Progetto di Ingegneria del Software 2

SWIMv2: Small World Hypothesis Machine v2

Autori: Professore:   
Bulla Jacopo Mottola Luca  
Caio Davide  
Cappa Stefano

A differenza del Project Planning, questo documento mostra più nel dettaglio le scelte di progettazione da parte del team di sviluppo. Lo scopo non è solo di fornire i modelli realizzati, ma mostrare l’intero processo di analisi dei requisiti, iniziando dalla scelta degli attori per definire gli scenari ed ottenere i casi d’uso. Da essi sono stati costruiti i Class, Sequence e BPMN Diagrams, per concludere con le specifiche Alloy.

Questo documento rappresenta la seconda deliverable. Insieme ad esso è stato fornito anche il Project Planning aggiornato, a seguito di un incontro coi responsabili del progetto in data 09-11-2012.

Indice

[1. Introduzione 4](#_Toc342524421)

[1.1. Errata Corrige 4](#_Toc342524422)

[1.2. Scopo 4](#_Toc342524423)

[1.3. Dominio applicativo 4](#_Toc342524424)

[1.4. Definizioni, acronimi e abbreviazioni 5](#_Toc342524425)

[1.5. Documenti di riferimento 6](#_Toc342524426)

[1.6. Panoramica 6](#_Toc342524427)

[2. Descrizione generale 7](#_Toc342524428)

[2.1. Funzioni del prodotto 7](#_Toc342524429)

[2.2. Caratteristiche degli utenti 8](#_Toc342524430)

[2.3. Assunzioni 8](#_Toc342524431)

[3. Requisiti, Diagrammi UML e BPMN 10](#_Toc342524432)

[3.1. Requisiti Funzionali 10](#_Toc342524433)

[3.2. Use Case Diagram 18](#_Toc342524434)

[3.3. Sequence Diagram 19](#_Toc342524435)

[3.4. BMPN Diagram 26](#_Toc342524436)

[3.5. Class Diagram 34](#_Toc342524437)

[4. Requisiti non funzionali 36](#_Toc342524438)

[4.1. Interfacce utente 36](#_Toc342524439)

[4.2. Interfacce software 36](#_Toc342524440)

[4.3. Interfacce hardware 36](#_Toc342524441)

[4.4. Interfacce di comunicazione 36](#_Toc342524442)

[4.5. Vincoli 37](#_Toc342524443)

[4.6. Scelte adottate dal team di sviluppo 37](#_Toc342524444)

[5. Alloy 38](#_Toc342524445)

[5.1. Rappresentazione del mondo in Alloy 38](#_Toc342524446)

[5.2. Rappresentazione dei vincoli 39](#_Toc342524447)

[5.3. Asserzioni 40](#_Toc342524448)

[5.4. Istanza 42](#_Toc342524449)

[Indice delle figure 44](#_Toc342524450)

# Introduzione

## Errata Corrige

Dopo al colloquio del 9-11-2012 presso la sede della società che ha commissionato il progetto, è stato corretto il documento di Project Planning (aggiornandolo alla versione 1.1).

Successivamente, in seguito ad alcune modifiche effettuate in fase di design è stato aumentato l’effort nella fase di revisione.

A causa di alcune ambiguità nel documento fornito dal committente, sono stati commessi alcuni errori nello stabilire le funzionalità del sistema. Per tale motivo, è stata corretta l’introduzione del capitolo 1 a cui è stato aggiunto un glossario, per spiegare meglio alcuni dei termini che saranno usati anche nel RASD.

## Scopo

Lo scopo di questo documento, chiamato RASD (Requirments Analysis Specification Document), è l’analisi dei requisiti e delle specifiche del progetto SWIMv2.

La panoramica generale è fornita tramite descrizioni e attraverso la definizione di alcuni “scenari”. Da essi sono realizzati alcuni modelli standardizzati in UML (Use Case Diagram, Activity Diagram, Sequence Diagram, Class Diagram), a cui sarà allegata anche la specifica Alloy.

## Dominio applicativo

Il dominio applicativo è l’ambiente in cui l’applicazione s’interfaccerà.

E’ stato rappresentato tramite l’approccio di Jackson-Zave, il quale suddivide il dominio in “World” (Mondo), “Machine” (Applicazione) e “Shared Phenomena” (Fenomeni condivisi).

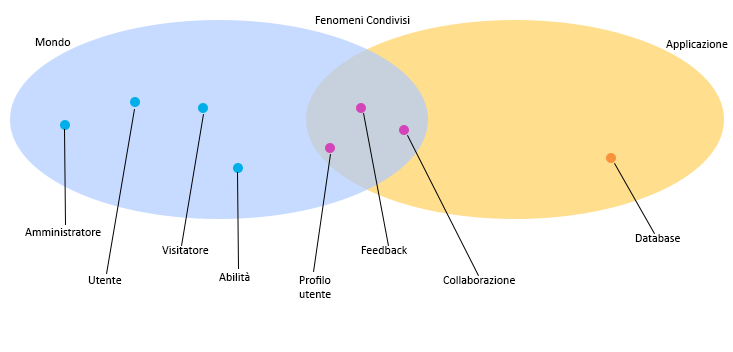


Fig. 1.1 – Diagramma di Jackson-Zave

1. Mondo: Amministratore, Utente, Visitatore, Abilità
2. Fenomeni condivisi: Profilo utente, Feedback, Collaborazione
3. Applicazione: Database

## Definizioni, acronimi e abbreviazioni

In tutto il documento saranno utilizzati i seguenti termini:

* **Visitatore: utente non registrato, cioè accede alla piattaforma senza inserire dati personali e registrarsi e di conseguenza non può utilizzare le funzionalità del sistema (a parte la ricerca per i visitatori**).
* **Utente**: utilizzatore registrato di SWIMv2 che ha creato un “profilo“ confermato dall’Amministratore.
* Profilo: pagina personale di un Utente in cui vi sono informazioni personali.
* Pagina di amministrazione: pagina web accessibile solo dall’Amministratore di SWIMv2 e da nessun altro Utente o Visitatore. In essa vi sono tutte le funzioni destinate alla gestione del social network.
* **Email**: identificativo univoco di un Utente del sistema.
* **Password**: parola segreta, che abbinata all’email, permette di accedere al sistema.
* Login: procedura in cui s’inseriscono Email e Password e si richiede l’accesso al sistema.
* Logout: procedura in cui l’utente chiude la connessione verso il social network. In ogni momento potrà eseguire il Login e riaprirla.
* Registrazione: procedura con cui un Visitatore può diventare un Utente e di conseguenza avrà accesso a tutti i vantaggi del possedere un profilo personale, come ad esempio la ricerca di aiuto e il rilascio feedback.
* Social Network: quando si utilizzerà questo termine, ci si riferirà ad un generico sistema in cui vi sono diversi utenti che interagiscono tra loro.
* Feedback: è il metodo con cui gli utenti che hanno richiesto aiuto esprimono una valutazione (voto da 1 a 5 ed eventualmente un commento) verso chi l’ha fornito.
* CV: diminutivo di “Curriculum Vitae” è un documento redatto al fine di presentare la situazione personale, scolastica e lavorativa di una persona. In SWIMv2 costituisce la prova delle conoscenze che possiede un utente e tramite esso sono definite/modificate/rifiutate le sue abilità.
* **Insieme generale delle abilità: lista di abilità che gli Utenti possono selezionare in fase di Registrazione,** fornendo il CV che ne attesti la competenza. Tale insieme rappresenta tutte le possibili voci selezionabili.
* **Insieme personale delle abilità: lista di abilità associate ad un singolo Utente.** Esse sono visualizzabili sul profilo personale dell’Utente e costituiscono un sottoinsieme dell’insieme generale delle abilità (in casi particolari potrebbero coincidere).
* **Modifica abilità: richiesta all’amministratore della modifica dell’insieme personale delle abilità**
* **Proposta abilità: richiesta all’amministratore dell’aggiunta di un elemento nell’insieme generale delle abilità**
* **Suggerimento: parola utilizzata per indicare i profili che il sistema mostra dopo aver stretto amicizia con un altro Utente. Nei successivi documenti sarà definita in modo più preciso, considerando anche un caso particolare.**

## Documenti di riferimento

* IEEE standard for RASD (IEEE 830-1993, vedi anche Blum 1992, pag. 160)
* Informazioni date dal committente
* Il documento di Project Planning realizzato nella prima fase del progetto

## Panoramica

La restante parte del documento è una descrizione complessiva del software che verrà prodotto.

Nella sezione 2 sono indicati i fattori generali che influiscono sul prodotto, le funzionalità del sistema, gli utenti a cui è destinato e alcune assunzioni che sono state fatte per chiarire certe ambiguità.

Si prevede che il sistema sarà utilizzato da alcune centinaia di persone entro il primo anno dal lancio.

Inoltre, si considera che inizialmente la maggior parte degli utenti sarà composta da studenti universitari.

Nelle sezioni 3 e 4 sono invece indicati tutti i requisiti software, ad un livello di dettaglio sufficiente per permettere a designer e programmatori di creare un sistema che soddisfi tali requisiti.

L’interfaccia utente potrà essere di due tipi: un’applicazione web, alla quale l’utente potrà accedervi con un qualunque browser e un’applicazione Java da scaricare ed installare sul proprio computer.

Per quel che riguarda il client, qualsiasi combinazione hardware è ritenuta sufficiente, poiché la maggior parte della computazione è delegata al server. Di conseguenza, il server dovrà avere un hardware adeguato, in grado di gestire anche centinaia di richieste in brevi lassi temporali.

L’applicazione è pensata limitatamente ad un ambiente urbano, anche se potrà rimanere aperta a possibili sviluppi futuri.

# Descrizione generale

Dopo l’incontro col committente in data 9-11-2012 presso il proprio ufficio nella sede centrale di Milano, sono stati compresi meglio alcuni aspetti del prodotto.

