamento della domanda in riferimento al dataset considerato. Per questo motivo vengono applicati al dataset di riferimento tutti gli algoritmi di classificazione descritti in precedenza ed analizzate e confrontate le misure di performance ottenute (descritte nel capitolo 1). Verrà scelto, in fase di progettazione, l'algoritmo che otterà performance migliori.

Di seguito sono mostrate le performance ottenute dall'applicazione degli algoritmi di classificazione al dataset utilizzando WEKA:

Albero di decisione J48

Numero di istanze classificate correttamente

```
Correctly Classified Instances          61               80.2632 %
Incorrectly Classified Instances        15               19.7368 %
```

Dimensione dell'albero

```
Number of Leaves  : 20
Size of the tree : 28
```

Matrice di confusione

```
=== Confusion Matrix ===
 a  b  c   <-- classified as
25  3  0 |  a = calo
 3 22  3 |  b = costante
 2  4 14 |  c = crescita
```

Precision, Recall, AUC

	TP Rate	FP Rate	Precision	Recall	ROC Area	Class
	0.893	0.104	0.833	0.893	0.927	calo
	0.786	0.146	0.759	0.786	0.869	costante
	0.7	0.054	0.824	0.7	0.841	crescita
Weighted Avg.	0.803	0.106	0.803	0.801	0.883	

Curva ROC

Domanda in calo

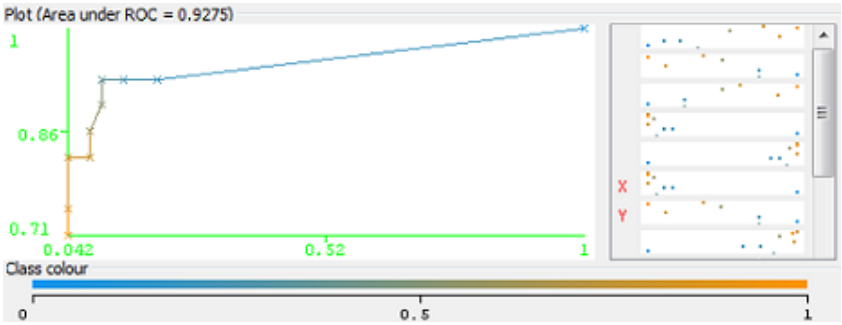

Figura 5.11: Curva ROC J48: Domanda in calo

Domanda costante

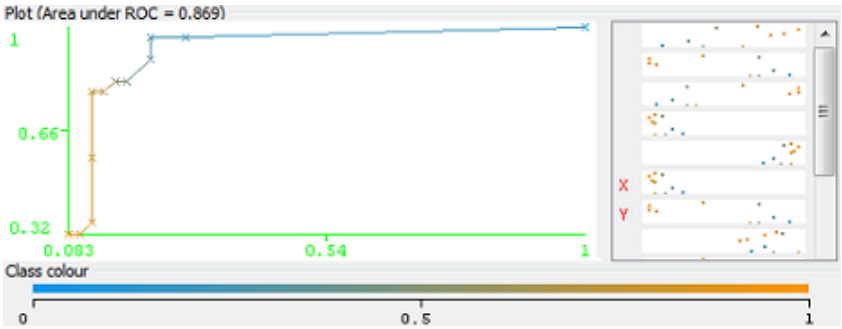

Figura 5.12: Curva ROC J48: Domanda costante

Domanda in crescita

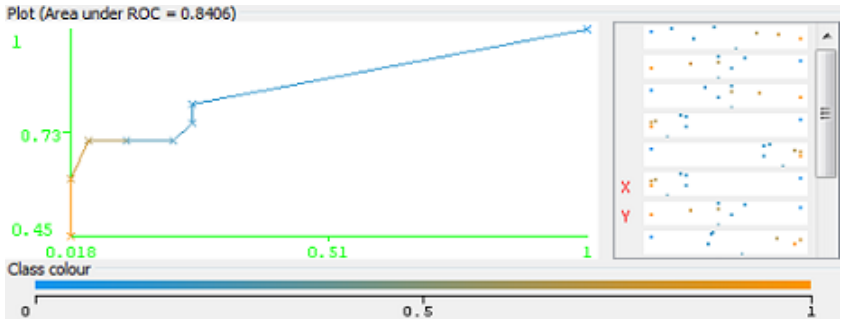

Figura 5.13: Curva ROC J48: Domanda in crescita

Rete Bayesiana Naive Bayes

Numero di istanze classificate correttamente

Correctly Classified Instances	40	52.6316 %
Incorrectly Classified Instances	36	47.3684 %

Matrice di confusione

```
=== Confusion Matrix ===
 a  b  c   <-- classified as
 9  7  2 |  a = calo
 8 14  6 |  b = costante
 5  8  7 |  c = crescita
```

Precision, Recall, AUC

	TP Rate	FP Rate	Precision	Recall	ROC Area	Class
	0.679	0.271	0.594	0.679	0.761	calo
	0.5	0.313	0.483	0.5	0.659	costante
	0.35	0.143	0.467	0.35	0.657	crescita
Weighted Avg.	0.526	0.253	0.519	0.52	0.696	

