ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

SMART - KITCHEN

Elaborata nel corso di: Sistemi Multi Agente LM

Relazione di:
MATTEO AMADUCCI,
LORENZO GARATTONI,
DAVIDE DUSI

Professore:
Prof. ANDREA OMICINI

ANNO ACCADEMICO 2010–2011

Indice

Capitolo 1

Analisi dei requisiti

1.1 Il sistema

Molte persone, oggi giorno, preferiscono utilizzare il web per la ricerca di ricette, anziché sfruttare la propria collezione di libri di cucina. Google ha addirittura stimato che piú dell' 1% delle ricerche effettuate giornalmente sono dedicate al settore della cucina. Il progetto SmartKitchen ha come scopo fondamentale quello di rendere ancora piú facile ed efficace la ricerca di ricette online sfruttando il concetto di semantica. Le ricette proposte dal sistema, infatti, tengono in considerazione non solo quello che l'utente desidera ricercare, come avviene nelle comuni ricerche web, ma anche quali ingredienti e tool da cucina sono necessari per la realizzazione di ogni singola ricetta, confrontando tali informazioni con quelle raccolte dagli elettrodomestici della cucina, come frigorifero e dispensa. In questo modo SmartKitchen é in grado di fornirci, per ogni ricetta, quanti e quali ingredienti mancano nella nostra cucina per poterla realizzare, e per ognuno di essi vengono proposti eventuali ingredienti alternativi. Ancora, in caso di ricerche poco precise , il sistema é in grado di individuare l'area di ricerca voluta dall'utente ed ampliare questúltima proponendo ricette che contengono ingredienti piú specifici. In sintesi, SmartKitchen é in grado di comprendere, dal punto di vista semantico, quello che lútente sta ricercando ed é capace di organizzare le informazioni trovate nella maniera piú comoda anche agli occhi di un utente poco esperto nel settore.

1.2 Casi d'uso

Il sistema descritto nel paragrafo precedente é pensato principalmente per un ambiente domestico in cui spesso gli utenti faticano a cimentarsi nella realizzazione di ricette e piatti sempre nuovi e gustosi per mancanza di tempo e organizzazione. Il progetto si propone quindi di facilitare la vita in cucina di molte persone, facendosi carico di alcune attivitá, anche complesse o dispendiose dal punto di vista temporale, normalmente svolte da utenti umani. Il caso d'uso principale riguarda dunque la situazione in cui un utente, probabilmente con poca esperienza in cucina o con tempo libero a disposizione molto limitato, chiede al sistema di proporgli delle ricette di suo gradimento e realizzabili nei tempi da lui specificati e con i soli ingredienti e utensili disponibili all'interno della cucina.

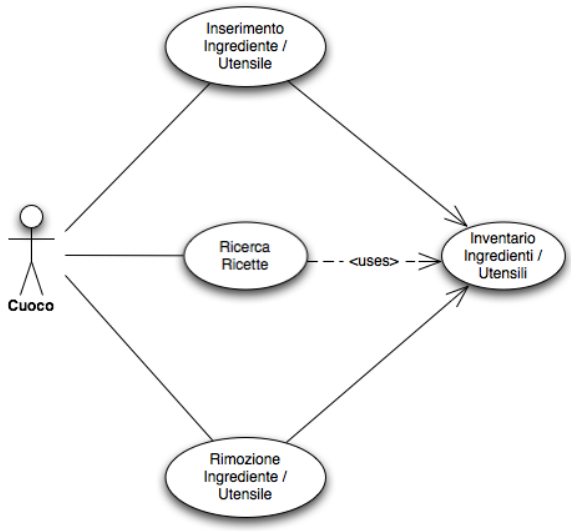


Figura 1.1: *Use case.*

Il sistema proposto potrebbe essere facilmente estendibile anche in ambiti più professionali, come cucine di ristoranti, in cui una migliore organizzazione e un'automatizzazione della spesa sarebbe, oltre che una comodità, un vantaggio dal punto di vista economico. Lo stesso discorso può essere esteso anche ad altri domini applicativi in cui la gestione e il controllo automatico dei magazzini e delle merci necessarie alla produzione potrebbe portare svariati benefici.

Capitolo 2

Analisi del problema

2.1 Architettura

In questa sezione saranno descritti i componenti fondamentali individuati per la realizzazione dell'intero sistema. La suddivisione proposta prevede l'utilizzo del meta modello AA (Agent and Artifact) con l'aggiunta di una sezione dedicata all'ontologia. Gli agenti rappresentano la parte attiva di un sistema software, sono in grado di svolgere compiti con un certo grado di autonomia, e di percepire l' ambiente e i suoi cambiamenti agendo di conseguenza. Gli artefatti, al contrario degli agenti, compongono la parte passiva del sistema e mettono a disposizione servizi tramite i quali gli agenti possono interagire con l'ambiente circostante. Il terzo ed ultimo componente fondamentale del sistema é l'ontologia, la quale raccoglie ed organizza una descrizione semantica del dominio applicativo.

2.1.1 Agenti

In questa sezione, saranno presentate le categorie logiche di agenti necessarie per la realizzazione di tutte le funzionalitá previste dal sistema.

- ***Agenti per la ricerca web :*** questa categoria di agenti riceve stringhe da utilizzare per la ricerca delle ricette sul web e, successivamente, riceve le ricette trovate in corrispondenza di tali ricerche . Ogni agente di questo tipo é in grado di richiedere un numero di ricerche web pari al numero di stringhe ad esso recapitate.

- ***Agenti per la raccolta delle informazioni all'interno della cucina:*** questa tipologia di agenti risiederanno all'interno degli elettrodomestici che compongono la cucina, e che contengono quindi ingredienti e tool per cucinare. In SmartKitchen si ha la necessitá di gestire un frigorifero ed una dispensa. Gli agenti in questione si incaricheranno di seguire le procedure di inserimento e rimozione di nuovi elementi dai due elettrodomestici, saranno quindi responsabili di redigere dinamicamente una lista, sempre aggiornata, degli ingredienti e dei tool a disposizione dell' utente.

- ***Agente per la GUI di ricerca :*** questo agente rende possibile l' interazione tra l' utente finale e il sistema. Le informazioni di ricerca inserite dall utente vengono, infatti, prelevate dall' agente in questione e girate ad agenti interni al sistema che poi effettueranno le dovute elaborazioni. Nel sistema saranno presenti tanti agenti di questa categoria quanti sono il numero di GUI di ricerca (solitamente una per ogni utente), e possono essere collocati su dispositivi mobili, quali smartphone, o in generale su dispositivi di comune utilizzo da parte dell' utente.

- ***Agente coordinatore :*** questo agente rappresenta il cuore del sistema. Esso riceve le informazioni inserite dall'utente per la ricerca, é tenuto costantemente aggiornato su ogni cambiamento che avviene all'interno del frigorifero e della dispensa, in maniera tale da avere una visione sempre aggiornata dello stato della cucina, sfrutta l'ontologia per trasformare le informazioni recapitatogli dall'ambiente esterno nel linguaggio interno usato dal sistema, e rende disponibili tali informazioni per la ricerca. Grazie alle operazioni svolte da questo agente é possibile dare, alle stringhe di ricerca inserite dall'utente, e agli ingredienti presenti delle ricette online, una semantica comune all'interno del sistema, aumentando il contenuto informativo che viene recapitato all'utente finale.

2.1.2 Artefatti

In questo paragrafo vengono presentati i principali artefatti che compongono il sistema, mettendo in evidenza soprattutto i servizi che questi mettono a disposizione degli agenti per il raggiungimento dei requisiti funzionali del progetto.

- ***Artefatti FridgeArtifact e PantryArtifact (GUI):*** FridgeArtifactGUI e PantryArtifactGUI sono artefatti con due funzionalità distinte:

  - Il primo obiettivo è quello di simulare il comportamento di un frigorifero e di una dispensa intelligenti. In particolare, questi dispositivi dovrebbero essere in grado di riconoscere gli ingredienti e gli utensili che l'utente inserisce al loro interno o rimuove. A tale scopo, sopra ognuno di questi elementi dovrà essere applicato un tag RFID o un codice a barre contenente tutte le informazioni, necessarie all'elettrodomestico in questione, per riconoscere gli oggetti e mantenere un proprio inventario. Lo scopo di questa categoria di artefatti è simulare tale funzionalità, permettendo direttamente all'utente di simulare l'inserimento e la rimozione di ingredienti da frigo o dalla dispensa specificando, attraverso un'intefaccia grafica, tutte le informazioni relative allo specifico oggetto (nome ingrediente, tipo, quantità ecc.)
  - Dall'altra parte questi artefatti fungono da interfaccia con cui gli agenti che compongono il sistema intergiscono con l'ambiente, in questo caso con il frigorifero e la dispensa intelligenti. Forniscono metodi per ottenere la lista degli ingredienti e degli utensili che si trovano all'interno dei due elettrodomestici.

- **Artefatto per la ricerca online**: questo artefatto è l'interfaccia che il sistema possiede verso il mondo di Internet, in particolare verso quelle pagine web che ospitano siti di cucina e ricettari. L'artefatto deve fornire agli agenti la possibilità di inserire una o più query di ricerca ottenendo come risultato tutte le ricette che soddisfano tali interrogazioni.

- **Artefatto ontologia**: come approfondito nel prossimo paragrafo, l'ontologia è una parte fondamentale del sistema che permette di rendere più efficace e utile ogni funzionalità che esso offre. Questo artefatto ha lo scopo di abilitare l'interazione fra gli agenti e l'ontologia, fornendo metodi utili per ottenere il concetto associato ad un termine, cercare eventuali concetti alterantivi, rendere una query più specifica e altre funzionalità che verranno approfondite nei prossimi capitoli.

- **Artefatto GUI** : questo artefatto è l'interfaccia che SmartKitchen mette a disposizione dell'utente. Attraverso la GUI, l'utilizzatore potrà inserire i termini di ricerca delle ricette e visualizzare ciò che il sistema gli propone in base agli ingredienti e utensili presenti in cucina, insieme a eventuali alternative e suggerimenti. L'applicazione prevede l'ampliamento a più artefatti GUI, uno per ogni dispositivo di intefacciamento con l'utente presente nel sistema come smartphone, interfaccia del frigorifero e così via.

2.1.3 Ontologia

L' ontologia infine é l'ultimo dei tre principali componenti del sistema SmartKitchen, viene utilizzata sostanzialmente per estendere le query di ricerca, per inferire nessi semantici tra gli ingredienti e dedurre equivalenze tra unitá di misura e utensili. La necessitá di usare un'ontologia é nata per fornire all'utente alcune interessanti funzionalitá: per consigliare ingredienti nel caso di inserimento di una categoria di alimenti(e.g Volatile, Carne Rossa, etc.), la traduzione di alcune particolari unitá di misura che non sono direttamente riconducibili a grandezze fisiche, edinfine il suggerimento di ingredienti che potrebbero sostituirne uno richiesto da una ricetta ma che non é momentaneamente disponibile all'interno del Fridge o nella Pantry.

In particolare l'ontologia viene utilizzata con lo scopo di classificare [Wikipedia] . Gli alimenti vanno a formare la gerarchia piú estesa al fine di avere la possibilitá di riconoscere il maggior numero di ingredienti ed avere cosí a disposizione il supporto induttivo dell'ontologia. Inoltre la presenza di gerarchie che riguardano le grandezze fisiche e gli utensili aiutano per la traduzione di unitá di misura particolari, é comune infatti nelle ricette l'uso di diciture come "un cucchiaio di ...", "un bicchiere di ...", le quali non rappresentano esattamente grandezze fisiche ed é quindi stato previsto un meccanismo grazie al quale un utensile viene ricondotto ad una particolare quantitá con relativa unitá di misura normalizzata(vedi tabella).