In particolare:

* **SWIMv2 è l’aggiornamento di un sistema limitato e unicamente su invito. Lo scopo di questa versione è di realizzarne una pubblica ed accessibile da tutti.**
* **Questo social network avrà inizialmente un pubblico limitato ad un’area urbana (una città) per poi essere esteso su una scala più vasta, in caso d’interesse da parte del pubblico.**

## Funzioni del prodotto

Le funzioni principali del prodotto, tratte dal documento ricevuto dal committente, sono le seguenti:

* Utenti
* Devono
* Scegliere almeno un’abilità da inserire nel proprio insieme personale, durante la registrazione.
* Possono
* Cercare altri utenti.
* Inviare/rispondere a richieste di amicizia.
* Ricevere suggerimenti di amicizia dal sistema. Quando un utente stringe amicizia con un altro (senza aver ricevuto precedenti suggerimenti), il primo visualizza una lista di amici consigliati del secondo.
* Cercare aiuto tra gli amici o nell’intera rete di utenti.
* Accettare/rifiutare richieste di aiuto da parte di altri utenti.
* Fornire un feedback dopo aver ricevuto aiuto.
* Modificare l’insieme personale delle abilità associato al suo profilo facendo una richiesta all’amministratore.
* Proporre all’amministratore l’aggiunta di nuovi elementi all’insieme generale delle abilità. In caso di approvazione, le modifiche saranno disponibili a tutti gli utenti del social network (per maggiori informazioni fare riferimento ai capitoli successivi).
* Visitatori
* Possono utilizzare la funzione di ricerca per i visitatori, che in base alle abilità richieste accedono alla lista degli utenti con quelle abilità, senza però la possibilità di avere contatti con loro.
* Possono registrarsi al sistema.
* Amministratore
* Deve definire l’insieme delle abilità degli utenti al momento della registrazione.
* Può confermare/rifiutare le modifiche all’insieme generale delle abilità (vedi definizione nel capitolo 1.3).
* Può confermare/rifiutare richieste di modifica da parte di utenti al proprio insieme personale di abilità, dopo aver visionato l’opportuna documentazione (CV) che ne attesti la competenza.

## Caratteristiche degli utenti

L’analisi degli utenti è stata svolta dal committente tramite interviste telefoniche, in modo anonimo e nel totale rispetto della privacy. Lo scopo è stato quello di comprendere meglio il numero di utenti interessati al servizio, le caratteristiche personali e le conoscenze dei potenziali utenti, prima ancora di iniziare la progettazione.

Grazie a quest’accurata analisi, **il committente ha fornito i seguenti dati:**

1. **SWIMv2 sarà probabilmente utilizzato da alcune centinaia di utenti nel primo anno dalla data di lancio per testare il sistema e prepararsi ad eventuali investimenti pubblicitari.**
2. **L’interfaccia grafica deve essere interamente in italiano (eventualmente traducibile ed adattabile in futuro).**
3. **Gli utenti non hanno abilità predominanti sulle altre. Probabilmente, durante le fasi iniziali gli studenti universitari con conoscenze informatiche saranno in numero maggiore, poiché sono i più propensi a testare nuovi software.**

## Assunzioni

Durante la prima lettura del documento fornito dal committente, sono stati individuati dei punti poco chiari, incompleti o ambigui. Di conseguenza, è stato fissato un appuntamento in data 9-11-2012 presso la sede centrale di Milano della società che ha commissionato il progetto. Durante il colloquio con i responsabili del progetto, sono state ridiscusse nel dettaglio tutte le possibili funzionalità (non specificate nel documento di descrizione).

* **Un utente si registra inserendo nome, cognome, email, password e selezionando una o più abilità.** Durante la registrazione **l’utente** **deve effettuare l’upload del suo CV**. In questo modo l’amministratore, visionando il curriculum può verificarne la legittimità, e accettare o meno le singole abilità.
* Per ora non conviene prevedere un sistema di chat, perché non se ne conosce la difficoltà implementativa ed inoltre non è espressamente richiesto. Nel caso in cui si rivelasse necessaria sarà implementarla in versioni successive dell’applicazione.
* **Ricerca per i visitatori:** il sistema prevede una funzionalità di ricerca anche per i visitatori. Questo tipo di ricerca consente ad un Visitatore di ricercare gli Utenti in base alle abilità. Tuttavia, una volta che il sistema ha terminato la ricerca e visualizzato i risultati, il visitatore può solamente leggere l’elenco dei nomi. Per tutte le altre funzionalità è richiesta la registrazione.
* **Ricerca di utenti:** la ricerca utenti eseguita svolta tramite l’inserimento di Nome e Cognome.
* **Ricerca di aiuto:** **gli utenti possono ricercare aiuto scegliendo le abilità che stanno ricercando. Il sistema restituisce una lista di utenti che sono in possesso di tutte le abilità selezionate.** Per ora non è stato scelto se usare menu a tendina, oppure semplici campi di testo per la selezione delle abilità.
* Feedback: la collaborazione può essere aperta e chiusa solamente dall’Utente che ha richiesto l’aiuto. Una volta terminata la collaborazione, l’Utente che l’ha richiesta ha la possibilità di rilasciare un feedback. Il feedback può essere rilasciato subito dopo la cessazione della collaborazione, o in un secondo momento accedendo alla sezione apposita. Qui l’utente troverà l’elenco di tutte le collaborazioni. Selezionando la singola collaborazione sarà possibile inserire un commento e una valutazione compresa tra 1 e 5, tramite un menu a tendina.
* **Proporre abilità all’amministratore:** compilando un’apposita sezione**, gli utenti hanno la possibilità di suggerire all’amministratore abilità da aggiungere all’insieme generale di abilità.** Se per esempio un Utente nota la mancanza dell’abilità “Avvocato” tra quelle selezionabili, può proporne l’aggiunta all’amministratore, in modo tale che in futuro possa essere scelta anche da altri utenti.
* **Amministratore**: **può ricevere proposte di aggiunta all’insieme generale delle abilità, o modifiche a quello personale. In quest’ultimo caso l’Amministratore ha la possibilità di accettare o rifiutare la richiesta, dopo aver letto il CV dell’Utente.**
* **L’Utente può richiedere la modifica di più elementi dell’insieme personale delle abilità,** ma una volta inviata la richiesta all’amministratore deve attendere la risposta positiva o negativa. Cioè, finché non arriva una risposta dall’amministratore, l’Utente non può inviare nuove richieste.

**L’Utente può proporre l’aggiunta di un solo elemento dell’insieme generale delle abilità.** Egli può inviare più proposte, anche consecutive, ma non riceverà nessuna conferma dall’Amministratore. Sarà quest’ultimo a valutarle e scegliere se aggiungere le abilità nell’insieme generale delle abilità.

**L’Utente non comunica con l’Amministratore in modo diretto, ma tramite una pagina da compilare.**

**In seguito, sarà specificata meglio la logica implementativa.**

**Attenzione: quando l’Utente vuole modificare l’insieme personale delle abilità deve caricare il CV aggiornato (comprensivo dei dati precedenti e quelli per provare le nuove conoscenze acquisite).**

* Non è previsto nessun sistema di sicurezza nella prima versione di SWIMv2. Eventuali miglioramenti potranno essere aggiunti in seguito.
* **Login Utente: il login si effettua inserendo EMAIL e PASSWORD**
* **Login Amministratore: il login sarà effettuato tramite EMAIL e PASSWORD in un’altra sezione rispetto a quello degli utenti.**
* L’insieme generale delle abilità iniziale sarà costituito da pochi e generici elementi. L’idea è che esso sia popolato a mano a mano dalle abilità proposte dagli utenti.

# Requisiti, Diagrammi UML e BPMN

## Requisiti Funzionali

* + 1. Attori

Gli attori sono le entità che interagiscono attivamente con il sistema. Quelli individuati sono i seguenti:

* Utenti: devono scegliere almeno un’abilità da inserire nel proprio insieme personale, cercare altri utenti, inviare/rispondere a richieste di amicizia o aiuto tra gli amici o nell’intera rete di utenti. Per far si che la qualità del servizio si mantenga sempre elevata, essi hanno la possibilità di fornire un feedback dopo aver ricevuto aiuto. Inoltre, possono modificare il proprio insieme personale di abilità e/o proporne l’aggiunta di nuove nell’insieme globale di abilità. Infine, quando un utente stringe amicizia con un altro (senza aver ricevuto precedenti suggerimenti), il primo visualizza una lista di amici consigliati del secondo.
* Visitatori: possono solamente visualizzare la lista degli utenti e le loro abilità, senza poter accedere al profilo.
* Amministratore: ha il compito di definire l’insieme generale delle abilità ed eventualmente confermare/rifiutare proposte di aggiunta. Inoltre, ha il compito di approvare/rifiutare richieste di modifica degli insiemi personali d’abilità degli utenti.
  + 1. Scenari
* **Attore principale: Visitatore**
* **Scenario 1**

Il visitatore Marco Bianchi è alla ricerca di un aiuto per lo sviluppo del sito web, della squadra di calcio in cui gioca.

Decide quindi di provare il nuovo social network SWMv2, suggeritogli da un amico. Una volta visualizzata l’home page, decide quindi di utilizzare la funzione di ricerca per i visitatori. Inserite le abilità di cui ha bisogno (conoscenza “slim3” e “jsp”), visualizza la lista di tuti gli utenti del sito con tali abilità. Notando che l’unico modo per entrare in contatto con gli utenti visualizzati è registrarsi, decide di non avvalersi del servizio.

* **Scenario 2**

Luca Neri è alla ricerca di aiuto per un lavoro, e decide di registrarsi al social network SWIMv2.

Accede quindi alla pagina di registrazione del sito, compila i dati e seleziona le abilità di cui è in possesso, scegliendo tra quelle elencate dal sistema. Dopo aver fatto l’upload del suo CV, invia la richiesta di registrazione.

* **Attore principale: Utente**
* **Scenario 3**

Mario Rossi accede al sistema tramite la pagina di login inserendo email e password. Una volta effettuato l’accesso, poiché si è registrato da poco, decide di farsi conoscere tra gli utenti del social network. Così sfruttando la funzione di ricerca del sistema, inizia ad inserire nomi di persone conosciute. Dopo una breve ricerca trova finalmente un suo ex collega di lavoro, e decide di inviargli una richiesta di amicizia.