Curva ROC

Domanda in calo

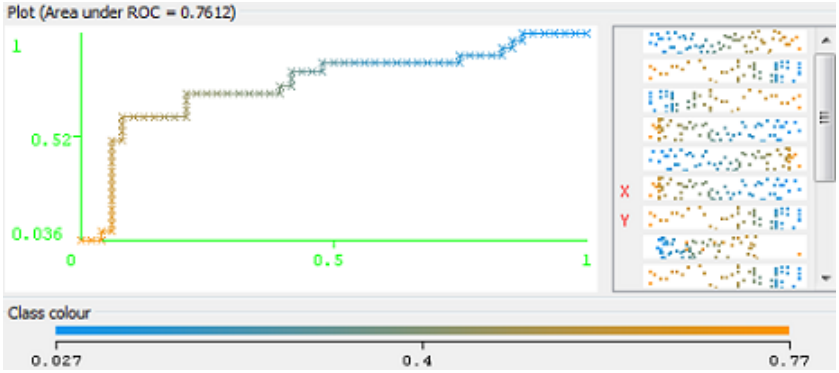

Figura 5.14: Curva ROC Naive Bayes: Domanda in calo

Domanda costante

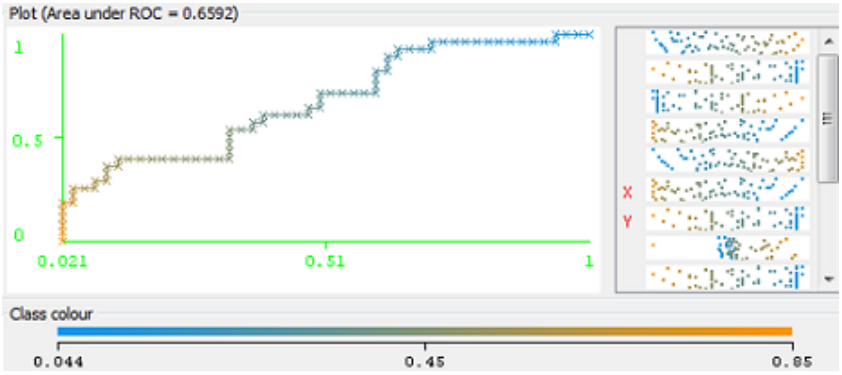

Figura 5.15: Curva ROC Naive Bayes: Domanda costante

Domanda in crescita

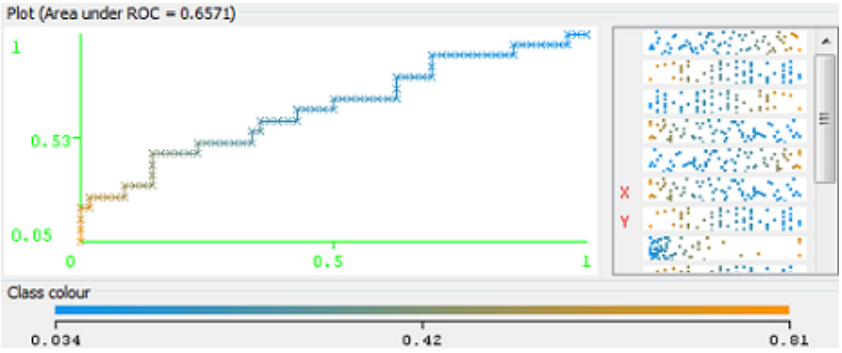

Figura 5.16: Curva ROC Naive Bayes: Domanda in crescita

NBTree

Numero di istanze classificate correttamente

```
Correctly Classified Instances          62               81.5789 %
Incorrectly Classified Instances        14               18.4211 %
```

Dimensione dell'albero

```
Number of Leaves  :  6
Size of the tree :  8
```

Matrice di confusione

```
=== Confusion Matrix ===
 a  b  c   <-- classified as
24  4  0 |  a = calo
 2 25  1 |  b = costante
 2  5 13 |  c = crescita
```

Precision, Recall, AUC

	TP Rate	FP Rate	Precision	Recall	ROC Area	Class
	0.857	0.083	0.857	0.857	0.934	calo
	0.893	0.188	0.735	0.893	0.88	costante
	0.65	0.018	0.929	0.65	0.914	crescita
Weighted Avg.	0.816	0.104	0.831	0.814	0.909	