Normalizzazione Strumenti	
Bicchiere	0.05 litri
Cucchiaio Piccolo	0.005 kg
Cucchiaio Grande	0.023 kg

Figura 2.1: *Tabella normalizzazione.*

2.2 Interazione

In questa sezione verranno descritte le modalitá di interazione realizzate all'interno del sistema. I soggetti interessati in tali interazioni

sono agenti, artefatti e l'utente finale. Nei sistemi multi agente le entitá in gioco (agenti ed artefatti) possono comunicare tra loro in tre differenti modi:

- **Mediante operazioni**: gli artefatti mettono a disposizione degli agenti un set di operazioni. Essi rappresentano dei processi computazionali eseguiti all'interno dell'artefatto. Le operazioni sono correlate da un'interfaccia che li rende noti all'ambiente esterno e quindi utilizzabili dagli agenti.

- **Mediante variabili osservabili** : queste variabili rappresentano variabili di stato dell'artefatto il cui cambiamento puó essere percepito all'esterno da chiunque sia in ascolto su di esse. Il valore di tali variabili puó cambiare dinamicamente, anche come risultato dell'esecuzione di un operazione interna.

- **Mediante signal** : l'esecuzione di un'operazione puó, oltre che cambiare il valore di una variabile di stato, anche generare una signal. Questa rappresenta un evento avvenuto all'interno di un artefatto e puó contenere informazioni utili alla descrizione di tale evento organizzate in una tupla. La generazione di una signal produce una variazione delle conoscenze di base dell'agente che si trova in ascolto dell'artefatto che l'ha generata.

Per quanto riguarda l'interazione tra il sistema e l'utente finale viene in aiuto la classica interfaccia grafica. L'altra tipologia di interazione fondamentale all'interno dei sistemi multi agente é quella fra gli agenti che lo compongono. Le piattaforme di sviluppo per i suddetti sistemi mettono a disposizione primitive mediante le quali gli agenti sono in grado di inviare e ricevere informazioni, solitamente organizzate come tuple, che rappresentano nuovi belief o goal da raggiungere. Di seguito saranno riportate le interazioni nei tre scenari principali del sistema SmartKitchen, ovvero durante l'inserimento e la rimozione di un ingrediente dal frigorifero (o dalla dispensa), la richiesta di ricerca di una ricetta da parte dell'utente e l'insieme dei due scenari descritti precedentemente.

2.2.1 Inserimento e rimozione di un ingrediente

Un utente puó decidere, in qualsiasi istante, di rimuovere o aggiungere un ingrediente, o un tool, dalla propria cucina. Questa operazione é molto importante, poiché va ad alterare la conoscenze di base che

il sistema possiede per poter ricercare le ricette in maniera efficace. Non appena il coordinatorAgent viene creato, ed entra a far parte degli agenti attivi del sistema, si realizza la prima comunicazione tra agenti, viene infatti avanzata una richiesta, al FridgeAgente, per ottenere gli ingredienti all'interno del frigor (medesima interazione per la dispensa). Il FridAgent inoltra la richiesta all'artefatto che realizza il frigorifero (e la dispensa), mediante l'invocazione dell'operazione getIngridients realizzata al suo interno. Il risultato di tale operazione sará comunicata al richiedente come valore di ritorno al termine dell'esecuzione dell'operazione stessa, mentre il FridgeAgent comunicherá la lista al coordinatorAgent, sfruttando le primitive messe a disposizione della piattaforma, le quali permettono la creazione di nuova conoscenza di base all'interno dell'agente. Per realizzare l'inserimento o la rimozione di un ingrediente é necessario l'interazione del sistema con l'utente. Quest'ultimo, infatti, invoca, all'interno dell'artefatto FridgeArtifactGUI, l'operazione che realizza l'inserimento di un nuovo ingrediente chiamata addIngredient. Questa azione provoca una reazione a catena, che vede coinvolti tutti gli agenti a valle del FridgeArtifactGUI. Mediante un'operazione di signal, infatti, l'artefatto frigorifero avverte, l'agente FridgeAgent, dell'esistenza di un nuovo ingrediente, mentre quest'ultimo realizza una comunicazione con il coordinatorAgent, inserendo nei propri belief, la presenza del nuovo ingrediente. La procedura di rimozione di un ingrediente, o di un tool, dal frigorifero, o dalla dispensa, avviene nel medesimo modo. Grazie a questo meccanismo di interazione, tra utente, agenti ed artefatti, é possibile monitorare dinamicamente il cambiamento di stato della cucina, rappresentato dagli ingredienti e tool presenti in essa.

2.2.2 Ricerca ricette

Il secondo scenario tipico del sistema smart-kitchen è quello che coinvolge maggiormente l'utente finale che, dopo aver inserito alcuni termini di ricerca, si aspetta che gli venga proposta una serie di ricette e di suggerimenti realizzabili nei tempi indicati e con il materiale già a sua disposizione in cucina. Lo scenario di interazione descritto ha inizio, dunque, all'inserimento dell'interrogazione dell'utente. La query viene prelevata dall'artefatto SearchArtifactGUI, notificata attraverso una signa all'agente agganciato all'artefatto che a sua volta la comunica al CoordinatorAgent. Il CoordinatorAgent, a questo punto, segnalerà l'inserimento di una nuova query all'agente incaricato di svolgere le ricerche online, il ResearchAgent. Quest'ultimo, dopo aver utizzato

le operazioni messe a disposizione dall'artefatto ResearchArtifact per dare inizio alla nuova ricerca, comincierà a ricevere come risposta le ricette trovate sul web (la prima ricetta, se ci sono risultati per la query inserita, sarà restituita immediatamente mentre le successive saranno notificate tramite delle signal per fare in modo che l'agente rimanga libero di ricevere ulteriori richieste di ricerca). Ogni risultato prodotto dall'interrogazione online viene tornato all'agente coordinatore, che a questo punto, prima di presentarlo come proposta direttamente all'utente deve filtrarlo in base alla sua conoscenza dello stato della cucina:

- Preleva la lista degli ingredienti e degli utensili, derivata dall'esecuzione (per ogni inserimento o rimozione) dello scenario descritto in precedenza.
- Richiede all'ontologyArtifact di filtrare la ricetta in base alla lista prelevata.
- Invia la ricetta filtrata al SearchGUIAgent che, grazie ad un'operazione dell'artefatto GUI la mostrerà all'utente.

Lo scenario poi continua, in quanto, è possibile che fra l'interrogazione inserita dall'utente sia, in parte o nella sua interezza, vaga e che sia quindi possibile, grazie ad una funzionalità dell'OntologyArtifact richiesta dall'agente coordinatore, migliorarla e renderla più precisa aumentando le possibili soluzioni ricercate. La nuova query ottenuta viene inviata ancora una volta al ResearchAgent che ripete l'operazione precedente. Anche le ricette restituite grazie a questa ricerca espansa dovranno essere filtrate e mostrate infine all'utente come approfondito in precedenza.

2.2.3 Inserimento o rimozione di un ingrediente durante l'esecuzione di una ricerca

Come anticipato in precedenza, questo scenario é frutto della combinazione dei due precedenti. Puó accadere infatti che, durante la ricerca di ricette, l'utente aggiunga o rimuova elementi dal frigorifero o dalla dispensa, cosí da costringere il sistema a reagire, rivalutando tutte le ricette precedentemente consegnate all'utente. Con il termine rivalutare si intende l'azione con la quale il sistema ricontrolla gli ingredienti mancanti, e inserisce eventuali nuovi suggerimenti. Anche in questo scenario, come nei precedenti, il coordinatorAgente é di vitale importanza per il funzionamento sopra descritto. Durante una

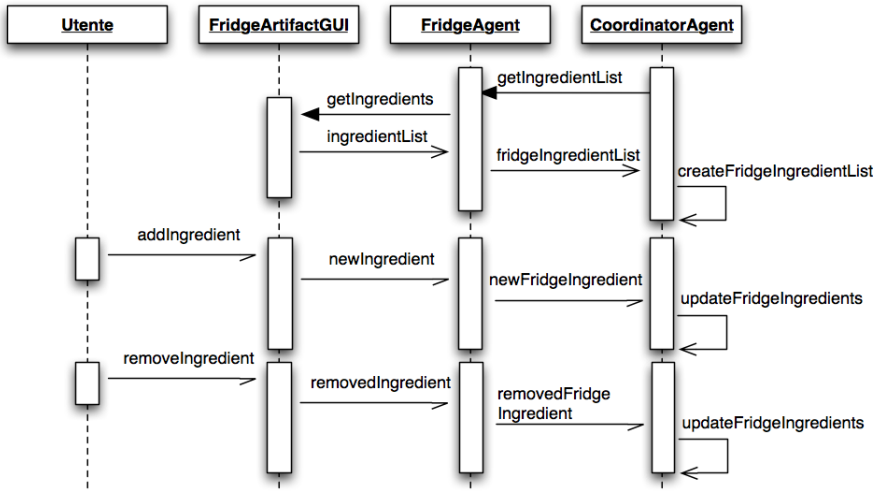


Figura 2.2: *Interazione tra agenti, artefatti ed utente durante l'inserimento e la rimozione di un ingrediente dal frigorifero.*

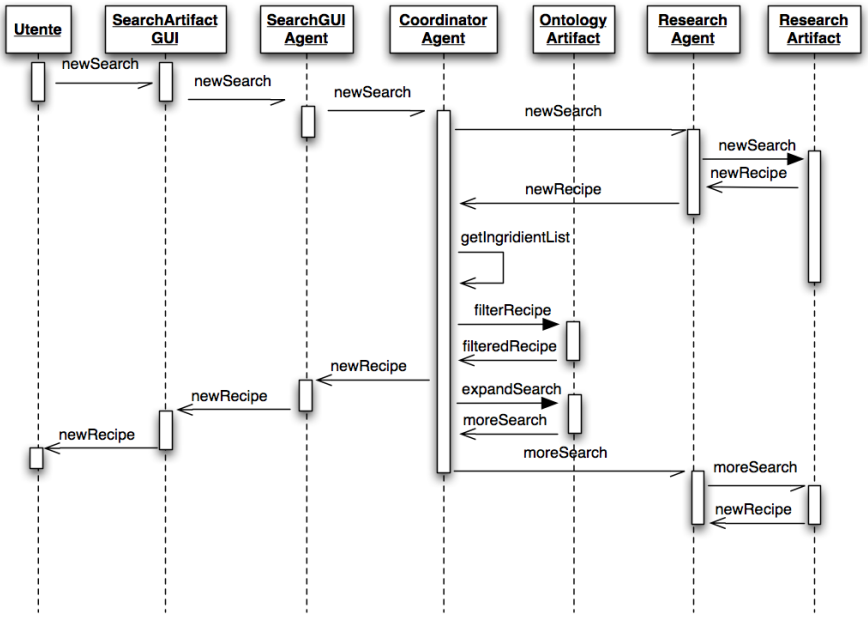


Figura 2.3: *Interazione tra agenti, artefatti ed utente durante la fase di ricerca ricette.*

ricerca, infatti, esso é sempre pronto a ricevere notifiche di aggiornamento provenienti dal frigorifero e/o dalla dispensa, in questo modo ogni inserimento o rimozione di un elemento dalla cucina puó essere catturato ed utilizzato per mantenere aggiornata la conoscenza interna del sistema. In altre parole, se al coordinatorAgent viene notificata la presenza di un nuovo ingrediente durante l'esecuzione di una ricerca, esso si incarica di prendere tutte le ricette fino a quel momento mostrate all'utente, e di controllare se la lista degli ingredienti mancanti, o suggeriti, cambia a fronte del nuovo ingrediente aggiunto in cucina. In caso affermativo, le ricette successivamente mostrate all'utente saranno aggiornate, tenendo conto del cambiamento avvenuto in cucina, viceversa, se il cambiamento in cucina non provoca variazioni per quanto riguarda le ricette cercate, esse verranno riproposte all'utente senza alcun cambiamento.