* **Scenario 4**

Alessandra Bassi ha bisogno di aiuto per riparare un mobile del suo appartamento. Purtroppo durante le feste di Natale è difficile trovare un falegname disponibile, così decide di collegarsi al social network SWIMv2, di cui fa parte già da diverso tempo. Inserisce Email e Password ed attende che il Sistema le fornisca l’accesso. A questo punto esegue una ricerca d’aiuto tra i suoi amici con la competenza “falegname”, ma purtroppo si rende conto che nessuno sa svolgere questa mansione. Di conseguenza, esegue la stessa ricerca tra tutti gli utenti e sceglie quello che ritiene più affidabile, un falegname di nome Matteo Rossi. Dopo aver visualizzato il profilo, invia una richiesta di aiuto e chiude la pagina, in attesa di una risposta.

* **Scenario 5**

Un falegname di nome Matteo Rossi accede al social network SWIMv2, per vedere se altri utenti hanno bisogno di lui. Egli trova quattro richieste di aiuto da parte di Mirko Verdi, Alessandra Bassi, Paolo Gialli e Silvia Bianchi ed una di amicizia di Marco Fumagalli, un collega. Dopo aver stretto amicizia con Marco, il sistema suggerisce altri potenziali amici di Marco che non sono già amici di Matteo, ma quest’ultimo decide di non svolgere tale procedura. Infatti, deve ancora visualizzare le richieste di aiuto. Una volta fatto ciò, accetta quelle di Alessandra e Paolo.

* **Scenario 6**

Attila Baldini ha appena pubblicato sull’Android Market un’applicazione che ha programmato grazie all’aiuto di Annibale Verdi, un amico conosciuto sul social network SWIMv2. Considerata la professionalità e la disponibilità che Annibale ha dimostrato, una volta terminata la “pratica di collaborazione”, Attila si reca nella pagina di rilascio del feedback, fornendo una valutazione di 5 (valore massimo) ed un commento. Conclusa questa procedura decide di aggiornare il suo profilo con le ultime abilità acquisite. Si reca nella sezione apposita, ma tra le abilità elencate nota la mancanza della voce “Programmatore Android”. Allora decide di proporne l’aggiunta, inviando la richiesta.

* **Attore principale: Amministratore**
* **Scenario 7**

L’Amministratore esegue il Login a SWIMv2 dalla pagina di accesso inserendo Email e Password, quindi attende che il sistema restituisca la pagina di amministrazione.

Il sistema avvisa che ci sono due nuove registrazioni da approvare, così seleziona la prima (utente Luca Mattei). Dopo aver letto le informazioni inserite da Luca e il suo CV, conferma la registrazione ed approva le abilità inserite dall’utente. Dopodiché, esegue lo stesso procedimento sulla seconda richiesta, ma nella lettura del CV si accorge che l’utente Massimo Belli non ha provato di saper programmare in PHP, nonostante lo abbia selezionato tra le abilità. Di conseguenza decide di confermare la registrazione dell’utente, ma disattiva l’abilità non provata.

* **Scenario 8**

L’Amministratore, dopo aver eseguito il Login dalla pagina di accesso di SWIMv2 visualizza la pagina di amministrazione e nota che ci sono proposte di nuove abilità da inserire nell’insieme generale delle abilità del social network. Inizia a leggerle, e decide di inserirle tutte. Accede quindi al servizio di aggiunta abilità, e le inserisce. Una volta elencate tutte le abilità da inserire, salva le modifiche.

* **Scenario 9**

L’amministratore esegue il Login dalla pagina di accesso di SWIMv2 inserendo Email e Password, quindi attende che il sistema restituisca la pagina di amministrazione.

Il sistema avvisa la presenza di una richiesta di modifica abilità (inserimento conoscenza del linguaggio Java) da parte dell’utente Andrea Verdi.

L’amministratore visualizza il CV aggiornato dell’utente in cui compare la voce “Conoscenze in Java a seguito dello sviluppo del progetto d’Ingegneria del Software 1 presso il Politecnico di Milano”. Ritendo valida la qualifica decide di approvare la richiesta, ed aggiunge all’insieme personale delle abilità di Andrea la conoscenza del linguaggio Java.

* + 1. Casi d’uso

**Attore principale: Visitatore**

* Caso d’uso 1

|  |  |
| --- | --- |
| Titolo | Ricerca aiuto Visitatore |
| Attori principali | **Visitatore** |
| Condizioni ingresso | Un **Visitatore** cerca aiuto per un lavoro |
| Flusso eventi | * **Visitatore** accede alla pagina iniziale (home page) * **Visitatore** accede alla funzionalità di ricerca per i visitatori * **Visitatore** inserisce nella giusta sezione le abilità che sta cercando * **Visitatore** clicca sul pulsante per ricercare * **Visitatore** esce dal sito |
| Condizioni uscita | Il **Visitatore** decide di non usare il Social Network |
| Eccezioni | Il sistema non riesce a concludere l’operazione e quindi viene mostrato un messaggio di errore |

* *Caso d’uso 2*

|  |  |
| --- | --- |
| Titolo | Registrazione a SWIMv2 |
| Attori | **Visitatore** |
| Condizioni ingresso | **Visitatore** non ha un profilo a SWIMv2 e si vuole registrare |
| Flusso eventi | * Il **Visitatore** clicca sul pulsante per la registrazione * Il sistema fornisce la pagina richiesta * Il **Visitatore** inserisce “nome”, “cognome”, “Email”, “Password”, “CV”, “foto del profilo” e sceglie le sue abilità * Il sistema verifica i dati e in caso positivo avvisa il **Visitatore** dell’avvenuta registrazione |
| Condizioni uscita | Il sistema avvisa il **Visitatore** dell’avvenuta registrazione |
| Eccezioni | * Il **Visitatore** ha inserito dati non validi, allora il sistema risponde con un messaggio d’errore e non permette l’accesso * Il sistema non riesce a concludere l’operazione e quindi viene mostrato un messaggio di errore |

**Attore principale: Utente**

* Caso d’uso 1

|  |  |
| --- | --- |
| Titolo | Login Utente |
| Attori | **Utente** |
| Condizioni ingresso | L’**Utente** accede a SWIMv2 |
| Flusso eventi | * L’**Utente** inserisce “Email” e “Password” * Il sistema controlla i dati inseriti per il Login * Il sistema restituisce l’homepage dell’**Utente** |
| Condizioni uscita | L’**Utente** esegue il “Logout” |
| Eccezioni | * L’Email”” o la “Password” inserite dall’Utente sono errate, in questo caso il sistema ripropone la pagina di login |

* Caso d’uso 2

|  |  |
| --- | --- |
| Titolo | Ricerca Utente |
| Attori | **Utente** |
| Condizioni ingresso | L’**Utente** ha effettuato il Login  L’**Utente** desidera ricercare una persona |
| Flusso eventi | * L’**Utente** accede alla sezione di ricerca utenti * L’**Utente** inserisce il “nome” e il “cognome” della persona che desidera cercare * L’**Utente** clicca sul pulsante per la ricerca |
| Condizioni uscita | Il sistema visualizza l’elenco di utenti con “nome” e “cognome” specificati |
| Eccezioni | * L’**Utente** ha inserito dati non validi e quindi il sistema ripropone la pagina di ricerca * Il sistema non riesce a terminare l’operazione e ripropone la pagina di ricerca * Non è stato prodotto nessun risultato dalla ricerca per “nome” e “cognome” |

* Caso d’uso 3

|  |  |
| --- | --- |
| Titolo | Richiesta amicizia |
| Attori | **Utente** |
| Condizioni ingresso | L’**Utente** ha effettuato il Login  L’**Utente** ha ricercato la persona |
| Flusso eventi | * L’**Utente** visualizza il profilo desiderato * L’**Utente** clicca sul pulsante per richiedere amicizia * Il sistema conferma l’operazione |
| Condizioni uscita | Il sistema inoltra la richiesta di amicizia |
| Eccezioni | * Il sistema non riesce a concludere l’operazione e mostra una pagina di errore * L’**Utente** si accorge che il profilo visualizzato è di un omonimo (stesso nome e cognome), quindi decide di non completare l’operazione |

* Caso d’uso 4

|  |  |
| --- | --- |
| Titolo | Ricerca aiuto Utente |
| Attori | **Utente** |
| Condizioni ingresso | L’**Utente** ha effettuato il Login  L’**Utente** desidera ricercare aiuto |
| Flusso eventi | * L’**Utente** accede alla sezione di ricerca aiuto * L’**Utente** inserisce la/e abilità richiesta/e * L’**Utente** clicca sul pulsante per la ricerca |
| Condizioni uscita | Il sistema propone l’elenco di utenti con le abilità specificate |
| Eccezioni | * Il sistema non riesce a terminare l’operazione e ripropone la pagina di ricerca * Non è stato prodotto nessun risultato dalla ricerca |

* Caso d’uso 5

|  |  |
| --- | --- |
| Titolo | Invio richiesta di aiuto |
| Attori | **Utente** |
| Condizioni ingresso | L’**Utente** ha effettuato il login  L’**Utente** è in cerca di aiuto |
| Flusso eventi | * L’**Utente** visualizza il profilo desiderato * L’**Utente** clicca sul pulsante per richiedere aiuto * Il sistema conferma l’operazione |
| Condizioni uscita | Il sistema inoltra la richiesta di aiuto |
| Eccezioni | * Il sistema non riesce a concludere l’operazione e mostra una pagina d’errore |

* Caso d’uso 6

|  |  |
| --- | --- |
| Titolo | Accettare una richiesta di aiuto |
| Attori | **Utente** |
| Condizioni ingresso | * L’**Utente** ha effettuato il Login * L’**Utente** ha ricevuto una richiesta di aiuto |
| Flusso eventi | * Il sistema avvisa l’**Utente** della presenza di richieste di aiuto * L’**Utente** visualizza la richiesta e decide di accettarla * Il sistema conferma l’operazione |
| Condizioni uscita | Il sistema notifica l’accettazione della richiesta di aiuto |
| Eccezioni | * Il sistema non riesce a concludere l’operazione e quindi viene mostrato un messaggio d’errore |