Curva ROC

Domanda in calo

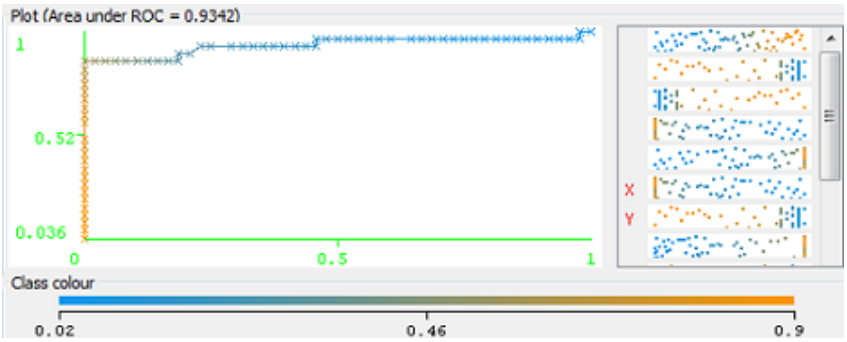

Figura 5.17: Curva ROC NBTree: Domanda in calo

Domanda costante

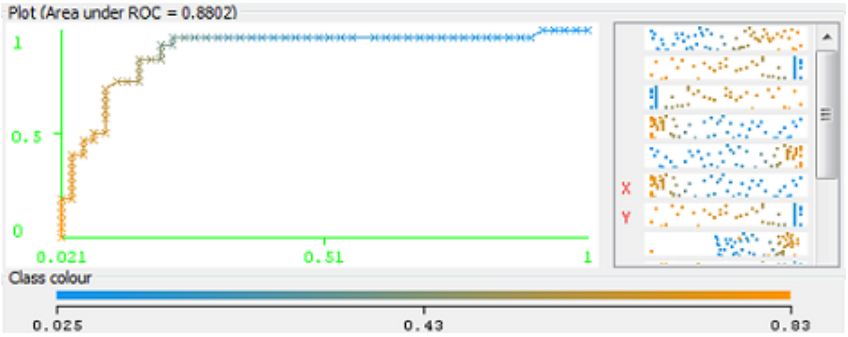

Figura 5.18: Curva ROC NBTree: Domanda costante

Domanda in crescita

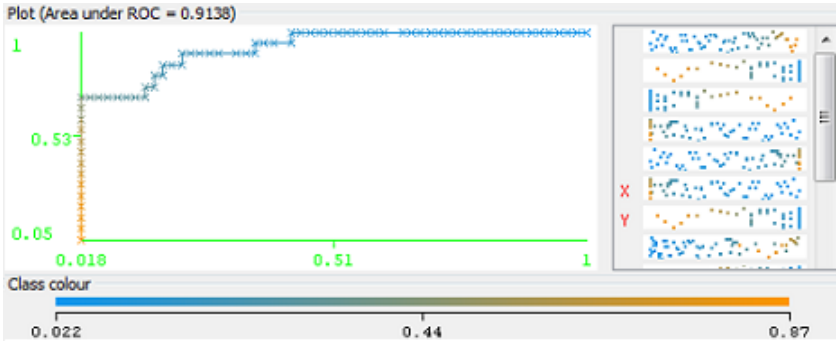

Figura 5.19: Curva ROC NBTree: Domanda in crescita

5.4.4 Scelta dell'algoritmo e classe Java per l'invocazione

Confrontando le misure delle performance ottenute mediante l'applicazione dei tre algoritmi di classificazione scelti e mostrate nella sottosezione precedente si nota che, con il dataset considerato, l'algoritmo più idoneo per la classificazione dell'andamento della domanda è l'NBTree, in quanto fornisce performance migliori:

- la percentuale di istanze classificate correttamente è la più alta;
- dimensione del modello minore;

- migliore in termini di precisione e recall;
- la curva ROC è quella che più si avvicina all'angolo in alto a sinistra del quadrante.

L'unico svantaggio rispetto agli altri metodi è che il tempo per il training della rete è superiore.

Viene scelto quindi l'algoritmo NBTree per la classificazione e realizzata una classe Java che ne permetta l'invocazione. Per l'utilizzo di WEKA in Java è necessario importare il package weka.jar presente nella cartella di Weka.

I metodi utilizzati per l'utilizzo di WEKA all'interno di una classe Java sono descritti in [24].

Creazione ArffLoader per caricare il dataset e determinazione dell'attributo classe

```
import weka.core.converters.ArffLoader;

//Load data
ArffLoader loader = new ArffLoader();
loader.setFile(new File("/Program Files/Weka-3-6/data/domanda5.arff"));
Instances structure = loader.getStructure();
Instances dataSet = loader.getDataSet();

//Setting class attribute
dataSet.setClassIndex(structure.numAttributes() - 1);
```

Discretizzazione del dataset (permette di ottenere risultati migliori)

```
import weka.filters.Filter;
import weka.filters.supervised.attribute.Discretize;

Discretize m_DiscretizeFilter = new Discretize();
m_DiscretizeFilter.setInputFormat(dataSet);
dataSet = Filter.useFilter(dataSet, m_DiscretizeFilter);
```

Creazione di un classificatore NBTree, che classifichi il dataset caricato

```
import weka.classifiers.trees.NBTree;

NBTree NBtree=new NBTree();
NBtree.buildClassifier(dataSet);
```

Cross Validation

```
import weka.classifiers.Evaluation;

 Evaluation eval = new Evaluation(dataSet);
 eval.crossValidateModel(NBtree, dataSet, 10, new Random(1));
```