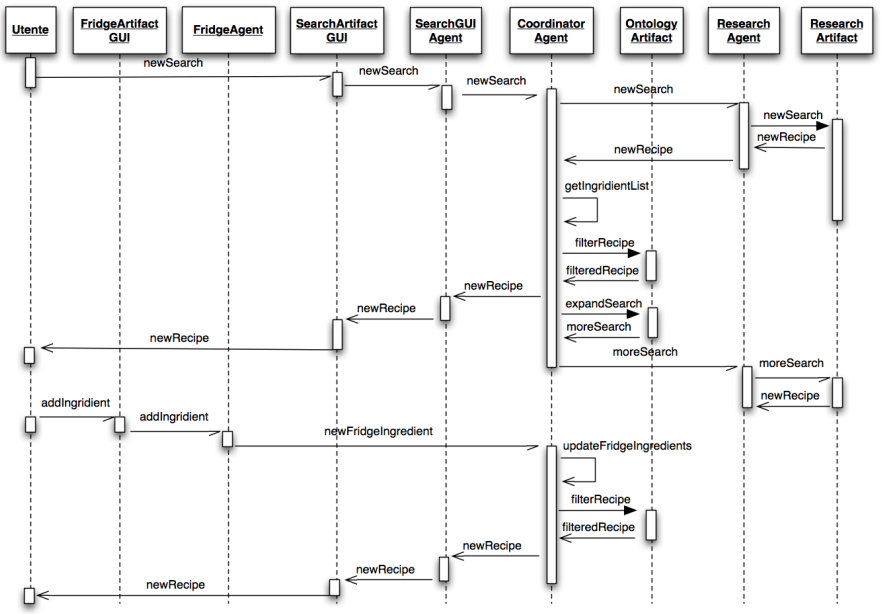


Figura 2.4: *Interazione tra agenti, artefatti ed utente durante la fase di ricerca ricette e modifica degli ingredienti nella cucina.*

2.3 Comportamento

Durante gli scenari descritti nel paragrafo precedente, il sistema assume diversi comportamenti interni. Questi sono necessari per estendere

la funzionalitá base del sistema, ovvero effettuare ricerca online di ricette, introducendo la possibilitá di assegnare un significato semantico a tutti i dati che il sistema stesso deve elaborare. Di seguito vengono descritte pi´ù nel dettaglio le operazioni svolte all'interno del sistema che ci consentono di arricchire le ricerche online con la semantica di ció che si desidera trovare.

2.3.1 Ampliamento della ricerca

L'ampliamento della ricerca é necessario per aumentare la conoscenza relativa a quello che deve essere ricercato. Puó capitare, infatti, che la stringa inserita dall'utente sia vaga o poco specifica, in tal caso il sistema, grazie all'ampliamento della ricerca, é in grado di aumentare il numero di risultati delle ricette trovare, e di aumentarne anche la precisione. Questa operazione é immediatamente successiva all'inserimento, da parte dell'utente, della stringa da ricercare all'interno della GUI, e viene eseguita mentre il sistema effettua la ricerca utilizzandola stringa originale. Vediamo meglio in cosa consiste questa fase, come viene realizzata e cosa fornisce come risultato finale.

La stringa originale, inserita dall'utente della GUI di ricerca, viene prima di tutto recapitata agli agenti responsabili della ricerca online. Viene quindi realizzata una prima fase di ricerca, che fa uso della semplice stringa inserita dall'utente, senza alcuna modifica. Mentre il sistema continua nella sua ricerca, e recapita le prime ricette trovate all'utente, il sistema si prepara ad analizzare la stringa di ricerca per tentare di estrapolare, da essa, altre informazioni utili relative alla semantica della ricetta alle quali l'utente é interessato. Se l'informazione semantica racchiusa all'interno dell'ontologia é sufficientemente ampia da consentire un mapping semantico della stringa, allora si procede con l'espansione della ricerca. Ad esempio, facendo riferimento al dominio applicativo di SmartKitchen, l'utente potrebbe voler ricercare tutte le ricette che hanno come ingrediente principale un volatile. Il sistema é in grado, in questo caso, di comprendere che la famiglia dei volatili racchiude un insieme di animali piú specifico, come ad esempio oche e anatre. A questo punto, la ricerca viene ampliata, e i nuovi termini vengono aggiunti alla ricerca giá in atto. Il responsabile di questo mapping fra sintassi e semantica é affidato all'artefatto Ontology, il quale si avvale di un parser, meglio descritto in fase di implementazione, il quale realizza l'analisi sintattica della stringa, consentendo l'estrapolazione i ogni singola parola della ricerca, e di un'ontologia

che racchiude tutta le gerarchia di ingredienti sotto forma di concetti semantici. Il meccanismo che realizza tale funzionalitá é il seguente:

- Per prima cosa si estrapola un termine per volta a partire dalla stringa originale inserita dall'utente.
- Si effettua una ricerca sull'ontologia utilizzando come stringa di query il termine precedentemente estrapolato
- Se la query restituisce un risultato non vuoto allora é possibile ricavare termini simili a quello cercato dall'utente ed andarli ad aggiungere come termini validi per la ricerca di ricette

Grazie all'espansione della ricerca il sistema é quindi in grado di fornire ricette che contengono ingredienti piú specifici di quelli inseriti in fase di ricerca.

2.3.2 Parsing degli ingredienti descritti in una ricetta

L'inserimento di una nuova query di ricerca da parte dell'utente scatena una serie di azioni interne del sistema che, per prima cosa, reagisce passando al ResearchAgent l'intergozione appena inserita. Questo agente, attraverso l'interazione con l'artefatto che si interfaccia con il mondo Internet (ResearchArtifact), comincia la ricerca di ricette online. A questo punto, se la query produce dei risultati, le ricette che soddisfano i termini di ricerca comincieranno ad arrivare all'agente, prelevate direttamente dal contenuto di pagine web. Ogni ricetta contiene una sezione dove sono elencati gli ingredienti richiesti, una che indica il tempo necessario per la preparazione e l'ultima contenente le istruzioni da seguire (su alcuni siti sono presenti anche il grado di difficoltá della ricetta, la fotografia del piatto finale e altre informazioni aggiuntive). A questo punto, il sistema dovrá essere in grado di capire le parti di ogni ricetta, prima di tutto suddividendo le sezioni e successivamente concentrandosi sul contenuto di ognuna. La parte piú importante é senza dubbio l'elenco degli ingredienti necessari alla preparazione della ricetta, in quanto é su questa informazione che il sistema Smart-Kitchen svolgerá i suoi controlli per proporre i risultati all'utente e le possibili alternative. É quindi necessario essere in grado di dividere un ingrediente dagli altri, associandogli tutte le informazioni specificate nel testo della ricetta e su cui i successivi calcoli si

concetreranno. É importante osservare inoltre che i modi per specificare un ingrediente possono essere molto vari, sia dal punto di vista sintattico, ad esempio in un'unica riga possono essere specificati più ingredienti separati da un '+' oppure dopo il nome possono essere inserite informazioni aggiuntive fra parentesi, sia da quello semantico in quanto uno stesso alimento puó essere indicato con nomenclature diverse. Per raggiungere gli scopi sopraccitati é indispensabile l'introduzione di un componente nel sistema, da inserire all'interno dell'artefatto ontology reponsabile del mapping sintassi-semantica, che permetta di analizzare sintatticamente il contenuto della sezione ingredienti di ogni ricetta, estrapolandone per ogni ingrediente il nome, la quantitá richiesta come numero intero o decimale e l'unitá di misura della quantitá. Come output del nuovo componente Parser si avrá quindi una lista di ingredienti con associate tutte le informazioni necessarie alle fasi successive, che consistono nell'associazione di ogni ingrediente alla suo concetto (intesa come categoria di alimenti a cui appartiene, nell'accezione data dall'ontologia descritta nel paragrafo 2.1.3), la normalizzazione delle unitá di misura e delle quantitá e la ricerca degli stessi ingredienti nella cucina dell'utente per poter mostrare, infine, le ricette possibili da realizzare e quelle con ingredienti non disponibili (specificando laddove possibile le alternative da poter utilizzare).

2.3.3 Normalizzazione degli ingredienti di una ricetta

Lo fase successiva al download della ricetta dal web e all'analisi sintattica portata a termine dal parser e descritta nel paragrafo precedente, é la normalizzazione degli ingredienti. Dal parser, infatti, esce una lista di ingredienti a cui sono associate tutte le informazioni specificate nella ricetta (il nome, la quantitá necessaria alla preparazione della ricetta in relazione all'unitá di misura). Queste informazioni rimangono peró del tutto eterogenee fra loro, ad esempio ingredienti che si riferiscono allo stesso alimento possono essere espressi con nomi diversi o essere quantificati con unità di misura eterogenee, renderdo il confronto impossibile. La normalizzazione di un ingrediente si compone quindi di due fasi:

- *Normalizzazione del nome:* é la fase in cui si ricava, a partire dal nome specificato nella ricetta, la classe (e di conseguenza la gerarchia di classi) di alimenti a cui esso appartie-

ne. A questo scopo si interroga l'ontologia di alimenti realizzata, attraverso l'interfaccia messa a disposizione dall'artefatto OntologyArtifact.

- *Normalizzazione della quantitá:* questa fase é forse più complessa della precedente, in quanto le quantitá e le unitá di misura espresse all'interno della ricette possono essere le piú diverse ed etoregenee. In cucina vengono usate tante unitá di misura, alcune precise e derivate dai sistemi internazionali, altre molto vaghe e approssimative mentre altre riferite ad utensili (come un cucchiaio o un bicchiere). É necessario quindi, per avere dei termini di confronto comuni, riportare ogni misura ad un'unica unitá per ogni stato (solido, liquido ecc). Ancora una volta viene utilizzata l'ontologia, all'interno della quale sono rappresentate le gerarchie e le equivalenze fra le diverse unitá in modo da avere i rapporti che permettono la normalizzazione.

Terminata questa fase sará possibile confrontare gli ingredienti specificati nelle ricette online con quelli presenti in cucina, misurandone anche le quantitá richieste e quelle disponibili in modo da poter decidere in modo preciso le indicazioni da riportare all'utente.

2.3.4 Suggerimento ingredienti alternativi a quelli mancanti

Una delle funzionalitá piú interessanti, e se vogliamo, intelligenti del sistema é quella di essere ingrado di suggerire, all'utente, ingredienti alternativi da poter utilizzare in mancanza di quelli necessari alla realizzazione della ricetta cercata. Anche questo meccanismo, come quelli precedenti, si basa sul concetto di semantica degli ingredienti, che consente a SmartKitchen di comprendere la natura di ogni singolo ingrediente presente nella cucina, e proporlo cosí come valida alternativa per ingredienti mancanti. Dopo aver parsato i diversi ingredienti delle ricette e averli normalizzati facendo uso dell'ontologia, gli ingredienti della ricetta sono paragonabili, in termini semantici, a tutti gli ingredienti presenti nella cucina, poiché ricondotti tutti ad una classe concettuale ben nota all'interno del sistema. Quando un ingrediente di una ricetta, non compare all'interno del frigorifero o della dispensa, il sistema tenta di sostituire quest'ultimo con un ingrediente il piú possibile equivalente, ed ovviamente, in possesso dell'utente. Utilizzando l'ingrediente mancante, ne viene estrapolata la classe

concettuale eseguendo una query all'interno dell'ontologia, dopodiché vengono prelevati, dalla gerarchia, tutti i termini che rappresentano ingredienti simili a quello mancante. Dalla lista risultante vengono ovviamente eliminati gli ingredienti non presenti nella cucina (poiché l'ontologia racchiude un'insieme di concetti che sono indipendenti dall'istanza applicativa in cui essa viene utilizzata), e anche tutti quelli presenti in quantitá inferiore rispetto a quella richiesta dalla ricetta. Gli ingredienti restanti vengono quindi proposti all'utente come possibile alternativa all'ingrediente mancante. Ovviamente la qualitá del suggerimento é tanto piú buona quanto é accurata la creazione della gerarchia all'interno dell'ontologia. Si potrebbe pensare, infatti, di creare addirittura un'ontologia utilizzata solamente per produrre suggerimenti di questo genere, e che al suo interno quindi crei gerarchie di ingredienti basati sulle equivalenze di ingredienti piuttosto che sulla tipologia.