* Caso d’uso 7

|  |  |
| --- | --- |
| Titolo | Accettare una richiesta di amicizia |
| Attori | **Utente** |
| Condizioni ingresso | * L’**Utente** ha effettuato il Login * L’**Utente** ha ricevuto una richiesta di amicizia |
| Flusso eventi | * Il sistema avvisa l’**Utente** della presenza di richieste di amicizia * **L’Utente** accetta la richiesta di amicizia * Il sistema suggerisce alcuni amici cercando tra quelli dell’utente con cui ha appena stretto amicizia |
| Condizioni uscita | Il sistema notifica l’accettazione della richiesta di amicizia |
| Eccezioni | Il sistema non riesce a terminare l’operazione e quindi mostra un messaggio d’errore |

* Caso d’uso 8

|  |  |
| --- | --- |
| Titolo | Rilascio del Feedback |
| Attori | **Utente** |
| Condizioni ingresso | * L’**Utente** ha effettuato il Login * L’**Utente** ha terminato una collaborazione * L’**Utente** non ha lasciato il Feedback associato a quella collaborazione |
| Flusso eventi | * L’**Utente** accede alla pagina per rilasciare il Feedback * Il sistema fornisce la lista delle collaborazioni * L’**Utente** fornisce la valutazione con un voto da 1 a 5 ed un commento * L’**Utente** clicca il pulsante per inviare il Feedback * Il sistema conferma l’operazione |
| Condizioni uscita | Il sistema notifica il rilascio del Feedback al collaboratore |
| Eccezioni | * Il sistema non riesce a terminare l’operazione e quindi è mostrato un messaggio d’errore * Non ci sono Feedback da rilasciare |

* Caso d’uso 9

|  |  |
| --- | --- |
| Titolo | Proposta aggiunta abilità |
| Attori | **Utente** |
| Condizioni ingresso | * L’**Utente** ha effettuato il Login * L’**Utente** vuole proporre all’amministratore l’aggiunta di un’abilità |
| Flusso eventi | * L’**Utente** accede al proprio profilo * L’**Utente** clicca sul pulsante per proporre ua nuova abilità * L’**Utente** inserisce il “nome di una nuova abilità” * L’**Utente** clicca sul pulsante per inviare la proposta * Il sistema conferma l’operazione |
| Condizioni uscita | Il sistema notifica la proposta all’amministratore |
| Eccezioni | * Il sistema non riesce a terminare l’operazione e quindi è mostrato un messaggio d’errore * L’**Utente** inserisce dati non validi |

**Attore principale: Amministratore**

* Caso d’uso 1

|  |  |
| --- | --- |
| Titolo | Login Amministratore |
| Attori | **Amministratore** |
| Condizioni ingresso | L’**Amministratore** accede al sito del Social Network |
| Flusso eventi | * L’**Amministratore** accede alla pagina di login per l’amministratore del sistema * L’**Amministratore** inserisce “Email” e “Password” * Il sistema controlla i dati inseriti * Il sistema restituisce l’home page dell’**Amministratore** |
| Condizioni uscita | L’**Amministratore** effettua il Logout |
| Eccezioni | * I dati inseriti dall’**Amministratore** sono errati, in questo caso il sistema ripropone la pagina di Login per amministratori |

* Caso d’uso 2

|  |  |
| --- | --- |
| Titolo | Conferma registrazione Utente |
| Attori | **Amministratore** |
| Condizioni ingresso | * L’**Amministratore** effettua il Login * Un utente ha inviato la richiesta di registrazione * La richiesta di registrazione è stata inoltrata con successo |
| Flusso eventi | * L’**Amministratore** accede alla pagina di richiesta di registrazione dell’utente * L’**Amministratore** verifica le informazioni inserite dall’utente, e l’insieme personale delle abilità * L’**Amministratore** invia al sistema la conferma di registrazione e relative abilità del nuovo utente |
| Condizioni uscita | * Il sistema conferma la riuscita dell’operazione all’**Amministratore** * Il sistema conferma la registrazione all’utente |
| Eccezioni | * Il sistema si blocca durante l’operazione di conferma, e ripropone all’**Amministratore** la procedura di conferma * Una o più abilità inserite dall’utente non sono documentate, l’**Amministratore** prima della conferma, deseleziona queste abilità |

* Caso d’uso 3

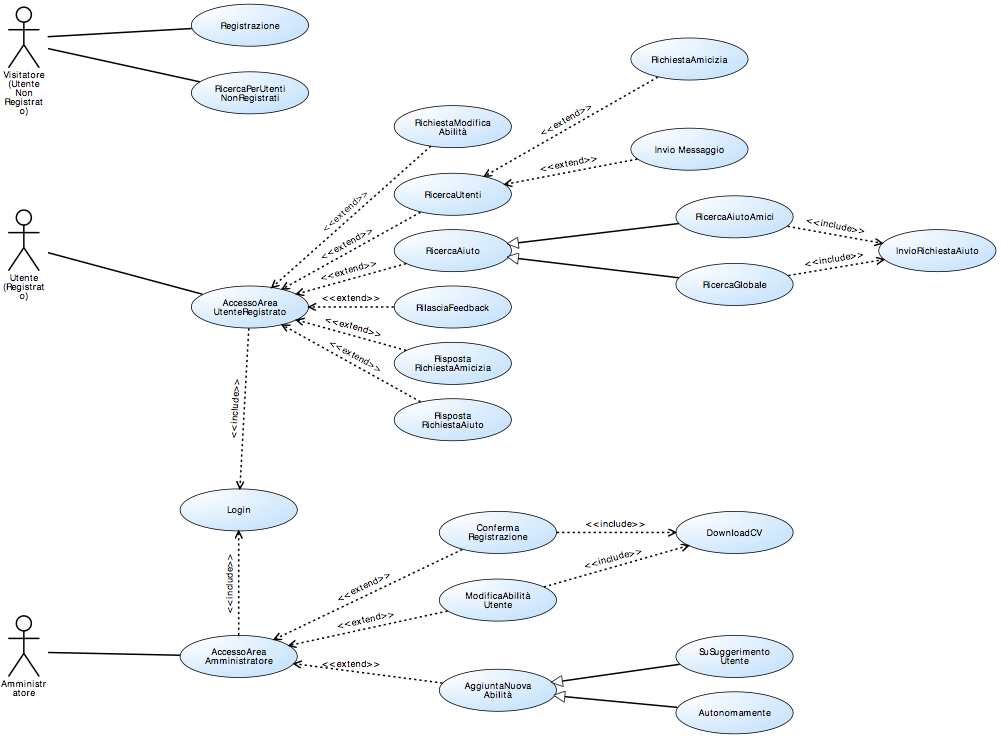
|  |  |
| --- | --- |
| Titolo | Inserimento nuova abilità nell’insieme globale delle abilità |
| Attori | **Amministratore** |
| Condizioni ingresso | * L’**Amministratore** effettua il Login * L’utente ha proposto una nuova abilità al sistema |
| Flusso eventi | * L’**Amministratore** visualizza la proposta di aggiunta nuova abilità dell’utente * L’**Amministratore** invia al sistema la nuova abilità da aggiungere |
| Condizioni uscita | * Il sistema conferma l’operazione |
| Eccezioni | * Il sistema fallisce nell’operazione d’inserimento della nuova abilità, e ripropone all’**Amministratore** la pagina di inserimento * **L’amministratore** decide di non seguire il suggerimento dell’utente, e non aggiunge nessuna abilità |

* Caso d’uso 4

|  |  |
| --- | --- |
| Titolo | Modifica abilità utente |
| Attori | **Amministratore** |
| Condizioni ingresso | * L’**Amministratore** effettua il Login * L’utente ha inviato una richiesta di aggiunta di abilità |
| Flusso eventi | * L’**Amministratore** visualizza la richiesta dell’utente * L’**Amministratore** verifica l’abilità richiesta dall’utente tramite il CV * L’**Amministratore** inserisce l’abilità richiesta nell’insieme personale delle abilità |
| Condizioni uscita | * Il sistema conferma all’**Amministratore** la riuscita dell’operazione * Il sistema notifica all’utente la nuova abilità aggiunta nell’insieme personale delle abilità. |
| Eccezioni | * Il sistema fallisce nell’inserimento della nuova abilità dell’utente e ripropone all’**Amministratore** la pagina di modifica delle abilità di un utente * L’**Amministratore** non ritiene valide le credenziali fornite dall’utente e decide di respingere la richiesta |

## Use Case Diagram

**Il sistema è stato modellato tramite “Use Case Diagram”**, cioè specificando “Attori”, “Casi D’Uso” ed “Associazioni”. Il motivo di questa scelta è che tale diagramma descrive in modo molto chiaro i casi d’uso specificati in precedenza e tutto ciò in un solo scherma. Infatti, **esso fornisce una migliore visione d’insieme sul funzionamento del sistema, descrivendo in modo chiaro i “soggetti” che interagiscono e quali azioni possono compiere.**

Fig. 3.1 - Use Case Diagram

## Sequence Diagram

Nel capitolo precedente è stato fornito lo “Use Case Diagram” per spiegare quali attori interagiscono nel sistema e quali azioni possono compiere, ma non per descriverne la sequenza.

Per tale motivo, **è stato scelto di rappresentare la sequenza di azioni nel tempo tramite “Sequence Diagram”.**

Per una migliore chiarezza, il team di sviluppo ha derivato la sequenza di operazioni da ogni scenario, fornendo il diagramma associato ad ognuno di essi. Infatti, questi diagrammi sono molto esplicativi e facilmente leggibili, ma solo se utilizzati per descrivere piccoli scenari e non per una visione completa del sistema come lo “Use Case Diagram”.

Per renderne ancora più facile la lettura, il team di sviluppo ha deciso di aggiungere, alle funzioni chiamate dagli utenti, alcuni valori di esempio (estratti direttamente dagli scenari).

Esempio: nella Figura 3.2, il Visitatore ricerca le abilità “Slim3” e “JSP”. Per chiarire meglio il significato di “3: ricercaAbilità”, nel Sequence Diagram 1 sono state aggiunte le due abilità come parametri, ottenendo “3: ricercaAbilità(Slim3,JSP)”. Tale scelta è stata fatta per tutti i Sequence Diagram che gli sviluppatori hanno ritenuto poco chiari.