Nella classe Java realizzata viene effettuata anche l'operazione di classificazione di una nuova istanza inserita dall'utente (ciò non è previsto dalla sola interfaccia grafica di WEKA). Quando l'utente inserirà i valori attuali di condizioni atmosferiche, presenza di vacanze, stagione e locazione verrà effettuata una previsione della classe di appartenenza dell'istanza, utilizzando come modello la rete addestrata precedentemente. Questo viene fatto scrivendo su un file con la stessa struttura del dataset l'istanza da classificare e invocando su di esso il metodo classifyInstance:

Scrittura su file

```
public static void scriviFile(String condAtm,double vacanze,
String stagione,String locazione) {
String path = "C:/Users/Ale/Documents/Weka/domanda1.arff";
try {
FileOutputStream istanza = new FileOutputStream(path);
PrintStream scrivi = new PrintStream(istanza);
scrivi.println("@relation domanda1");
scrivi.println("@attribute condAtm {sole,nuvole,pioggia,neve}");
scrivi.println("@attribute vacanze numeric real");
scrivi.println("@attribute stagione {autunno,inverno,primavera,estate}");
scrivi.println("@attribute locazione {mare,montagna,città}");
scrivi.println("@attribute trendDomanda {calo, costante, crescita}");
```

```
 scrivi.println("@data");
 scrivi.println(condAtm+","+vacanze+","+stagione+","+locazione+",calo");
 scrivi.flush();
 scrivi.close();
 }
 catch(IOException e) {
 e.printStackTrace(); }
}
```

Caricamento dataset per la classificazione dell'istanza, set della classe e discretizzazione

```
//Load data
ArffLoader loader1 = new ArffLoader();
loader1.setFile(new File("C:/Users/Weka/domanda1.arff"));
Instances dataSet1 = loader1.getDataSet();

//Setting class attribute
dataSet1.setClassIndex(structure.numAttributes() - 1);

//Discretizzazione dataset
Discretize m_DiscretizeFilter1 = new Discretize();
m_DiscretizeFilter1.setInputFormat(dataSet1);
dataSet1 = Filter.useFilter(dataSet1, m_DiscretizeFilter1);
```

Classificazione istanza

```
for (int i = 0; i < dataSet1.numInstances(); i++) {
 double pred = NBtree.classifyInstance(dataSet1.instance(i));
 System.out.println("predizione: "+ dataSet1.classAttribute().value((int) pred));
 }
```

5.5 Orchestrazione dei web service

Per la progettazione e lo sviluppo di applicazioni complesse è necessario coordinare diversi web service per il raggiungimento di un obiettivo comune. Per fare ciò vengono utilizzati dei meccanismi di gestione dei workflow.

Nel framework realizzato per l'orchestrazione dei servizi viene utilizzato BPEL (Business Process Execution Language), un linguaggio basato sull'XML costruito per descrivere formalmente i processi commerciali ed industriali in modo da permettere una suddivisione dei compiti tra attori diversi. Vengono definiti l'ordine di esecuzione dei servizi e le condizioni sotto le quali vengono attivati.

Per la realizzazione dell'applicazione viene utilizzato il plug-in di Eclipse come BPEL designer e il server Apache ODE (Orchestration Director Engine) 1.3.5 alla porta 8080 come BPEL engine.

Nello scenario di gestione delle scorte e allo scopo di fornire un esempio concreto dei concetti teorici introdotti, vengono realizzati due processi, uno per la selezione granulare del fornitore e uno per la previsione proattiva della domanda e l'aggiornamento della granularità di monitoraggio.

5.5.1 Previsione della domanda e aggiornamento della granularità di monitoraggio

Nella figura 5.20 è mostrato il diagramma BPEL del processo di previsione della domanda utilizzando WEKA e di aggiornamento della granularità di monitoraggio.
Viene creato un partner link per ogni servizio da invocare ed orchestrate le operazioni da svolgere mediante adeguati costrutti.

Nel processo di riferimento vengono inseriti dall'utente alcuni dati contestuali attuali e il servizio AnalisiWeka si occupa di prevedere l'andamento della domanda.

Sulla base del risultato viene invocato un secondo metodo AggiornamentoMagazzinoAbstract che si connette ad un ESB ed invoca il servizio concreto idoneo che effettua il corretto aggiornamento della granularità di monitoraggio.