Capitolo 3

Implementazione

3.1 Premessa

In questo capitolo saranno descritti tutti i componenti sviluppati all'interno del sistema e come questi interagiscono per raggiungere le funzionalitá richieste. La primo paragrafo é dedicato ad una breve introduzione a tutti i tool esterni di supporto utilizzati nello sviluppo, alcuni di essi costituiti semplicemente da librerie e APIs, altri anche da ambienti grafici. Nei paragrafi successivi verranno discusse le modalitá attraverso le quali sono state implementate le entitá che compongono il sistema, suddivise in base alle fasi o alle funzionalitá per cui esse sono state introdotte. Infine verrá presentata una panoramica del sistema finale. Prima di cominciare la descrizione dettagliata é necessario fare alcune premesse fondamentali per comprendere alcuni aspetti dell'implementazioni e soprattutto dei comportamenti del sistema una volta terminato:

- In questa prima realizzazione, Smart-Kitchen sará in grado di effettuare ricerche on-line solamente all'interno di un sito web, in particolare il sito www.gustissimo.it. Questa scelta é dovuta al fatto che le pagine web, soprattutto quelle riguardanti ricette culinarie, sono strutturate in modi molto diversi e l'estrapolazione delle informazioni di interesse al loro interno é quindi difficile da standardizzare. L'utilizzo di questa pagina é dunque da considerarsi un esempio, estendibile, grazie ad opportune modifiche ad-hoc, ad altri siti. C'é da dire inoltre che per l'estrapolazione delle informazioni all'interno di pagine web potrebbero essere pensati ed utilizzati altri meccanismi come algoritmi di text mining, che non erano peró fra gli obiettivi i questo corso.

- Una seconda limitazione del sistema é dovuta alla simulazione del frigo e della dispensa. Non essendo in possesso fisicamente degli elettrodomestici, questi sono simulati da due interfacce utente in cui é possibile aggiungere o rimuovere ingredienti. Durante queste azioni é necessario inserire anche il tipo dell'ingrediente, cosí come definito all'interno dell'ontologia in modo da non costringere il sistema a normalizzare anche questi ingredienti. Questa é una limitazione di cui soffre solamente il primo prototipo poiché, se si fosse in possesso del frigo e della dispensa intelligenti con cui il sistema finale dovrebbe lavorare, sarebbero i sistemi implementati al loro interno (ad esempio lettori di tag RFID applicati sugli ingredienti e contenenti giá informazioni normalizzate in base all'ontologia) a svolgere questa funzione.

- L'ontologia realizzata contiene un sottoinsieme limitato sia della gerarchia di tutti gli alimenti che soprattutto delle istanze delle classi definite, in quanto coprire tutti gli ingredienti possibili richiederebbe un lavoro minuzioso e molto dispendioso in termini temporali. Inoltre sarebbe possibile estendere tale ontologia, o affiancarne una nuova, con i termini in altre lingue in modo da poter riconoscere e ricercare ricette anche all'interno di pagine web non italiane.

- Un'ultima premessa riguarda la sintassi con cui vengono specificati i nomi degli ingredienti sulle ricette. Uno stesso ingredienti puó essere specificato in svariati modi eterogenei fra loro e questo causa diverse difficoltá nel riconoscimento automatizzato. Il sistema Smart-Kitchen si comporta bene in alcuni casi mentre in altri non riesce a riconoscere l'ingrediente esatto. Ad esempio la polpa di pomodoro se scritta in questo modo verrá riconosciuta correttamente mentre se specificata come pomodoro in polpa non verrà trovata nell'ontologia, ma considerata semplicemente pomodoro. Questo problema riguarda tutti gli ingredienti riguardanti parti di alimenti che saranno riconosciuti solamente se il nome sulla ricetta comincerá con il nome della parte.

3.2 Tool

Per la realizzazione dell'intero sistema é stato necessario integrare e lavorare con diverse tecnologie contemporaneamente. Di seguito viene riportato un elenco dei tool necessari alla realizzazione di SmartKitche.

- **Protégé:** Protégé é una piattaforma open source che fornisce una suite di strumenti per costruireapplicazioni basate su modelli di dominio e conoscenza sfruttando le ontologie. Al suo interno, Protégé supporta la creazione, la visualizzazione e la manipolazione di ontologie in vari formati di rappresentazione. Mediante la creazione di ontologie é possibile descrivere concetti e relazioni importanti in un determinato dominio applicativo, fornendo un vocabolario specifico per tale dominio utilizzabile come un dizionario contenete anche concetti semantici che il vocabolario stesso intende descrivere. All'interno del sistema SmartKitchen l'ontologia creata mediante il tool Protégé realizza l'unica fonte di conoscenza di base.

- **Jason:** é un interprete per la versione estesa di AgentSpeak. Implementa la semantica operazionale di tale linguaggio e fornisce una piattaforma per lo sviluppo di sistemi multi agente. All'interno del sistema Jason é utilizzato per la creazione degli agenti, e per la realizzazione della comunicazione tra essi.

- **Cartago:** é un'infrastruttura che realizza un ponte tra agenti creati sulla piattaforma Jason e artefatti realizzati in Java. Mediante Cartago é possibile quindi definire artefatti Java, che mettono a disposizione una serie di servizi, sotto forma di semplici metodi, utilizzabili dagli agenti Jason mediante una chiamata a metodo.

- **JENA:** Semantic Web Framework open source, utile alla costruzione di applicazioni basate sul web semantico, offre un ambiente programmatico per RDF, RDFS, OWL e SPARQL ed include un motore inferenziale rule-based.

- **SPARQL:** é un linguaggio di query per RDF. IRDF sono una delle forme piú comuni per la rappresentazione dell'informazioni nel web. La sintassi del linguaggio di query SPARQL é definita proprio dalle specifiche RDF. Questo linguaggio é utilizzato per interrogare la base di conoscenza in possesso del sistema SmartKitchen, ovvero l'ontologia.

- **JSoup:** é una libreria Java che consente di lavorare con file HTML. Esso fornisce API per l'estrazione e la manipolazione di dati. Queste librerie sono utilizzare per filtrare le informazioni presenti all'interno delle pagine contenti le ricette, in particolare vengono utilizzate per estrapolare le informazioni necessarie al

sistema per la rappresentazione di una ricetta (titolo, ingredienti, tempo e preparazione).

- **Google AJAX API:** mediante queste librerie Google fornisce un servizio che consente di effettuare ricerche online direttamente dalla propria applicazione. Le ricerche sono limitate peró ad un motore di ricerca custom.

3.3 Ricerca di ricette online

In questa sezione si descrivono piú dettagliatamente gli elementi che permettono al sistema di effettuare la ricerca di ricette all'interno delle pagine web del sito gustissimo.it. In particolare viene utilizzato l'artefatto WebRecipeBook che rappresenta l'interfaccia che il sistema ha verso il ricettario on-line e gli agenti webSearchRequestDispatcher e researcherAgent che utilizzano tale artefatto per effettuare ricerche e gestiscono le informazioni che questo restituisce in modo che il sistema possa utilizzarle al meglio.

3.3.1 Artefatto WebRecipeBook

L'artefatto WebRecipeBook é l'elemento che abilita il sistema alle ricerche di ricette on-line. La sua interfaccia é costituita da un'unica operazione searchRecipes. Gli agenti che vogliono effettuare la ricerca on-line usano questa operazione passando in input la query di ricerca e aspettandosi come risultato la prima ricetta trovata dall'artefatto (nel caso la query non produca risultati verrá restituito un messaggio di errore). Successivamente, l'artefatto continua la ricerca, grazie ad un'operazione interna, notificando all'agente i nuovi risultati attraverso l'utilizzo della primitiva signal. Si é scelto questo modello implementativo per fare in modo che l'agente rimanga in attesa solo del primo risultato e successivamente possa essere reattivo all'arrivo e alla gestione di nuovi risultati. La ricerca on-line, infatti, potrebbe richiedere tempi anche lunghi e un'attesa bloccante in questo caso sarebbe dannosa per il sistema. All'interno delle operazioni descritte, si sfruttano le google-ajax-apis (open source) per effettuare la ricerca, la quale restituisce una serie di link cosí come una ricerca eseguita sul motore web di google. A questo punto, per ogni link, é necessario fare il parsing della pagina html corrispondente, estrapolando le sole sezioni relative alla ricetta descritta (se la pagina contiene una ricetta). A tale scopo si utilizza un altro tool, jsoup, che permette

appunto di fare il parsing e di filtrare le informazioni contenute nelle pagine html. Terminata questa fase, le ricette (con le sezioni titolo, ingredienti, tempo, immagine e preparazione giá suddivise) possono essere passate all'agente che aveva richiesto la ricerca come descritto in precedenza.

3.3.2 Agenti webSearchRequestDispatcher e researcherAgent

La gestione delle ricerche di ricette on-line è gestita, all'interno del sistema da due agenti:

- **webSearchRequestDispatcher:** questo agente, unico all'interno del sistema, si occupa di inoltrare le richieste al researcherAgent associato all'utente che sta facendo la richiesta di ricerca. In particolare all'interno di Smart-Kitchen viene mantenuto un agente researcherAgent e un centralAgent per ogni utente che richiede una ricerca online, in modo da permettere a piú utenti di utilizzare il sistema contemporaneamente e che i risultati vengano processati da un unico agente e proposti nella GUI corretta. Il Dispatcher ha quindi lo scopo di generare un nuovo researcherAgent nel caso in cui sia la prima ricerca di un nuovo utente (cioé di un nuovo centralAgent) e mantenere le associazioni fra centralAgent e researcherAgent in modo da poter inoltrare le richieste di ricerca nel modo corretto (i risultati trovati researcherAgent verranno restituiti direttamente al centralAgent senza passare dal Dispatcher). Lo stesso meccanismo é utilizzato sia per le nuove ricerche che per l'espansione dei termini di ricerca indicati in precedenza.