**Scenario 1**

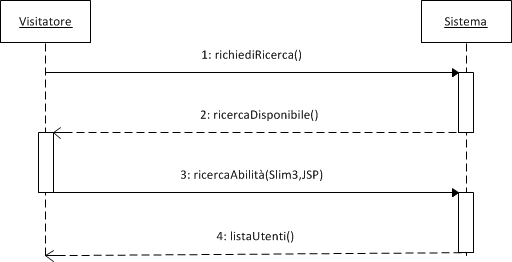


Fig. 3.2 - Sequence Diagram - Scenario 1

**Scenario 2**

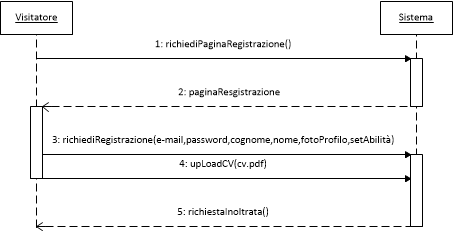


Fig. 3.3 - Sequence Diagram - Scenario 2

**Scenario 3**

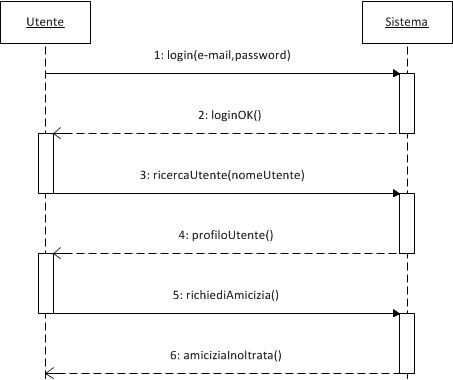


Fig. 3.4 - Sequence Diagram - Scenario 3

**Scenario** **4**

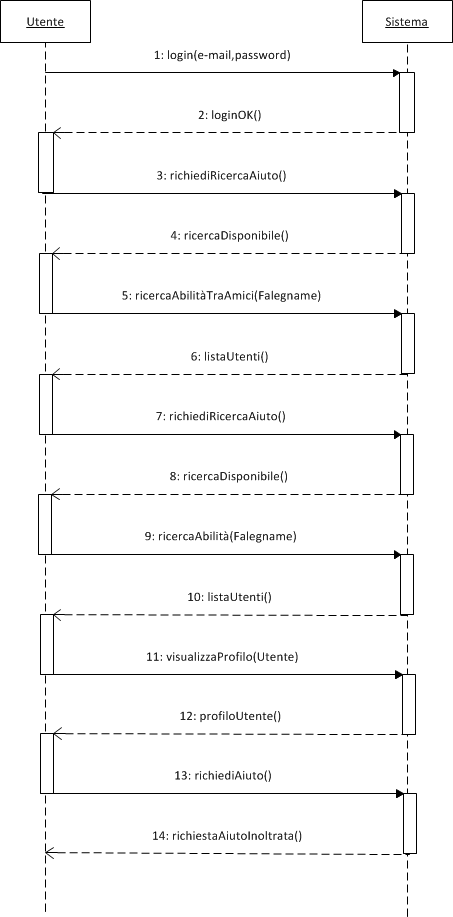


Fig. 3.5 - Sequence Diagram - Scenario 4

**Scenario 5**

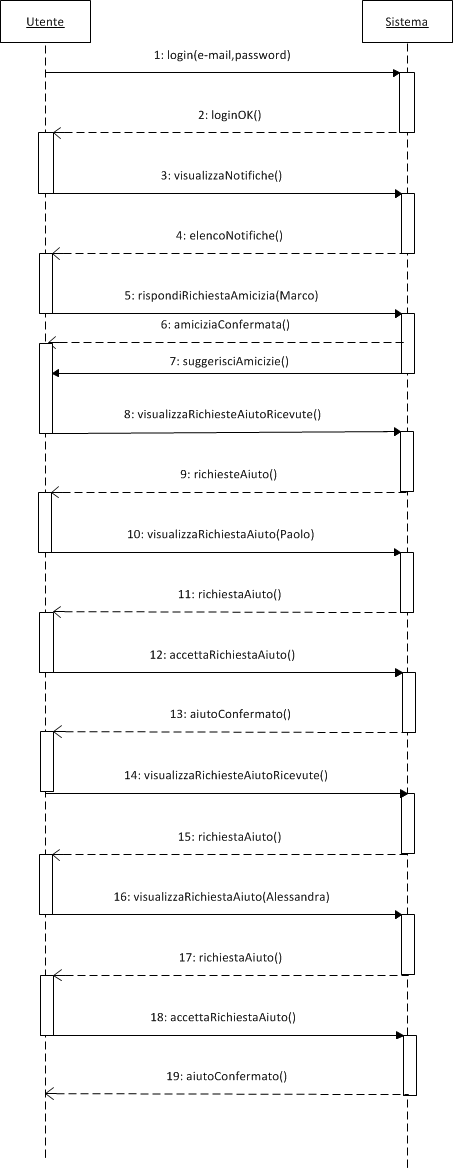


Fig. 3.6 - Sequence Diagram - Scenario 5

**Scenario 6**

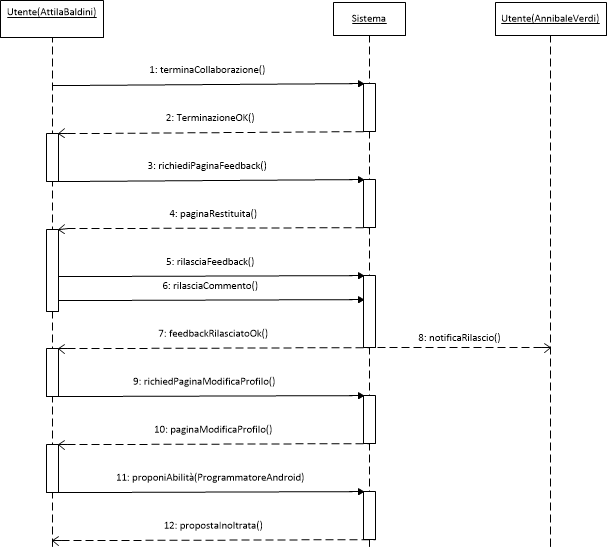


Fig. 3.7 - Sequence Diagram - Scenario 6

**Scenario 7**

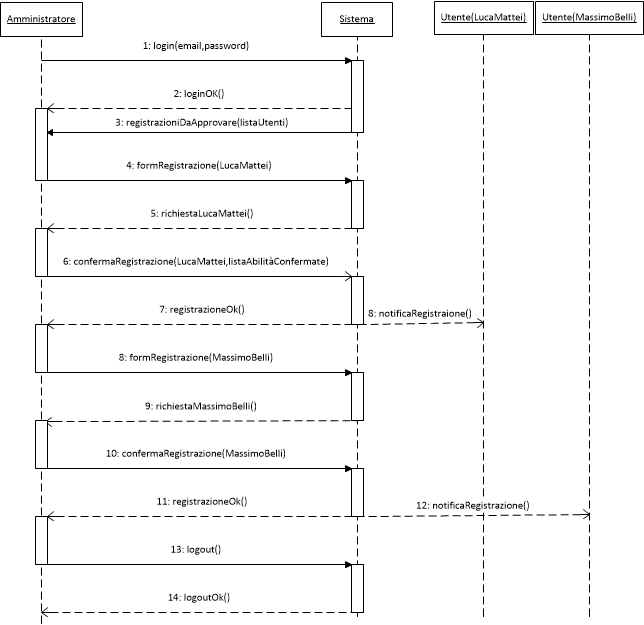


Fig. 3.8 - Sequence Diagram - Scenario 7

**Scenario 8**

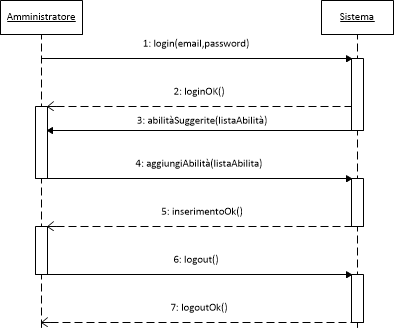


Fig. 3.9 - Sequence Diagram - Scenario 8

**Scenario 9**

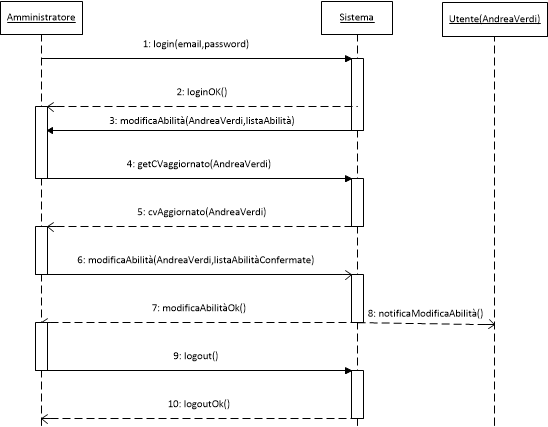


Fig. 3.10 - Sequence Diagram - Scenario 9

## BMPN Diagram

Dopo aver descritto le possibili azioni di ogni singolo attore del sistema e la sequenza di operazioni per ogni scenario, **il team di sviluppo ha ritenuto necessario descrivere anche l’interazione tra gli attori.**

**Per raggiungere questo obiettivo ci sono due possibilità: “Activity Diagram” e “BPMN Diagram”. Il team di sviluppo ha preferito la seconda scelta perché mette in risalto le azioni svolte in ogni scenario, permettendo anche di distinguerle tra gli attori che le hanno compiute.**