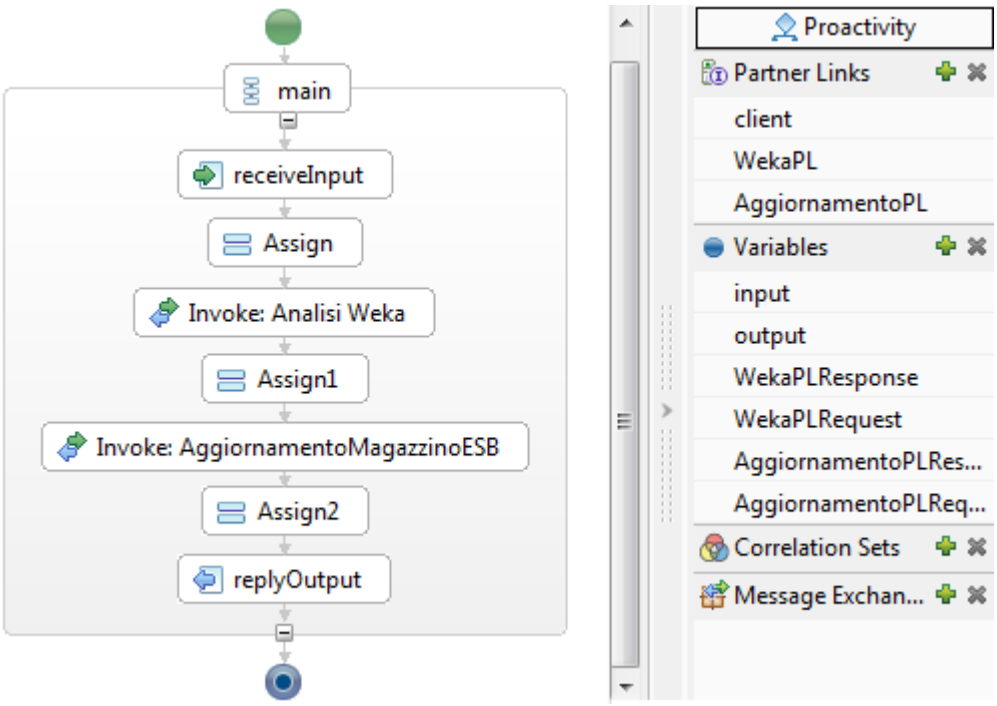

Figura 5.20: BPEL analisi domanda e aggiornamento granularità di monitoraggio

Il WSDL del processo realizzato è descritto nel file ProactivityArtifacts.wsdl ed è descritto in figura 5.21.

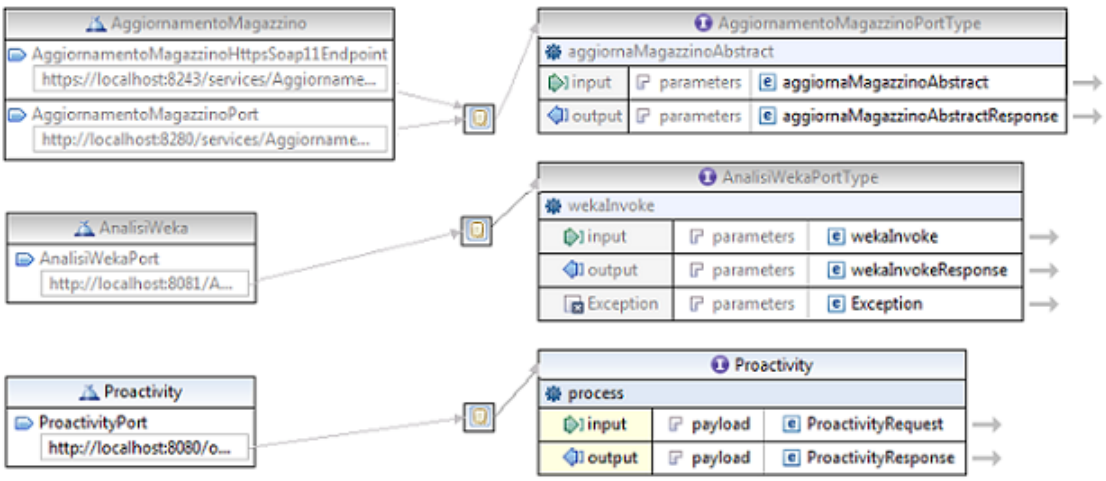

Figura 5.21: WSDL del processo

Una volta realizzato il BPEL e connesso ai rispettivi web service, viene

effettuato il deploy dell'applicazione utilizzando Apache ODE. Viene creato da ODE un file deploy.xml che dev'essere opportunamente configurato, come mostrato in figura 5.22.