- **researcherAgent:** L'agente researcherAgent ha come scopo principale quello di gestire le richieste di ricerche on-line provenienti dal centralAgent, e quindi dall'utente, utilizzando in modo opportuno l'artefatto WebRecipeBook ottenendo i risultati, per poi restituirli, strutturati in modo da facilitarne l'utilizzo e la manipolazione, al centralAgent richiedente. Cosí come introdotto nel punto precedente, il sistema sará composto da tanti researcherAgent quanti sono gli utenti (le GUI di ricerca) che abbiano richiesto una ricerca di ricette on-line. Questi agenti sono creati dal webSearchRequestDispatcher nel momento in cui un nuovo utente richiede una ricerca e, da quel momento, rimangono associati al proprio utente attraverso il centralAgent

relativo. Un'altra particolaritá di questo agente é la modalitá di utilizzo dell'artefatto WebRecipeBook. Questo artefatto, descritto nel paragrafo precedente, ha la possibilitá di ricercare in base ad una sola interrogazione in quanto, una volta cominciata la ricerca, esso rimane impegnato a cercare nuovi risultati senza mai terminare il suo lavoro (o comunque se la query produce un numero importante di risultati la ricerca potrebbe essere molto dispendiosa in termini temporali). Per questo motivo vengono creati dal researcherAgent piú artefatti WebRecipeBook corrispondenti a tutte le query espanse, grazie all'utilizzo dell'ontologia e al meccanismo di espansione della ricerca che sará descritto in seguito, a partire dall'interrogazione inserita dall'utente, in modo che possano lavorare tutti in parallelo per produrre dei risultati di buona qualitá. Nel momento in cui l'utente inserisce una nuova query di ricerca, tutti gli artefatti creati per l'interrogazione precedente vengono distrutti, in quanto ormai inutili, e si comincia la costruzione di quella relativi alla nuova ricerca, con un meccanismo identico a quello descritto in precedenza. Come detto, le ricette prodotte dagli artefatti, sono notificate ai researcherAgent attraverso delle signal; l'agente provvede poi a trasformarle, strutturandole attraverso termini prolog, e a restituirle con questa nuova forma di piú facile manipolazione, al centralAgent relativo all'utente.

3.4 Simulazione frigorifero e dispensa

La simulazione degli elementi della cucina con i quali SmartKitchen comunica, ovvero il frigorifero e la dispensa, é essenziale ai fini della manipolazione della conoscenza del sistema, che é rappresentata dagli ingredienti e dai tool per cucina a disposizione dell'utente. Nel caso specifico la simulazione dei due elettrodomestici consente di avere una variazione degli ingredienti dinamica che permette quindi di testare anche la reattivitá del sistema ai cambiamenti dell'ambiente circostante. Come verrá meglio illustrato di seguito é stato deciso di realizzare tale simulazione mediante interfacce grafiche, implementate all'interno di artefatti Java, che consentiranno all'utente di aggiungere, e/o rimuovere, in maniera del tutto asincrona gli elementi all'interno della cucina.

3.4.1 Artefatti GUI FridgeSimulatorView e PantrySimulatorView

Questi due artefatti, del tutto equivalenti dal punti di vista concettuale, e molto simili anche graficamente, rappresentano una interfaccia grafica che consente la manipolazione, degli ingredienti e dei tool da cucina, semplice ed intuitiva. Pensando ad un caso reale, queste due interfacce potrebbero essere presenti in ogni frigorifero o dispensa reale per consentire l'interazione fra l'utilizzatore e l'elettrodomestico stesso. Esse esibiscono una serie di campi, che l'utente deve necessariamente compilare in fase di inserimento (nome, tipo, marca, quantitá e data di scadenza dell'elemento), pulsanti per la conferma di inserzione o rimozione elemento e un'area che visualizza un sorta di summary degli ingredienti o dei tool disponibili. I metodi che queste due interfacce mettono a disposizione, sono quindi, metodi che simulano il comportamento di un eventuale utilizzatore che usa la cucina quotidianamente, quindi é previsto solamente la rimozione o l'aggiunta di elementi dalla cucina. Ogni artefatto grafico si appoggia ad un agente, al quale viene inviata una signal che rappresenta il tipo di evento percepito dalla GUI.

3.4.2 Agenti fridgeSimulator e pantrySimulator

Questi agenti sono associati sempre alle relative GUI. Il primo gestisce la GUI del frigorifero, e percepisce due tipi di eventi:

- **addOK:** quanto l'utente, dopo aver compilato tutti i campi relativi ad un ingrediente da inserire (nome, tipo, marca, quantitá, unitá di misura e data di scadenza), ha confermato l'operazione.
- **removeOK:** quanto l'utente, dopo aver selezionato l'ingrediente da eliminare, ho confermato l'operazione.

Per il secondo, che gestisce la GUI della dispensa, la tipologia di eventi é del tutto equivalente, con l'unica differenza che esso deve distinguere tra ingredienti e tool da cucina. A fronte di uno qualsiasi di questi eventi gli agenti interagiscono con altri due artefatti, Fridge e Pantry, i quali rappresentano la simulazione non piú dell'interfaccia grafica dei due elettrodomestici, bensí, dell'elettrodomestico vero e proprio, in grado quindi di mantenere al suo interno una lista degli elementi che contiene.

Figura 3.1: *Da sinistra, GUI di inserimento ingredienti del frigorifero, GUI della pantri durante una fase di rimozione ingrediente.*

3.4.3 Artefatti Fridge e Pantry

Questi due artefatti, come citato precedentemente, rappresentano concretamente i due elettrodomestici. Essi forniscono due servizi, addIngredient e removeIngredient utilizzati rispettivamente per l'inserimento e la rimozione di un elemento. I due servizi sono utilizzati dagli agenti che gestiscono le relative interfacce grafiche, e vengono richiamati alla pressione dei pulsanti di conferma operazione. Gli artefatti Fridge e Pantry memorizzano, inoltre, la lista degli elementi che contengono, e la mantengono sempre aggiornata in modo da rendere la

base di conoscenza del sistema in linea con lo stato dell'ambiente circostante. A tale scopo i due artefatti comunicano, mediante l'emissione di una signal con altri due agenti, Fridge e Pantry, che rappresentano una sorta di boundary tra gli elettrodomestici e il sistema. Essi dispongono anche di un altro servizio, messo a disposizione dai due artefatti, getIngredients, spiegato meglio nel paragrafo successivo.

3.4.4 Agenti Fridge e Pantry

Questi due agenti consentono il passaggio delle informazioni relative al contenuto del frigorifero e della dispensa all'interno del sistema SmartKitchen. In particolare, alla alloro creazione, viene richiesto, sia all'artefatto Pantry che al Fridge, una lista contenente tutti gli elementi in loro possesso. Da questo momento in poi la base di conoscenza del sistema é del tutto allineata con l'informazione contenuta dall'ambiente esterno. La richiesta della lista degli ingredienti avviene mediante il metodo getIngredients sul frigorifero, e getTools assieme con getIngredients sulla dispensa. A questo punto, i due agenti, possono memorizzarsi le informazioni pervenute dai due elettrodomestici e memorizzarle a loro volta localmente in una lista. Ogni qual volta verrá aggiunto o rimosso un ingrediente, gli agenti Fridge e Pantry verranno avvisati mediante l'emissione della signal newIngredient e removedIngredient da parte dei due artefatti Fridge e Pantry. Sfruttando tale meccanismo, la lista degli ingredienti e dei tool in possesso del sistema rimane sempre aggiornata. A fronte della ricezione di tali signal, i due agenti non vanno solo a modificare le liste in loro possesso, ma notificano a loro volta tale cambiamento al centralAgent (che potrebbe essere anche piú di uno), sfruttanto la primitica .send(destinatario,tell,informazione), dove per informazione si intende l'ingrediente o il tool aggiunto o rimosso. Tale comunicazione fa si che la base di conoscenza del centralAgent venga, costantemente mantenuta aggiornata.

3.5 Mapping semantico

In questo paragrafo sono descritti i componenti e le modalitá utilizzate in Smart-Kitchen per attribuire una semantica ai termini utilizzati all'interno del sistema, dagli ingredienti alle interrogazioni online. Questi meccanismi permettono al sistema di avere una precisione molto maggiore rispetto ad una elaborazione strettamente sintattica e di

fornire, in questo modo, dei servizi e delle funzionalitá mirate ed interessanti per l'utente. Il mapping semantico é realizzato da una serie di componenti strutturati con un'architettura a livelli. A livello piú alto, il componente direttamente utilizzabile a livello applicativo é il SemanticArtifact che offre funzionalitá per normalizzare gli ingredienti specificati in una ricetta, controllare che questi ingredienti siano presenti in cucina (e nel caso sia possibile fornire alternative a quelli mancanti) e ampliare l'interrogazione di ricerca. Questo artefatto utilizza i servizi messi a disposizione da altri due componenti: il primo é un parser grazie al quale gli ingredienti scritti all'interno delle ricette vengono separati e associati a tutte le informazioni specificate (nome, quantitá e unitá di misura), il secondo, l'OntologyArtifact, é l'artefatto di livello inferiore che fornisce servizi di base per l'accesso e l'interrogazione dell'ontologia. La struttura appena descritta é rappresentata in figura.

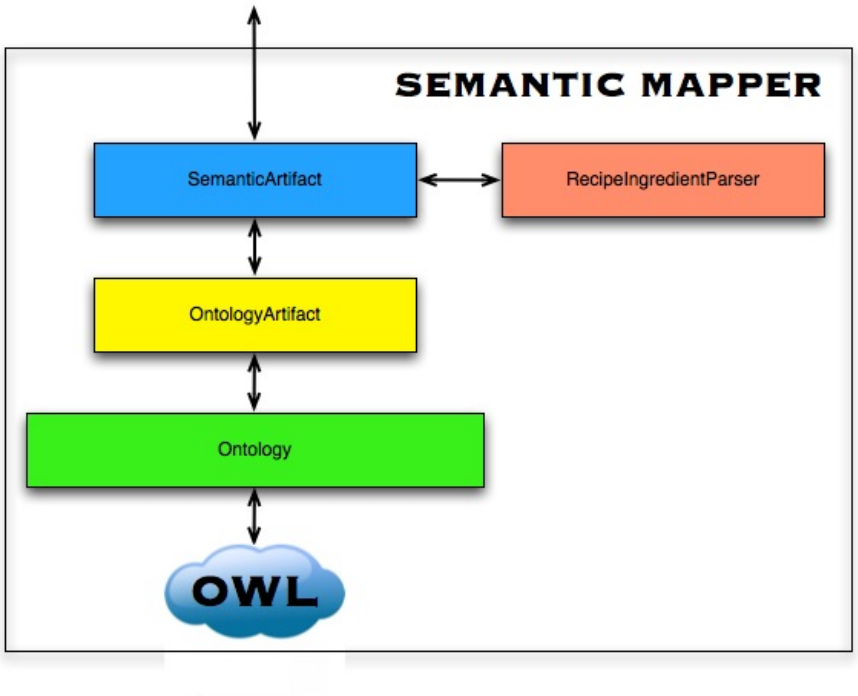


Figura 3.2: *Modulo di mapping semantico*

Ognuno dei componenti introdotti sará presentato piú in dettaglio nel seguito, associato alle operazioni svolte per il completamento delle singole funzionalitá fornite dal sistema.