**Nota bene: il BPMN Diagram riferito allo scenario 1 è stato ignorato in quanto molto semplice ed intuitivo.**

**Scenario 2**

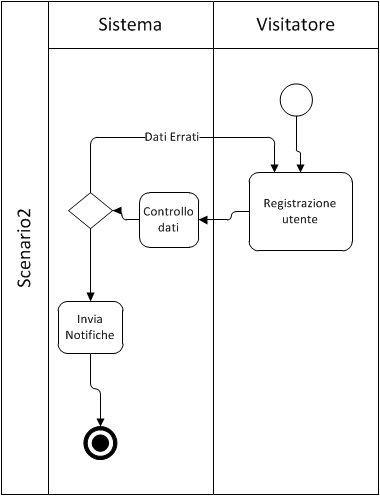


Fig. 3.11 – BMPN – Scenario 2

**Scenario 3**

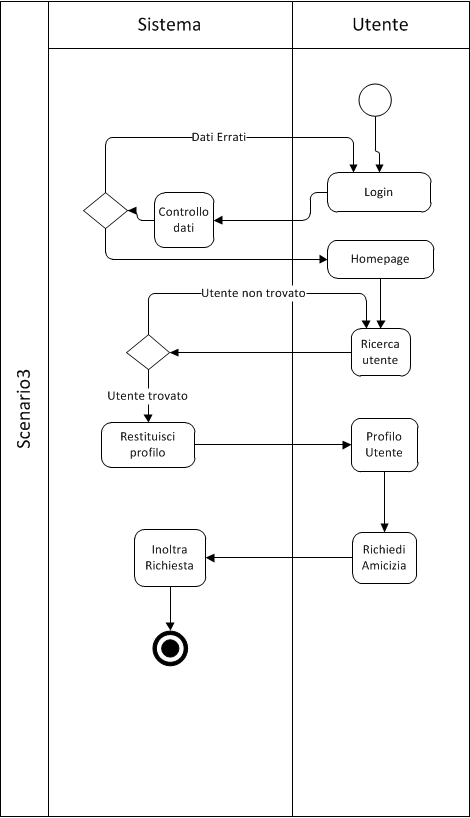


Fig. 3.12 – BMPN – Scenario 3

**Scenario 4**

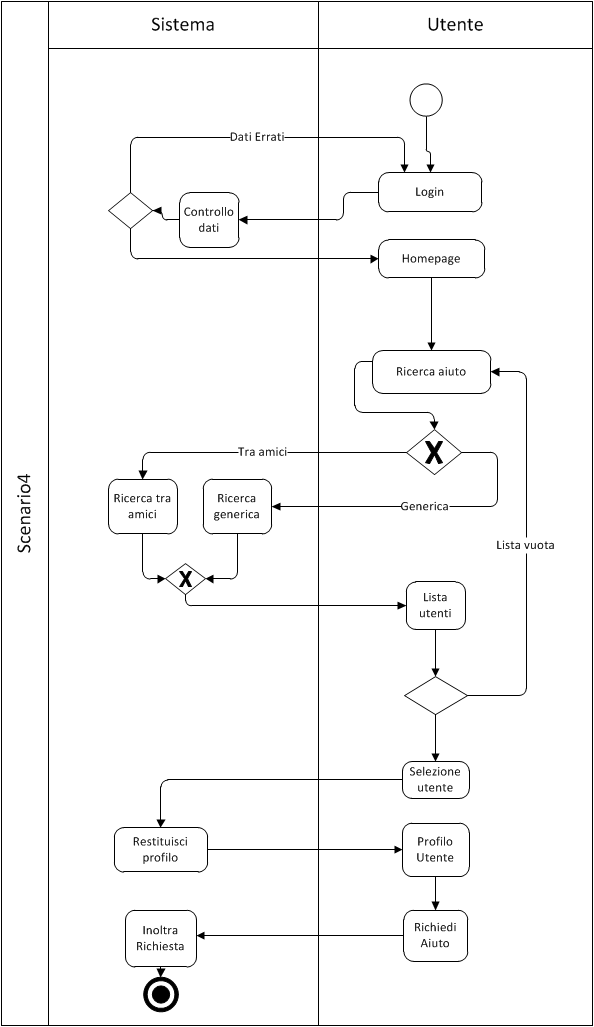


Fig. 3.13 – BMPN – Scenario 4

**Scenario 5**

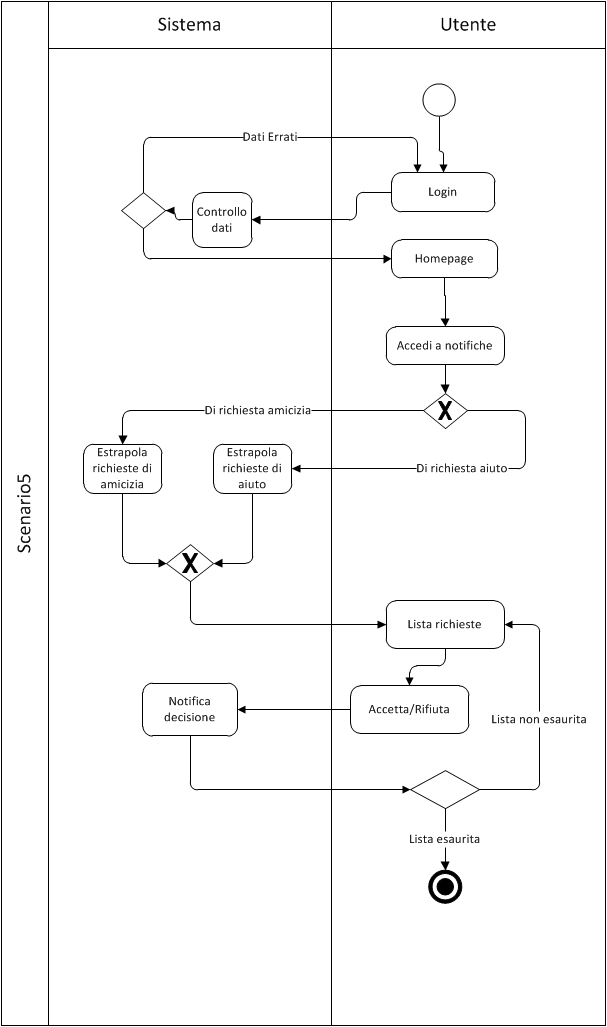


Fig. 3.14 – BMPN – Scenario 5

**Scenario 6**

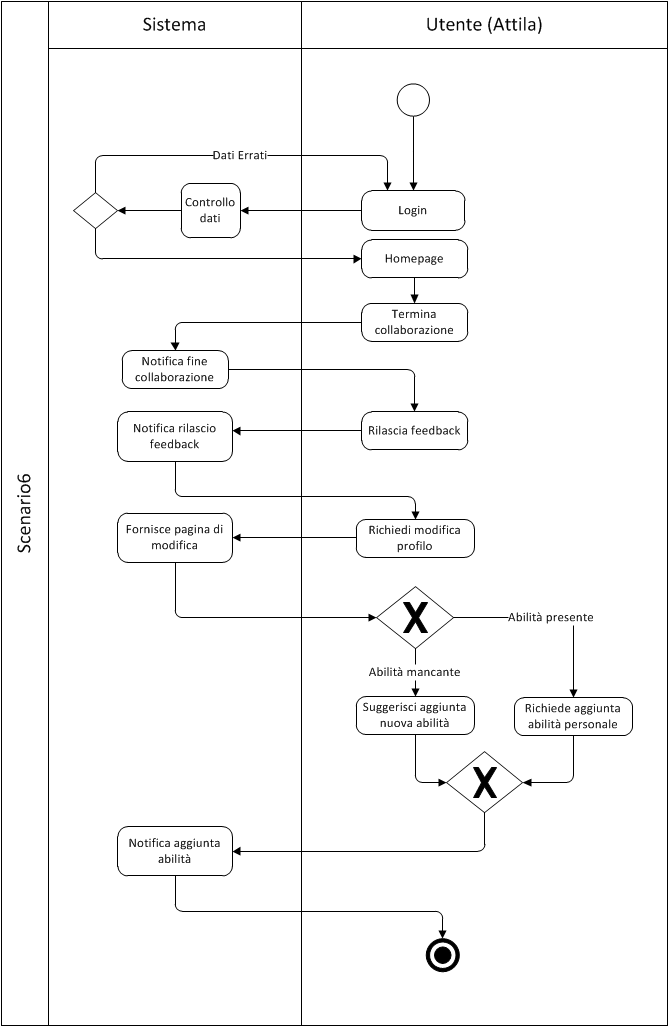


Fig. 3.15 – BMPN – Scenario 6

**Scenario 7**

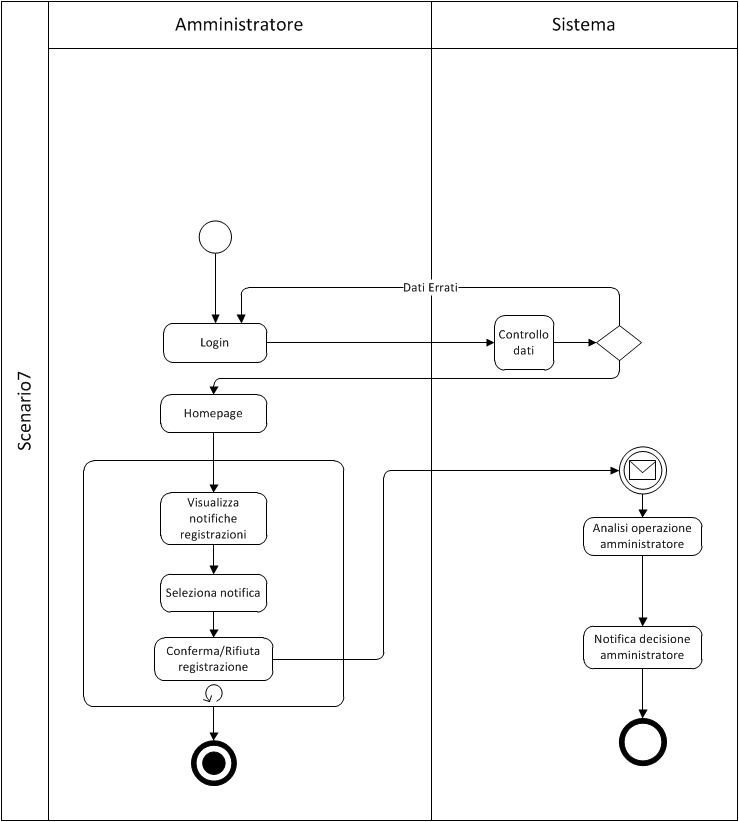


Fig. 3.16 – BMPN – Scenario 7

**Scenario 8**

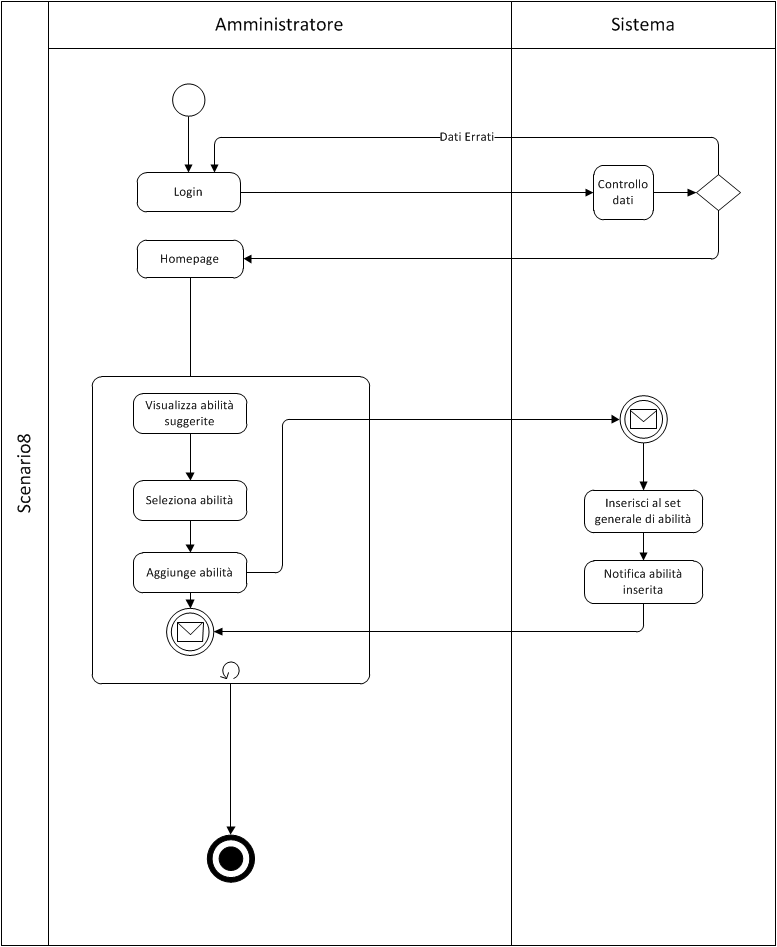


Fig. 3.17 – BMPN – Scenario 8

**Scenario 9**

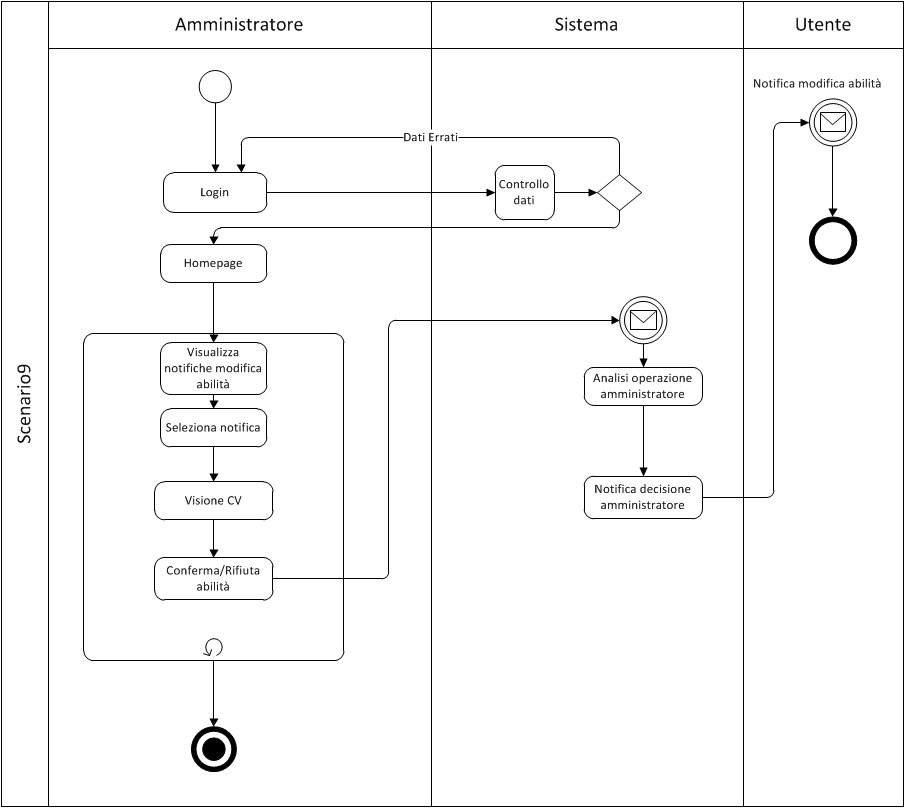


Fig. 3.18 – BMPN – Scenario 9

## Class Diagram

**In questo capitolo è stato inserito il “Class Diagram” dell’intero sistema, evidenziando gli attributi più rilevanti delle classi.**

Questo schema è essenziale per:

* comprendere meglio le possibili “Entità” e le loro relazioni
* derivare le specifiche Alloy (vedi capitolo 5)

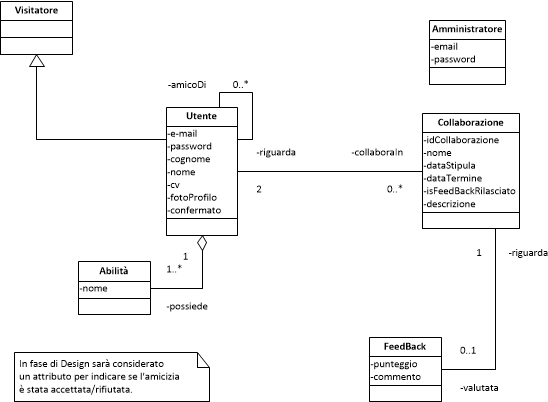


Fig. 3.19 - Class Diagram

Considerazioni sulle scelte di progettazione del team di sviluppo:

* **Visitatore è superclasse di Utente**: perché queste due entità condividono l’indirizzo IP. Ovviamente un Utente possiede delle informazioni per il Login e alcuni dati personali e di conseguenza la relazione di generalizzazione è abbastanza ovvia.
* **Utente possiede Abilità, ma non l’inverso: l’Utente possiede un insieme personale di Abilità (almeno una scelta durante la fase di registrazione), ma le Abilità non sono associate all’Utente perché possono essere condivise tra più utenti.** In un caso reale, questo è veritiero, perchè l’Abilità “programmatore Java” sarà posseduta da molti utenti nel Social Network. Questa relazione è detta “aggregazione”. Nota **l’attributo “confermato” che sarà utilizzato in seguito per distinguere tra utenti, la cui registrazione è stata approvata dall’Amministratori e tra utenti in attesa/rifiutati.**
* **Amministratore è un’entità separata**: perché con Utente non condivide nulla, poiché Utente può svolgere determinate funzioni che però l’Amministratore non esegue. Inoltre, l’Amministratore non appare nemmeno nelle ricerche, non possiede Abilità, non riceve Feedback ecc…  **Dato che l’Amministratore accede al Social Network attraverso una pagina diversa dall’Utente, gli sviluppatori hanno deciso si separare in modo netto le due entità.**
* **Utente è amico di altri utenti**: la relazione descritta come auto-anello su Utente, descrive il fatto che ogni utente può essere amico di altri Utenti (nessuno o più). Nota l’aggiunta per specificare che, nella fase di Design verrà utilizzato anche un attributo aggiuntivo per specificare se la richiesta di amicizia sia stata accettata o rifiutata.