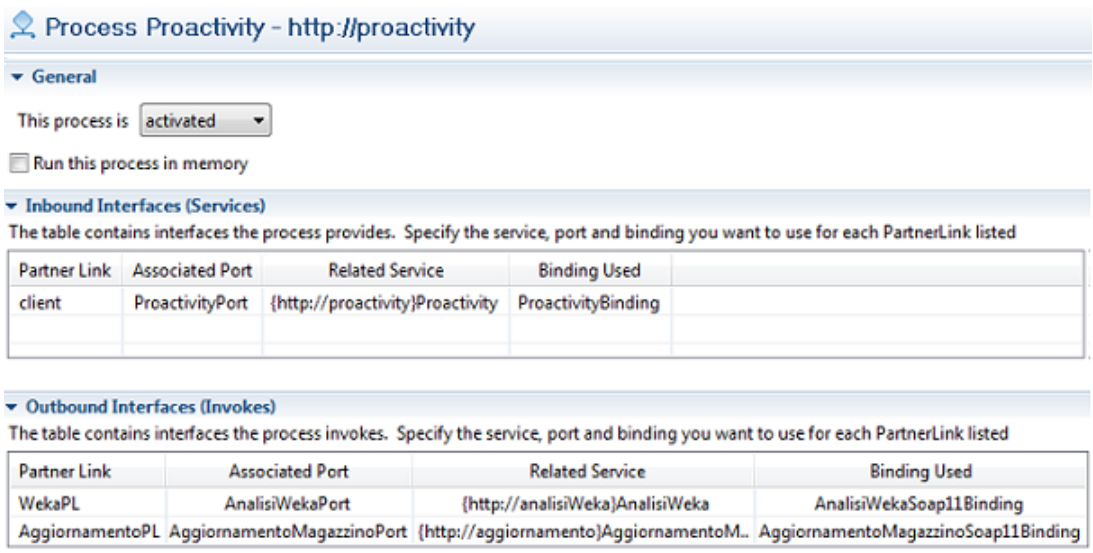

Figura 5.22: File deploy.xml

I web service utilizzati all'interno del pocesso sono pubblicati alla porta 8081 sul server Apache Tomcat, come mostrato in figura 5.23.

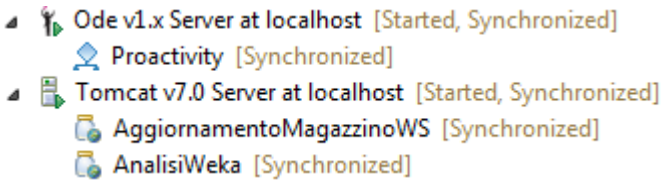

Figura 5.23: Server

Il processo è testato utilizzando il Web Service Explorer di Eclipse. Viene visualizzata un'interfaccia grafica in cui inserire i parametri richiesti dal processo e svolti i web service componenti nell'ordine prestabilito. Nella figura 5.24 è mostrato un esempio di esecuzione del processo:

SOAP Request Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:q0="http://proactivity"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:ProactivityRequest>
      <q0:condAtm>nuvole</q0:condAtm>
      <q0:vacanze>7</q0:vacanze>
      <q0:stagione>primavera</q0:stagione>
      <q0:locazione>montagna</q0:locazione>
    </q0:ProactivityRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ProactivityResponse xmlns="http://proactivity">
      <tns:result xmlns:tns="http://proactivity">Tendenza domanda
      COSTANTE: Nessun aggiornamento nel magazzino </tns:result>
    </ProactivityResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 5.24: Esecuzione del processo

Nella console di Eclipse si nota l'esecuzione sequenziale dei due servizi come previsto:

```
13:52:49,795 INFO  [ExternalService] Response:
<?xml version="1.0" encoding="UTF-8"?>
<message><parameters><wekaInvokeResponse
xmlns="http://analisiWeka"
xmlns:ns="http://analisiWeka"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<return>costante</return></wekaInvokeResponse></parameters></message>

13:53:04,633 INFO  [ExternalService] Response:
<?xml version="1.0" encoding="UTF-8"?>
<message><parameters><aggiornaMagazzinoAbstractResponse
xmlns="http://aggiornamento"
```

```
xmlns:ns="http://aggiornamento"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<return>Tendenza domanda COSTANTE: Nessun aggiornamento nel magazzino</return>
</aggiornaMagazzinoAbstractResponse></parameters></message>
```

5.5.2 Profilazione granulare del fornitore

Il secondo processo sviluppato è quello che si occupa della profilazione granulare del fornitore. Viene inserito dall'utente il punto vendita in cui si trova ed effettuata un'analisi dei prodotti presenti a magazzino, individuando quali tra essi hanno raggiunto il reorder level e devono essere ordinati. Una volta ricevuti i prodotti vengono raccolti dei feedback sulla transazione, sulla base della puntualità della consegna e della conformità dei prodotti. Il feedback raccolto viene pesato con il valore salvato nel database sulla base del numero di transazioni ed aggiornato. Quando il feedback tra fornitore e punto vendita scende sotto un certo livello, si sceglie di selezionare un nuovo fornitore, eseguendo il processo di selezione fornitore. Esso viene eseguito in maniera granulare a seconda dell'importanza del prodotto all'interno del punto vendita, si connette ad un ESB e viene scelto il livello di dettaglio del servizio concreto da invocare. Nella figura 5.25 sono mostrati i partner link che entrano in gioco.

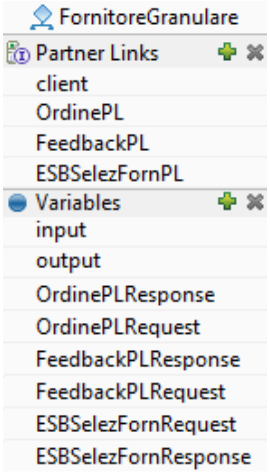

Figura 5.25: BPEL profilazione granulare fornitore, partner link

Nella figura 5.26 è invece mostrata l'orchestrazione dei servizi, effettuata tramite il diagramma BPEL del processo. Il file deploy.xml è configurato analogamente a quello dell'esempio precedente e anche il test dell'applicazione avviene in maniera analoga.