3.5.1 Parsing ingredienti

Il primo componente approfondito all'interno del mapping semantico é il parser (RecipeIngredientParser). Questa entitá viene utilizzata dal SemanticArtifact come primo passo per effettuare la normalizzazione degli ingredienti presenti in una ricetta. La prima operazione da effettuare, infatti, é la separazione di ogni ingrediente e l'individuazione di tutte le informazioni che lo riguardano specificate nel testo estrapolato dalla ricetta. L'artefatto mette a disposizione un'unica operazione parseIngredients che, presa in ingresso la stringa che rappresenta la sezione di ricetta che descrive gli ingredienti, torna una lista di oggetti Recipeingredient che conterranno tutte le informazioni, dal nome alla quantitá richiesta, specificate nella stringa in ingresso per ogni ingrediente. Per svolgere il servizio descritto si realizza un parser che riconosca la grammatica mostrata di seguito, rappresentativa della descrizione degli ingredienti nel solo sito web preso in considerazione, e che come risultato ne costruisca l'Abstract Sintax Tree che sará come detto una lista di oggetti RecipeIngredient. La grammatica riconosciuto é la seguente:

```
List : [] | Ing '\n' List
Ing: Ingredient Rest
Rest: [] | '+' Ing
Ingredient: Quantity IngredientRest | Name
IngredientRest: UnitOfMeasure Name | Name
Name :  String
Quantity:  Dou
UnitOfMeasure: "kg." | "gr." | "ml." | "l." | "cucchiaio" |
               "bicchiere" | ....
```

Il parser realizzato é di tipo ricorsivo discendente, in cui ogni metodo interno é responsabile del parsing di un simbolo non terminale della grammatica e restituisce la parte di AST di sua competenza. Per fare un esempio possiamo considerare che la stringa in ingresso al parser contenga un solo solo ingrediente scritto nel modo seguente (il tutto si puó espandere alla presenza di piú ingredienti separati da un "line feed" in modo del tutto equivalente):

1 kg. di fettine di tacchino

Dal metodo principale si comincia il parsing del singolo ingrediente (Ing). Si richiama Ingredient che deve cominciare con una quantitá, cioè un double che nel caso in esame vale 1. Riconosciuta la quantitá, si torna al livello di Ingredient e si passa al successivo simbolo non terminale, ovvero IngredientRest. A questo punto la prima alternativa é la presenza di una unitá di misura, che nell'esempio é presente ed é "kg.". Parsata l'unitá, si passa obbligatoriamente al nome dell'ingrediente, che sará tutta la stringa rimanente ("di fettine di tacchino"). Terminato il riconoscimento, tornando verso la radice nello stack delle chiamate viene costruito l'oggetto RecipeIngredient inserendo al suo interno, come campi della classe, tutte le informazioni appena riconosciute. Se fossero stati presenti piú ingredienti sarebbe stata creata una lista di oggetti RecipeIngredient. Terminata questa fase la lista degli ingredienti con tutte le informazioni necessarie associate, é pronta per essere normalizzata e manipolata in tutte le successive fasi di lavoro del sistema.

3.5.2 Normalizzazione e check ingredienti

La fase di normalizzazione degli ingredienti, e successivamente di check, é una delle fasi piú importanti svolte all'interno del sistema. La prima mira ad associare ad un ingrediente, precedentemente parsato secondo la grammatica interna, una classe dell'ontologia, che rappresenta il concetto semantico che l'ingrediente incarna, e successivamente avviene anche la normalizzazione dell'unitá di misura in modo da rendere la quantitá degli ingredienti facilmente confrontabile tra di loro. La seconda, invece, consiste nel verificare se all'interno della cucina sono presenti, anche nella quantiá richiesta, tutti gli ingredienti necessari alal realizzazione della ricetta. Il confronto avviene, non utilizzando il nome vista come stringa sintattica dell'ingrediente, bensí sul concetto che esso esprime. Durante il check, avviene anche, per gli ingredienti mancanti, un controllo per proporre eventuali ingredienti alternativi.

- **Normalizzazione** : la normalizzazione degli ingredienti avviene all'interno del SemanticArtifact, in particolare nel metodo normalizeIngredients. In questa operazione, dopo aver eseguito il parsing, come descritto nel paragrafo precedente, si procede alla fase di normalizzazione vera e propria della lista di ingredienti tornata dal parser. Per ogni ingrediente si effettua per prima cosa la normalizzazione del tipo, in questo modo: si tokenizza il

nome non considerando eventuali parole poco significative come preposizioni e congiunzioni; partendo dal primo token si va a verificare, usando l'operazione normalizeName messa a disposizione dall'artefatto OntologyArtifact, se la stringa sia un'istanza di qualche classe dell'ontologia o se sia la sottostringa iniziale di qualche istanza di classe e, nei casi in cui si verifica questa condizione, ci si salva questa o queste classi (l'operazione infatti potrebbe tornare anche piú di un concetto nel caso in cui la stringa fosse una sottostringa di un'istanza di classe. Ad esempio "polpa_ " é la sottostringa iniziale di "polpa_pomodoro" ma anche "polpa_granchio" e "polpa_frutta"); a questo punto si concatena il token successivo del nome, separato con un '_', e si esegue la stessa operazione, sostituendo i concetti salvati con i nuovi eventuali restituiti con questa stringa maggiore della precedente; tale ciclo si ferma quando l'operazione normalizeName torna una lista di concetti vuota o i token del nome sono terminati; a questo punto si utilizza come tipo dell'ingrediente, andando a riempire il campo Type, l'ultima lista non vuota di concetti tornata dal ciclo descritto (se tutte le liste tornate sono vuote si usa "Unknown Concept"). Il secondo passaggio da effettuare é la normalizzazione dell'unitá di misura. Questa operazione é svolta dal metodo normalizeUnit dell'OntologyArtifact a cui vengono passati sia l'unitá e la quantitá specificati nella ricetta, sia il tipo dell'ingrediente, informazione utile per sapere se l'ingrediente é liquido o solido in modo da trasformare le unitá di conseguenza. All'interno di normalizeUnit, le quantitá vengono normalizzate tutte ad un'unica unitá di misura di riferimento in modo da permettere alla fase successiva di check di eseguire calcoli e confronti in modo agevole. Le modalitá con cui le unitá vengono trasformate all'interno del metodo sono descritte piú approfonditamente nei paragrafi 3.6.3 e 3.6.4 riguardanti l'ontologia.

- **Check** : questa fase viene realizzata tramite il metodo *checkIngForRecipe* all'interno della classe *SemanticsArtifact*. Al suo interno si é in possesso di tutti gli ingredienti, normalizzati, quindi dei quali é ben noto il tipo semantico, necessari alla realizzazione di una determinata ricetta. Il check consiste nel confrontare il tipo dell'ingrediente ricetto, con i tipi degli ingredienti presenti all'interno del frigorifero. Nel caso in cui si abbia un match, se ne controlla la quantitá, e nel caso in cui anch'essa sia maggiore

o uguale a quella richiesta, l'ingrediente viene considerato presente. Nel caso in cui l'ingrediente ricercato non sia presente in cucina, si controlla, se tra gli ingredienti presenti, ve ne sia qualcuno sotto classe di quello richiesto. In caso affermativo significa che nella cucina esiste un ingrediente appartenente alla classe semantica dell'ingrediente richiesto. Facciamo un esempio:
Tra gli ingredienti richiesti in una ricetta puó essere presente l'ingrediente *volatile*. É difficile che, all'interno di un frigorifero, sia presente un ingrediente etichettato come *volatile*. É molto piú probabile, invece, che si trovi un ingrediente del tipo *Pollo* o *Tacchino*, i quali rappresentano una sotto classe di volatile, e per questo sono accettati come ingredienti validi.
In tutti gli altri casi l'ingrediente viene etichettato come ingrediente mancante, e viene aggiunto nell'apposita lista che sará visualizzata successivamente all'utente nel caso in cui venissero visualizzati i dettagli della ricetta. Una volta che tutti gli ingredienti hanno passato la fase di check, e la lista degli eventuali ingredienti mancanti é stata correttamente riempita, si continua con la fase cosí detta *degli hints*. Essa avviene sempre all'interno del metodo *checkIngForRecipe*, ma viene realizzata concretamente all'interno della classe *OntologyArtifact* nel metodo *getHint*. Ad esso viene passata l'intera lista degli ingredienti mancanti, e per ognuno di essi vengono prelevati i fratelli, intesi come concetti simili a quello espresso dall'ingrediente. Vediamo un esempio: uno degli ingredienti mancanti per realizzare la ricetta é l'ingrediente *Pollo*. A questo punto vengono presi tutti gli ingredienti aventi la stessa super classe di *Pollo*, nel caso specifico tutte le sotto classi di *Volatile*. Il risultato é quello di avere un'insieme di ingredienti alternativi che rimangono nel dominio concettuale dell'ingrediente richiesto, nel nostro caso rimaniamo nell'ambito dei *Volatile*, e piú in generale nell'ambito di animali con carne bianca. Il risultato di tale operazione restituirá gli ingredienti *Tacchino* e *Anatra*. Se i due ingredienti saranno presenti nella cucina in quantitá sufficiente per la ricetta verranno proposti all'utente come valida alternativa all'ingrediente mancante *Pollo*.
Ovviamente la ricerca dei suggerimenti all'interno dell'ontologia é tanto piú buona quanto la gerarchizzazione dell'ontologia é costruita ad hoc per fornire tale servizio. Si potrebbe, in altre parole, costruire un'ontologia le cui classi rappresentino ingredienti simili, equiparabili, intercambiabili, piuttosto che una gerarchia volta alla rappresentazione delle tipologie di ingredienti.

3.5.3 Espansione ricerca online

L'espansione della ricerca, giá introdotta del capitolo 2, é utile al rafforzamento della ricerca delle ricette. Essa consente al sistema di aumentare l'entropia di informazioni da ricercare sul web, derivando, dalla semplice stringa inserita dall'utente, una serie di concetti ad essa vicini, che permettono, in alcuni casi di specializzare la ricerca introducendo termini piú mirati rispetto a quelli usati dall'utente, in altri casi, invece, consente di comprendere a quale concetto l'utente vuole riferirsi nel caso in cui esso stia usando una terminologia per la ricerca non propria del dominio applicativo. Gli elementi necessari per effettuare l'espansione della ricerca sono quelli visibili nell'immagine riportata sopra, andiamo ad analizzare piú nel dettaglio il flusso di lavoro e i meccanismi implementativi. Dopo che la stringa di ricerca inserita dall'utente viene recapita al centralAgent, esso si incarica di inviarla al semantic mapper per poter estrapolare eventuali informazioni utili al fine della ricerca della ricetta desiderata. Il primo artefatto che viene utilizzato é il *sematicArtifact*. Al suo interno é presente il metodo *expandSearch(String query, OpFeedbackParam¡LinkedList¡String¿¿ moreSearch)*, il quale, come é visibile dalla signature, vuole come parametri la stringa che rappresenta la query inserita dall'utente e restituisce i termini, estrapolati dall'ontologia, che verranno aggiunti alla ricerca. All'interno di tale metodo la stringa, dopo essere stata filtrata dei termini inutili come le congiunzioni, viene suddivisa in token, ognuno dei quali rappresenta un termine che verrá ricercato all'interno dell'ontologia del sistema. La query effettuata con tale termine potrá avere due esiti :

- il termine é una foglia del grafo gerarchico descritto all'interno dell'ontologia.
- il termine non viene ritrovato nell'ontologia.

Per controllare i due casi precedenti ci si appoggia alla classe *OntologyArtifact* richiamando il metodo *isIstance* il quale agisce direttamente sul file OWL dell'ontologia. Nel caso in cui il metodo ritorni positivamente sará necessario procedere concatenando un eventuale token successivo a quello corrente ed effettuare una nuova query, questo fino a quando non si ricade nel secondo caso, e i caratteri concatenati tra di loro mediante il simbolo _ non sono rappresentati da alcuna istanza all'interno dell'ontologia. Di seguito un esempio:

Stringa inserita dall'utente : *Volatile arrosto.*

- query1 = *volatile* : all'interno dell'ontologia esiste un'istanza che contiene come termine quello cercato. Si prosegue con la creazione della query2.

- query2 = *volatile_arrosto* : questa volta la stringa cercata all'interno dell'ontologia non é presente, nessuna istanza fa match con essa. A questo punto si retrocede di un passo, tornando al risultato ottenuto con la query immediatamente precedente a quella corrente, e se ne ricavano le super classi corrispondente, nel caso in questione *Volatile* .

Questo ultimo step viene eseguito sfruttando il metodo *findMoreSearch* all'interno della classe *OntologyArtifact*. In particolare al suo interno sono richiamati altri due metodi, definiti nella medesima classe, *getChild* e *getSynonyms*. Il primo consente di prelevare tutte le istanze figlie della super classe, ovvero concetti simili a quello espresso dalla stringa ricercata nell'ontologia, mentre il secondo preleva le istanze della super classe, che rappresentano quindi sinonimi per esprimere il medesimo concetto. Nel caso specifico i risultati di tale operazione sono :

- **Figli** : Anatra, Pollo e Tacchino.