* **Associazione bidirezionale Utente-Collaborazione**: tale associazione è abbastanza scontata perché è logico dire che un Utente può collaborare e che la Collaborazione riguarda 2 utenti.  
  La cardinalità delle relazioni è giustificata dal fatto che per esistere, una Collaborazione deve riguardare due utenti. Inoltre, un Utente può non avere Collaborazioni, per esempio nel momento della registrazione.
* **Associazione bidirezionale Collaborazione-Feedback**: una collaborazione può avere un Feedback (nel caso sia stato rilasciato dall’Utente che ha richiesto aiuto), ma può anche non averlo (nei casi in cui la collaborazione sia appena stata aperta oppure nel caso in cui l’utente che ha richiesto aiuto non lo abbia ancora rilasciato). Inoltre, il Feedback è associato ad una ed una sola Collaborazione perché contiene anche un commento (oltre al voto) legato univocamente alla Collaborazione in questione.

Quando un Utente richiede aiuto, “apre” una Collaborazione con i riferimenti dell’Utente che l’ha aperta (primo elemento) e che l’ha accettata (secondo elemento), tutti gli altri attributi assumono valore=null e isFeedbackRilasciato = false.

E’ bene chiarire il significato degli attributi:

* dataStipula: vale null se la collaborazione non è ancora stata stipulata (non confermata), assume come valore una data, nel caso in cui venga confermata dall’Utente (data del momento dell’accettazione).
* dataTermine: vale null se la collaborazione non è ancora stata terminata (in corso/ancora da confermare o rifiutata), assume come valore una data, nel caso in cui l’Utente che l’ha aperta decida di chiuderla (data del momento della chiusura/termine della collaborazione).
* isFeedbackRilasciato: boolean che vale “true” se la Collaborazione ha ricevuto il Feedback dall’Utente che l’ha aperta, “false” nel caso contrario.

# Requisiti non funzionali

## Interfacce utente

L’interfaccia utente potrà essere di due tipi:

* Un’applicazione web, alla quale l’Utente potrà accedervi con un qualunque browser.
* Un’applicazione Java da scaricare ed installare sul proprio computer.

In ogni caso sarà semplice ed intuitiva, per permettere a qualsiasi utente, con un minimo di conoscenze informatiche, di poterla usare.

## Interfacce software

* + 1. Lato Client

Nel caso di scelta della soluzione interamente basata sul web, i browser consigliati sono:

* Microsoft Internet Explorer
* Apple Safari
* Mozilla Firefox
* Google Chrome

Nel caso in cui il lato client fosse realizzato con un’applicazione specifica in Java, sarà necessario installare sul proprio PC/MAC una Java Virtual Machine (JRE 6 o superiore) ed una connessione internet stabile.