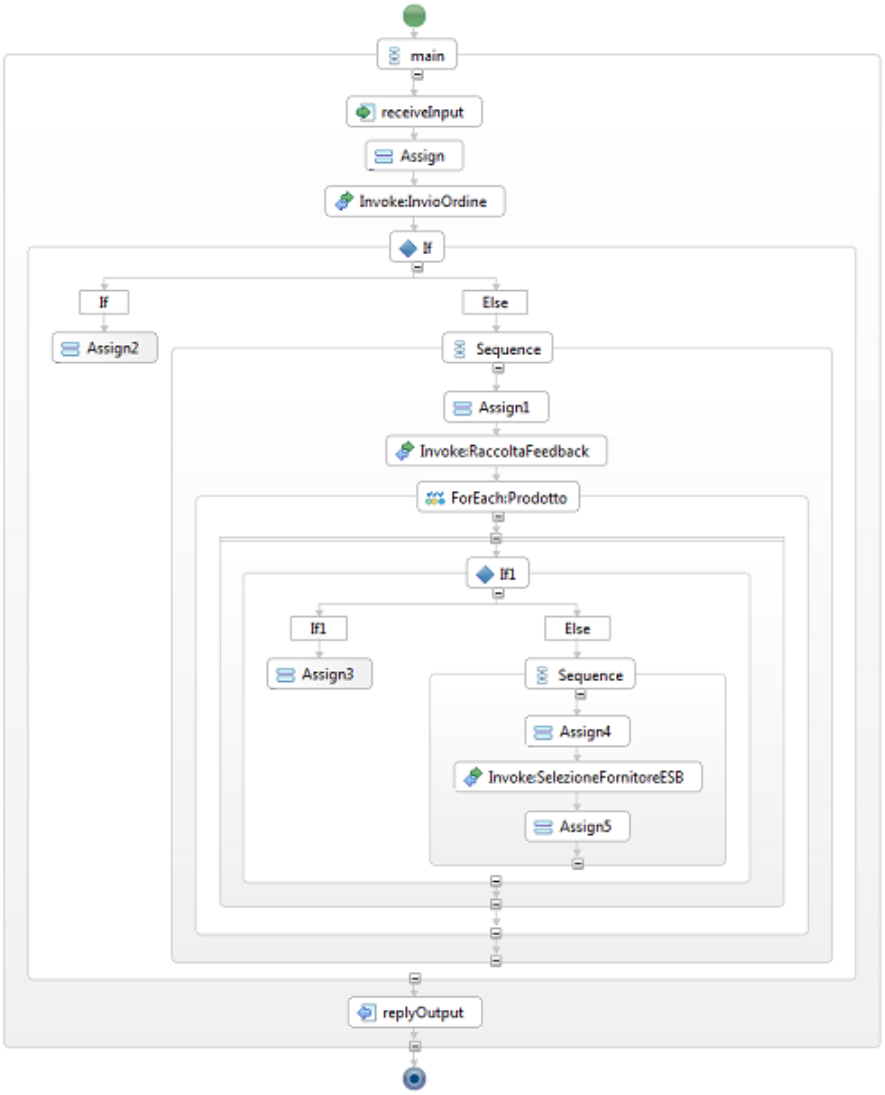

Figura 5.26: BPEL profilazione granulare fornitore

Capitolo 6

Conclusioni e sviluppi futuri

L'obiettivo della tesi è stato quello di descrivere un framework per la realizzazione dell'adattività mediante un approccio proattivo e considerando i dati contestuali correlati con la loro granularità sia informativa che di monitoraggio. Per garantire il corretto svolgimento delle operazioni, evitando inefficienze temporali ed errori a runtime, viene rilevata l'esigenza di lavorare con dati ad uno specifico livello di dettaglio e misurati ad una frequenza idonea in relazione alla specifica istanza in esecuzione dell'applicazione. A questo scopo viene fornito un modello formale del contesto che tenga in considerazione i dati associati alla loro granularità informativa e dove significativo ad un periodo discreto di tempo che descriva la frequenza con cui devono essere ricalcolati. Questi parametri sono considerati nello svolgimento delle operazioni: la granularità informativa è utilizzata per l'esecuzione di sottoprocessi a differente livello di dettaglio a seconda della condizione contestuale in cui si svolge l'applicazione. La granularità di monitoraggio è invece aggiornata dinamicamente a seconda di un'analisi proattiva del contesto di esecuzione, permettendo all'applicazione di utilizzare dati aggiornati anche in situazioni anomale, prevedendole sulla base di stime del comportamento dell'applicazione effettuate prima del verificarsi dell'anomalia sulla base di dati storici e con il supporto di tecniche di apprendimento automatico. Essendo i dati contestuali monitorati a runtime ed in riferimento alla singola istanza

dell'applicazione in esecuzione, viene proposto un meccanismo di invocazione dinamica dei servizi, sulla base della configurazione e dell'utilizzo di un proxy service. Inoltre, per la realizzazione di processi di business complessi vengono utilizzati tool di gestione dei flussi di lavoro allo scopo di orchestrare i singoli web service componenti, definendone l'ordine di esecuzione e le condizioni sotto le quali devono essere svolti.