- **Sinonimi** : Pennuto, Uccello.

Le stringhe sopra riportate vengono recapitare al *centralAgent*, il quale si preoccuperá di inviarle agli agenti responsabili della ricerca. In questo modo i risultati pervenuti all'utente saranno, non solo numericamente superiori rispetto ai soli pervenuti utilizzando la stringa iniziale, ma anche piú specifici. Da sottolineare il fatto che l'espansione della ricerca riesce a muoversi all'interno dello spazio concettuale desiderato dall'utente ricavando informazioni semantica dalla stringa da esso inserita.

3.6 Ontologia

É stato possibile ampliare le query di ricercare di un ingrediente grazie all'ontologia, questa viene utilizzata dal *CentralAgent* attraverso l'*OntologyArtifact*, il quale offre le funzionalità necessarie per il corretto utilizzo. La sua struttura principale comprende quattro macroclassi(sottoclassi di *Thing*):

- **Alimento**: é sicuramente la gerarchia più estesa e offre una classificazione di molti cibi, sia liquidi che solidi. Pensata in modo da identificare facilmente gli alimenti;
- **Equivalenza**: questa gerarchia comprende le classi di tutte le possibili equivalenze, sia che riguardino due unità di misura differenti sia che riguardino un utensile e una grandezza fisica;
- **Grandezza**: comprende le possibili grandezze fisiche riscontrabili in ricette;
- **Utensile**: gerarchia degli utensili che possono aver un significato di quantità in una ricetta.

Sono presenti anche numerose *Object Properties* e *Data Properties* utili per l'implementazione di alcune funzionalità, in particolare modo impiegate per ricondurre un utensile ad una quantità ed infine sono presenti gli *Individuals* che possiedono un ruolo molto importante.

3.6.1 Riconoscimento di un ingrediente o di un suo sinonimo

La ricerca del nome di un ingrediente non trova sempre riscontro diretto con ciò che è scritto nell'ontologia. Un piccolo cambiamento potrebbe non generare alcun match con le classi preesistenti e ciò non permetterebbe di eseguire ulteriori operazioni. Per questo è stato sviluppato un meccanismo, con al centro gli individui, per cercare di massimizzare il matching. Quando viene letto un ingrediente da una ricetta, si cerca una corrispondenza parziale o totale con il nome dell'ingrediente tra gli individui, sia che si trovi un match totale sia che si trovi solo un match parziale(e.g. *polpa di . . . , fegato di . . . , patat. . .*) sará possibile classificare l'ingrediente. É possibile che siano presenti più individui appartenenti alla stessa classe, il sistema è quindi in grado di ricondurre sinonimi(o nomi plurali) di ingredienti alla stessa categoria.

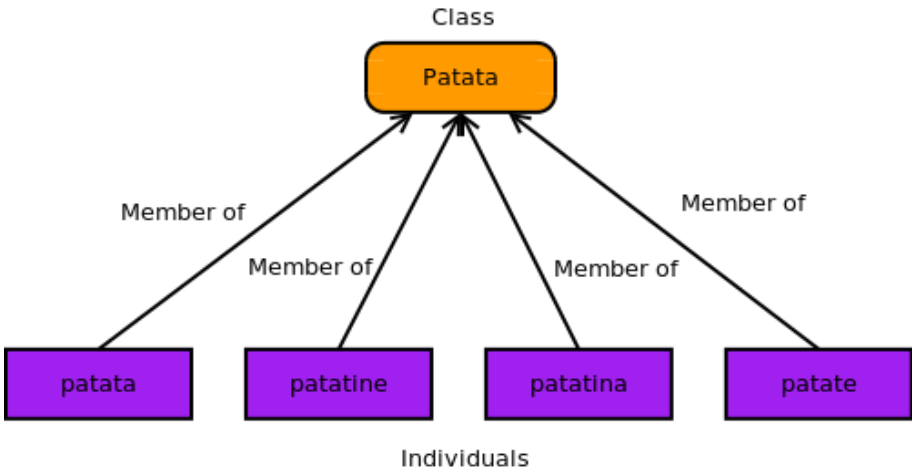


Figura 3.3: *Individui della classe Patata.*

Questo meccanismo é facilmente espandibile, basta infatti la semplice aggiunta di un ingrediente come individuo membro di una certa classe per aumentare la conoscenza dell'ontologia.

3.6.2 Parti di Alimento

Durante lo studio dei ricettari si é notato come molti alimenti non vengano usati nella loro interezza, e questo ha portato ad alcune problematiche. Per esempio nella ricetta *Petto di pollo ripieno* viene usato *1 petto di pollo*, supponendo di avere un *1 pollo* intero all'interno del Fridge, la ricerca dell'ingrediente *"petto di pollo"* non porterebbe alcun risultato. Rimane ovvio il fatto che non sia corretto, in quanto se abbiamo a disposizione un pollo intero si dispone anche di: 1 petto di pollo, 2 ali di pollo, 2 coscie di pollo, etc. Una prima possibile soluzione portava le parti di un alimento come sottoclassi dell'alimento stesso(e.g. PettoDiPollo subClassOf Pollo, etc.), ma ció é semanticamente sbagliato in quanto le parti non sono specializzazione della classe in questione. Si è quindi dovuta sviluppare una soluzione alternativa, é stata realizzata una classe *PartiDiAlimento* (sottoclasse di *Alimento*), nella quale vengono racchiuse eventuali parti degli alimenti classificati, insieme a due *Object Properties*(*prodottoDa, usatoIn*) che mettono in relazioni le parti e i relativi alimenti completi.

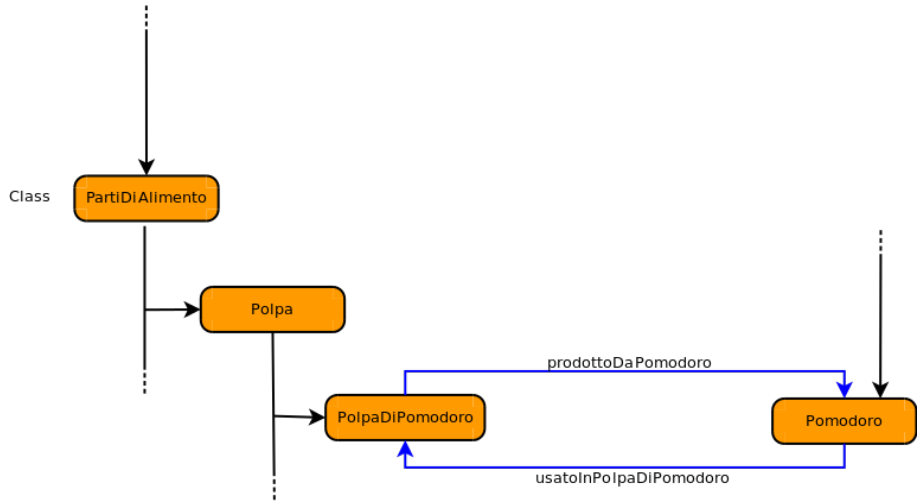


Figura 3.4: *Parti di alimento.*

3.6.3 Grandezze Fisiche

La ricerca di una ricetta viene effettuata sopratutto per venire a conoscenza di quali ingredienti servono e in quali quantitá. Nella classe *Grandezza* troviamo una classificazione delle piú comuni unitá di misura che possono trovarsi nei ricettari, in questo modo é possibile indentificare sia l'ingrediente sia la quantitá con relativa unitdi misura, sará quindi possibile effettuare conversioni di unitá (e.g. from *millilitro* to *litro*) per avere misure normalizzate, utili ad una stima omogenea e unificata delle quantità. Infatti in una ricetta è possibile trovare ingredienti diversi con diverse unità di misura, grazie ai meccanismi descritti nella sezione seguente delle *Equivalenze* é possibile ottenere gli ingredienti con le unitá di misura normalizzate. Ovviamente nel caso queste sia riconducibili alla stessa unitá normalizzata, per esempio tutte le misure di volume verranno convertite in *Litro*, quelle di peso in *Kilogrammo* e cosí via.

3.6.4 Equivalenze

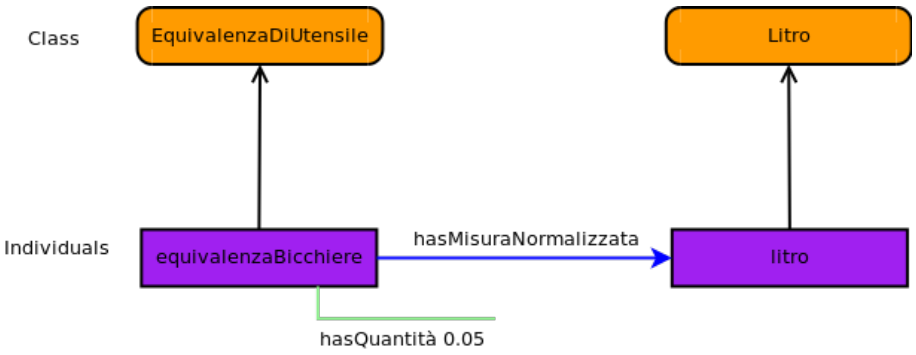
Negli ingredienti di molte ricette si trovano alcune espressioni che non possono essere direttamente ricondotte ad unitá di misura(e.g. *"un cucchiaio di ..."*, *"un bicchiere di ..."*, etc.), inoltre puó capitare che alcuni ingredienti abbiano divese unitá di misura per esprimere la stessa grandezza fisica(vedi sezione precedente[link]). Queste due ricorrenze comportano non poche problematiche, sono stati quindi svi-

Normalizzazione delle Grandezze	
Non Normalizzate	**Normalizzate**
Grammo	Kilogrammo
Decilitro Centilitro Millilitro	Litro

Figura 3.5: *Normalizzazione delle grandezze.*

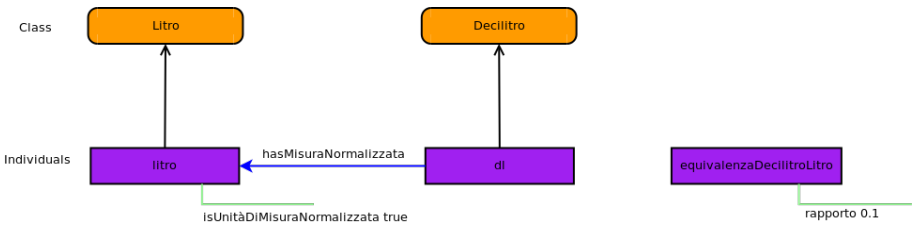
luppati i meccanismi di equivalenze con i quali è possibile ottenere la lista degli ingredienti in unitá di misura normalizzate.

La classe *EquivalenzaDiUtensile*(sottoclasse di *Equivalenza*) definisce una serie di membri grazie ai quali un utensile viene ricondotto ad un determinata quantitá con relativa unitá di misura normalizzata. Si prenda in analisi l'individuo *equivalenzaCucchiaioGrande*, questo possiede due proprietá: la proprietà di oggetto *hasMisuraNormalizzata* serve a definire in quale unitá di misura verrá tradotta la quantitá (indicativa dell'utensile) memorizzata all'interno della proprietá di dato *hasQuantitá*. Dopo aver definito tale struttura, l'*OntologyArtifact* discrimina se un'unitá di misura é normalizzata, non verranno quindi eseguite operazioni, se l'unitá di misura é da normalizzare(vedi paragrafo seguente) oppure se é un un utensile(e.g. *Bicchiere*), in questo caso verifica se é presente un individuo di equivalenza(e.g. *equivalenzaBicchiere*) ed esegue le operazioni di conversione. Legge la proprietá *hasQuantitá*, la moltiplica per il numero degli utensili e crea un nuovo oggetto con unità e quantità normalizzate.