Nel framework i dati contestuali utilizzati per l'analisi dell'applicazione sono salvati in relazione alla propria granularità all'interno del database. Si suppone quindi che i dati non siano monitorati ed estratti fisicamente con il giusto livello di dettaglio da reti di sensori a runtime, ma sia solo analizzata la granularità dei dati da utilizzare nello svolgimento delle operazioni. Sarebbe interessante estendere il framework per considerare anche l'estrazione e la misurazione a runtime del dato in relazione al livello di dettaglio più idoneo, allo scopo di cercare di ridurre oltre allo spreco in termini operazionali e temporali anche quello in termini di energia.

Bibliografia

[1] Luca Bazzani. L'analisi abc per la gestione del magazzino, 2011.

[2] S. Bergamaschi, F. Guerra, and M. Vincini. A data integration framework for e-commerce product classification. In *International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*. Springer, 2002.

[3] A. Bucchiarone, C. Cappiello, E. Di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore. Design for adaptation of service-based applications: Main issues and requirements. In *ICSOC/ServiceWave Workshops*, pages 467–476, 2009.

[4] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. Di Nitto, and V. Mazza. A context-driven adaptation process for service-based applications. In *PESOS 2010 – 2nd International Workshop on Principles of Engineering Service-Oriented Systems. (To Appear)*, 2010.

[5] E. Camossi, M. Bertolotto, and E. Bertino. Spatio-temporal multigranularity: Modelling and implementation challenges. Technical report, 2009.

[6] Cinzia Cappiello, Marco Comuzzi, Enrico Mussi, and Barbara Pernici. Context management for adaptive information systems. *Electr. Notes Theor. Comput. Sci.*, 146(1):69–84, 2006.

[7] Facoltà di economia UniCal. La gestione delle scorte, 2011.

[8] S. Dustdar and W. Schreiner. A survey on web services composition. *IJWGS*, 1(1):1–30, 2005.

[9] C. Dyreson, W. Evans, H. Lin, and R. Snodgrass. Efficiently supported temporal granularities. *IEEE Trans. Knowl. Data Eng.*, 12(4):568–587, 2000.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and H. Ian. The weka data mining software, 2009.

[11] W. Hummer, A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. *VRESCo - Vienna Runtime enviroment for service-oriented computing.* Springer, 2010.

[12] Philipp Leitner, Florian Rosenberg, and Schahram Dustdar. Daios: Efficient dynamic web service invocation. *IEEE Internet Computing*, 13(3):72–80, 2009.

[13] V. Manganaro. Tecniche di dm: Alberi di decisione ed algoritmi di classificazione.

[14] A. Metzger, E. Schmieders, C. Cappiello, E. Di Nitto, R. Kazhamiakin, B. Pernici, and M. Pistore. Towards proactive adaptation: A journey along the s-cube service life-cycle. In *MESOA: 4th International Workshop on Maintenance and Evolution of Service-Oriented Systems*, 2010.

[15] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. Publish/subscribe in the vresco soa runtime. *ACM*, 2008.

[16] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. Comprehensive qos monitoring of web services and event-based sla violation detection. *ACM Press*, 2009.

[17] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. End-to-end support for qos-aware service selection, binding, and mediation in vresco. *IEEE T. Services Computing*, 3(3):193–205, 2010.

[18] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. Monitoring, prediction and prevention of sla violations in composite services. In *ICWS*. IEEE Computer Society, 2010.

[19] S. Modafferi, B. Benatallah, F. Casati, and B. Pernici. *A Methodology for Designing and Managing Context-Aware Workflows*. Springer-Verlag, 2005.

[20] B. Pernici. *Mobile information systems: infrastructure and design for adaptivity and flexibility*. Springer, 2006.

[21] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *TAAS*, 4(2), 2009.

[22] M. Treiber, H. Truong, and S. Dustdar. On analyzing evolutionary changes of web services. *ICSOC Workshops*, pages 284–297, 2008.

[23] Richard T. Watson, Marie-Claude Boudreau, and Adela J. Chen. Information systems and environmentally sustainable development: Energy informatics and new directions for the is community. *MIS Quarterly*, 34(1):23–38, 2010.

[24] Weka wikipage. Use weka in your java code.

[25] Wikipedia. Gestione delle scorte — wikipedia l'enciclopedia libera, 2011.

[26] A. Zisman, J. Dooley, and G. Spanoudakis. Proactive runtime service discovery. In *IEEE SCC*. IEEE Computer Society, 2008.