Equivalenza di utensili.

In ultima analisi rimane il caso in cui l'unitá di misura non sia normalizzata. Alla base della struttura per la normalizzazione ci sono gli individui appartenenti alla classe *EquivalenzaUnitàDiMisura*(sottoclasse di *Equivalenza*), con la proprietá di dato *rapporto* e gli individui delle classi d'unitá di misura, che possono avere o la proprietá di dato *isUnitáDiMisuraNormalizzata* o la proprietá di oggetto *hasMisuraNormalizzata* che contiene il riferimento all'unitá normalizzata. Supponiamo che l'*OntologyArtifact* debba elaborare un'unitá di misura da normalizzare, per esempio *"20 dl"*, questo procede come segue: cerca un individuo che abbia una corrispondenza con il nome *"dl"*, trovato l'individuo controlla se questo possiede il campo *isUnitáDiMisuraNormalizzata* e quindi non necessita di ulteriori operazioni, oppure il campo *hasMisuraNormalizzata* come nel caso in analisi. Questo campo contiene il riferimento all'individuo(nell'esempio *litro*) che rappresenta l'unità normalizzata *Litro*. Ora rimane da analizzare se esiste un individuo che identifichi un'equivalenza di unitá di misura, l'*OntologyArtifact* quindi effettua una ricerca di *"equivalenza + classe di dl + classe di litro"*, se presente questo individuo conterrà la proprietà *rapporto* contenente il valore con cui effettuare la conversione, *0.1*. L'artefatto conclude l'intero processo costruendo un oggetto con unità di misura *Litro* e quantità normalizzata(20 * 0.1).



Equivalenza di unitá di misura.

3.7 Interfaccia utente

L'interfaccia utente é l'unico mezzo grazie al quale l'utilizzatore puó sfruttare tutte le funzionalitá del sistema. Grazie ad essa, infatti, é possibile effettuare ricerche di ricette online, mediante l'inserzione di una stringa che identifica ció che si desidera trovare, é ovviamente possibile avere la lista delle ricette che corrispondono ai nostri criteri di ricerca e infine é possibile anche consultare i dettagli di ogni ricetta nel caso in cui si desideri preparare il piatto. Di seguito vengono descritti pi´ù nel dettaglio gli artefatti e gli agenti che contribuiscono alla realizzazione e al funzionamento di tale interfaccia.

3.7.1 Artefatto SearchGUI

All'interno dell'artefatto SearchGUI viene implementata concretamente l'interfaccia con la quale l'utente interagisce durante l'uso del sistema. Tale interfaccia é suddivisa in tre parti principali:

- **Zona di ricerca**: composta da una text area all'interno della quale specificare il criterio di ricerca della ricetta, e un pulsante per confermare ed iniziare la ricerca online.
- **Zona titoli ricette**: che contiene tutti i titoli delle ricette trovate.
- **Zona dettagli ricetta**: all'interno della quale vengono visualizzati tutti i dettagli della ricetta selezionata, a partire dal tempo di preparazione, ingredienti necessari, ingredienti mancanti e lavorazione.
- **Zona immagine**: all'interno della quale compare una piccola immagine che rappresenta il risultato finale della ricetta che l'utente sta consultando in quel momento.

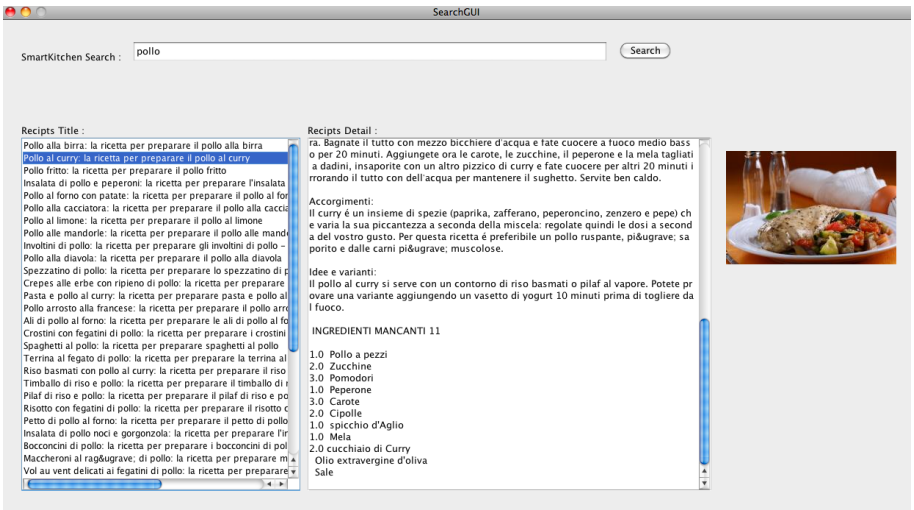


Figura 3.6: *Interfaccia utente durante la ricerca di una ricetta con ingrediente principale il pollo.*

Questo artefatto si preoccupa essenzialmente di ricevere le ricette recapitategli dal sistema e visualizzarle correttamente all'interno di ogni sezione, e di inviare la stringa di ricerca al sistema ogni qual volta

l'utente desidera effettuare una nuova ricerca online. Quest'ultimo evento viene segnalato direttamente al proprio agente searchGUIAgent mediante l'emissione di una signal newSearch alla quale é associata la stringa di ricerca inserita dall'utente.

3.7.2 Agente searchGUIAgent

Questo agente comunica non solo ,come detto in precedenza, con la SearchGUI, ma anche con il centralAgent. In particolare ogni volta che viene emessa una signal riguardante una nuova ricerca da effettuare, esso cattura l'evento e invia al centralAgent la stringa che l'utente desidera cercare. Viceversa, quando il centralAgent é in possesso delle ricette trovare, esse vengono recapitare al searchGUIAgente mediante il predicato recipe(Title,Ingredients,Time,Instructions,MissingTerm,Image). Le ricette saranno poi comunicate alla GUI che sará responsabile della vsualizzazione.

3.8 Sistema finale

L'immagine seguente mostra quali sono, all'interno del sistema SmartKitchen, gli elementi fondamentali che lo realizzano. Lo schema si limita a mostrare i blocchi concettuali, senza soffermarsi sui dettagli.

Come é ben visibile, il *centralAgent* ha un ruolo centrale e fondamentale all'interno del sistema, e funge da perno per tutte le comunicazione tra agenti e artefatti che vengono realizzate. Di seguito si vuole mettere in evidenza l'importanza di tale componente, elencandone le responsabilitá e gli scenari nei quali esso viene tirato in ballo.

- **Inizio di una ricerca online**:
  - Ricezione della stringa da ricercare con successivo inoltro al *RequestDispatcher* per fare avviare la ricerca online.
  - Richiesta al frigorifero e alla dispensa della lista di tutti gli ingredienti e i tool presenti nella cucina. Tale lista verrá salvata localmente e mantenuta aggiornata durante tutta la vita dell'agente.
- **Espansione della ricerca**:
  - Invio della stringa inserita dell'utente al semantic mapper.

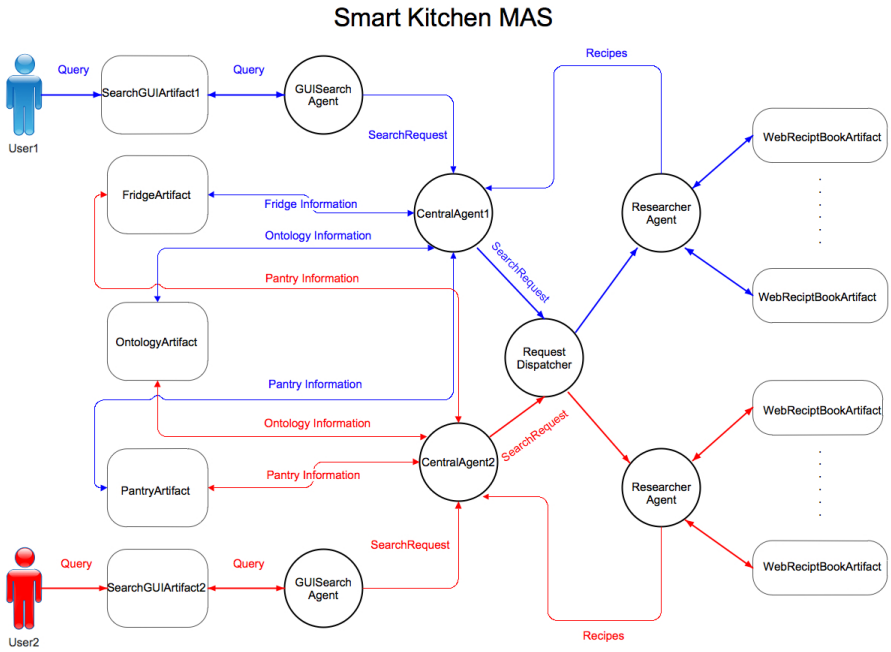


Figura 3.7: *Schema concettuale del funzionamento dell'intero sistema SmartKitchen.*

  - Ricezione dei nuovi termini estrapolati dall'ontologia e relativo invio al *RequestDispatcher*.

- **Normalizzazione ingredienti**:
  - Ricezione delle ricette trovate online, con relativa memorizzazione in locale.
  - Prelievo della lista degli ingredienti necessari alla realizzazione di ogni singola ricetta.
  - Invio della lista al semantic mapper e attesa della nuova lista degli ingredienti conformi al linguaggio interno del sistema.

- **Check ingredienti**:
  - Dopo la fase di normalizzazione, invia la lista degli ingredienti nuovamente al semantic mapper.
  - Ricezione della lista di eventuali ingredienti mancanti con relativi ingredienti di suggerimento.

  - Impacchetta la ricetta, completa di tutte le informazioni (titolo, tempo, ingredienti, preparazione, ingredienti mancanti, suggerimenti e immagine), e la spedisce alla GUI per la visualizzazione.

- **Modifica degli ingredienti nella cucina durante una ricerca**:

  - Ricezione dell'ingrediente aggiunto o rimosso dalla cucina con conseguente aggiornamento della lista locale.
  - Nuova esecuzione della fase di check degli ingredienti delle ricette fino a quel momento memorizzate in locale.
  - Invio di tutte le ricette alla GUI.

3.8.1 Conclusioni

Nel complesso il sistema realizza tutte le funzionalitá desiderate, unica pecca é rappresentata dalle prestazioni non troppo elevate, e dalla difficoltá di connessine con il motore di ricerca custum di google. In futuro il sistema potrebbe essere ampliato e migliorato sviluppando maggiormente le seguenti parti:

- **Ricerca online**: aumentare il numero di siti internet consultabili per le ricette. Questo comporta un aumento della capacitá di parsing sia delle pagine web, sia della gramamtica usata per descrivere gli ingredienti. In qesta fase sarebbe anche possibile inserire algoritmi di DataMining per la ricerca piú mirata delle informazioni.

- **Memorizzazione ricette**: supportare il sistema di un meccanismo per la memorizzazione delle ricette che consenta una consultazione rapida e basata sulla semantica del contenuto.

- **Filtri di ricerca**: aumentare le informazioni che consentono di filtrare le ricette a seconda dei gusti, delle necessitá e delle abitudini dell'utente.

Bibliografia

[1] aricci, asanti: *cartago-by-examples, env-prog-cartago-tech-rep.* 2010

[2] Rafael H. Bordini1 and Jomi F. H ubner2: *Jason: A Java-based interpreter for an extended version of AgentSpeak.*

[3] Wiley: *Programming Multi Agent Systems in AgentSpeak using Jason* Dec 2007.

[4] Andrea Omicini Michele Piunti: *Programming Intentional Agents in AgentSpeak(L) and Jason.*

[5] Elena Nardini: *Jason Agents in CArtAgO Working Environments.*