* + 1. Lato Server

Requisiti non funzionali dal lato server:

* Java Runtime Enviroment
* JBoss come application server
* Un web server, per accedere via protocollo HTTP
* Database SQL
* Un qualsiasi sistema operativo

## Interfacce hardware

Per quel che riguarda il client, qualsiasi combinazione hardware è ritenuta sufficiente, poiché la maggior parte della computazione è delegata al server. Di conseguenza, il server deve avere un hardware adeguato, in grado di gestire anche centinaia di richieste in brevi lassi temporali.

## Interfacce di comunicazione

Una volta installato il Web Server, sarà richiesta la disponibilità di HTTP come protocollo per la comunicazione client-server.

## Vincoli

Vincoli di progettazione:

* Piattaforma JEE (Java Enterprise Edition), in particolare usando EJB.
* Nella versione finale dovrà gestire un numero di utenti in un’area geografica urbana, come una città (di conseguenza non c’è un vincolo molto stringente).
* Dovrà poter essere scalabile ad applicazioni su scala più vasta sia in termini di utenti sia come copertura geografica.

Vincoli di prestazioni:

* Nessuna richiesta particolare oltre al “corretto funzionamento”.
* Tempi di risposta e affidabilità del sistema accettabili e migliorabili in caso di estensione del Social Network.

Vincoli hardware:

* Nessun limite specifico. E’ comunque necessario usare “macchine” col supporto alla piattaforma richiesta dal committente.

## Scelte adottate dal team di sviluppo

I requisiti non funzionali di progettazione del sistema riguardano gli strumenti di sviluppo adottati dal team:

* **Microsoft Project 2010**: per la fase di pianificazione. E’ stato utilizzato per creare il “Diagramma di Gantt” nel documento di “Project Planning”.
* **Microsoft Visio 2010**: per la fase di analisi dei requisiti e creazione dei modelli, cioè per la stesura di questo documento. In particolare, con Visio sono stati realizzati: Class Diagram, Sequence Diagram e i BPMN Diagram.
* **Signavio**: è una piattaforma web utilizzata per creare lo “Use Case Diagram”.
* **Poseidon 6**: utilizzato per realizzare gli UX Model e i BCE Model nel Design Document.
* **Microsoft Word 2010**: per la stesura di tutta la documentazione.
* **Eclipse Indigo o NetBeans**: IDE per la stesura del codice Java.
* **JUnit**: plugin di Eclipse o NetBeans per testare il codice.
* **Alloy Analyzer 4.2**: software per la stesura delle specifiche Alloy e per la loro esecuzione.
* **Oracle JDK: 1.6 o 1.7** per compilare il codice Java.

# Alloy

**Alloy è un linguaggio dichiarativo che permette di esprimere le specifiche dei modelli di un sistema o software.** **E’ stato inserito in questo documento, perché espressamente richiesto dalle specifiche fornite dal committente.**

## Rappresentazione del mondo in Alloy

Di seguito è riportato il codice della rappresentazione del mondo attraverso le signature di Alloy.

*module SWIMv2*

*//Classi del Class Diagram portate in Alloy*

***sig*** *Utente* ***extends*** *Visitatore {*

*possiedeAbilita:* ***some*** *Abilita, //può avere una o più abilita*

*collaboraIn:* ***set*** *Collaborazione,*

*amicoDi:* ***set*** *Utente*

*}*

***sig*** *Amministratore {*

*}*

***sig*** *Visitatore {*

*}*

***sig*** *Collaborazione {*

*feedback:* ***lone*** *Feedback, //lone perche' inizialmente non e' stato ancora rilasciato*

*utenteCheHaRichiesto:* ***one*** *Utente, //one perche' per essere avviata un utente deve averla richiesta*

*utenteCheHaRicevuto:* ***one*** *Utente //one perche' deve esserci anche l'utente che ha ricevuto la proposta di collaborazione (non e' detto che l'abbia accettata)*

*}*

***sig*** *Feedback {*

*voto:* ***lone*** *Int, //rappresenta il voto assegnato alla collaborazione*

*riguarda:* ***one*** *Collaborazione,*

*} {*

*voto > 0 and voto < 6 //voto compreso tra 1 e 5*

*}*

***sig*** *Abilita {*

*}*

## Rappresentazione dei vincoli

Qui è riportato il codice Alloy per rappresentare i vincoli del mondo, cioè i fatti, attraverso appunto i fact di Alloy.

//un utente non può essere amico di se stesso

***fact*** *noAmicoDiSeStesso {*

***no*** *u: Utente* ***|*** *u* ***in*** *u.amicoDi*

*}*

*//se esiste una collaborazione, allora i due utenti di una collaborazione devono essere differenti*

***fact*** *collaboratoriDiversi {*

***all*** *c: Collaborazione | c.utenteCheHaRichiesto* ***!=*** *c.utenteCheHaRicevuto*

*}*

*//una collaborazione può riguardare solo 2 utenti, e questi sono esattamente il richiedente e il richiesto*

***fact*** *collaboraIn{*

***all*** *c:Collaborazione,u1:Utente,u2:Utente* ***|*** *(c* ***in*** *u1.collaboraIn* ***&&*** *c* ***in*** *u2.collaboraIn)* ***=>*** *((u1=c.utenteCheHaRichiesto* ***or*** *u1=c.utenteCheHaRicevuto)* ***&&*** *(u2=c.utenteCheHaRicevuto* ***or*** *u2=c.utenteCheHaRichiesto))*

*}*

*//dove è definita, la relazione riguarda è l'inverso della relazione feedback*

***fact*** *feedbackAssociatoACollaborazione{*

***all*** *f:Feedback, c:Collaborazione | f.riguarda=c* ***<=>*** *c.feedback=f*

*}*

*//la relazione collaboraIn è l'inversa di utenteCheHaRichiesto+utenteCheHaRicevuto*

***fact*** *utenteCollaborazione{*

***all*** *c:Collaborazione* ***|*** *c* ***in*** *c.utenteCheHaRichiesto.collaboraIn* ***&&*** *c* ***in*** *c.utenteCheHaRicevuto.collaboraIn*

*}*

*//la relazione amicizia è bidirezionale*

***fact*** *amiciziaBidirezionale{*

***all*** *u1:Utente, u2:Utente | u1* ***in*** *u2.amicoDi* ***<=>*** *u2* ***in*** *u1.amicoDi*

*}*

## Asserzioni

Si definiscono poi alcune asserzioni per effettuare controlli sull’effettiva integrità del sistema.

*//controlliamo che non esiste una collaborazione con i due utenti uguali*

***assert*** *utenteNoInCollaborazioneConSeStesso {*

***all*** *c:Collaborazione* ***|****c.utenteCheHaRichiesto!=c.utenteCheHaRicevuto*

*}*

***check*** *utenteNoInCollaborazioneConSeStesso*

*//controlliamo che un utente non può essere amico di se stesso*

***assert*** *utenteNoAmicoDiSeStesso{*

***no*** *u:Utente****|****u* ***in*** *u.amicoDi*

*}*

***check*** *utenteNoAmicoDiSeStesso*

*//controlliamo che un feedback non può esistere senza una collaborazione*

***assert*** *feedBackNonEsisteSenzaCollaborazione{*

***no*** *f:Feedback | f.riguarda=* ***none***

*}*

***check*** *feedBackNonEsisteSenzaCollaborazione*

*//controlliamo che può esistere una collaborazione senza feedback*

*//in questo caso ci deve ritornare un controesempio*

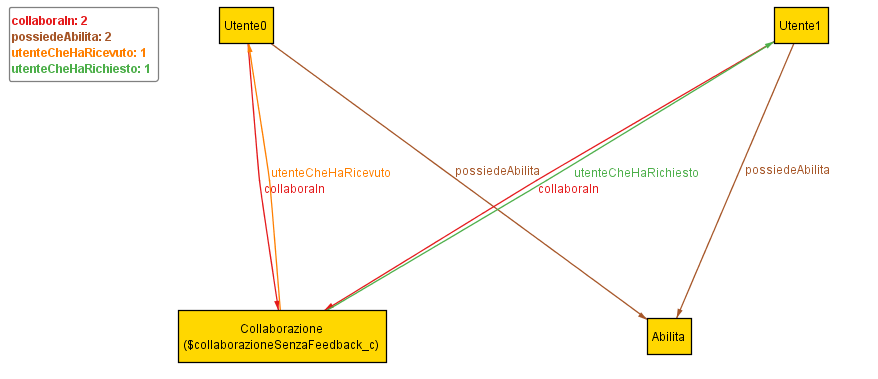
***assert*** *collaborazioneSenzaFeedback{*

***all*** *c:Collaborazione| c.feedback!=****none***

*}*

***check*** *collaborazioneSenzaFeedback*

La mappa di un controesempio trovato da Alloy Analyzer è la seguente. Da essa si può notare l’assenza del Feedback dell’unica collaborazione rappresentata.



*//controlliamo che un feedback non abbia un voto maggiore di 5*

***assert*** *valoreVoto{*

***all*** *f:Feedback****|*** *f.voto<=5*

*}*

***check*** *valoreVoto*

*//controlliamo che una collaborazione deve essere per forza tra due persone*

***assert*** *collaborazioneE'ConDuePersone{*

***all*** *c:Collaborazione****|*** *c.utenteCheHaRichiesto!=* ***none******&&*** *c.utenteCheHaRicevuto!=****none***

*}*

***check*** *collaborazioneE'ConDuePersone*

*//controlliamo che un utente possegga almeno una abilità*

***assert*** *almenoUnaAbilitaPerUtente{*

***all*** *u:Utente****|******#****u.possiedeAbilita>=1*

*}*

***check*** *almenoUnaAbilitaPerUtente*

*//controlliamo che l'amicizia sia sempre bidirezionale*

***assert*** *amiciziaBidirezionale{*

***all*** *u1:Utente,u2:Utente****|*** *u1 in u2.amicoDi* ***<=>*** *u2 in u1.amicoDi*

*}*

***check*** *amiciziaBidirezionale*

## Istanza

E’ riportata la mappa di un’istanza eseguita direttamente da Alloy Analyzer e il codice per visualizzarla.

***pred*** *show(){*

***#****Utente=5*

***#****amicoDi=2*

***#****Abilita=3*

***#****Collaborazione=2*

***#****Feedback=2*

***#****voto=2*

*}*

***run*** *show* ***for*** *10*

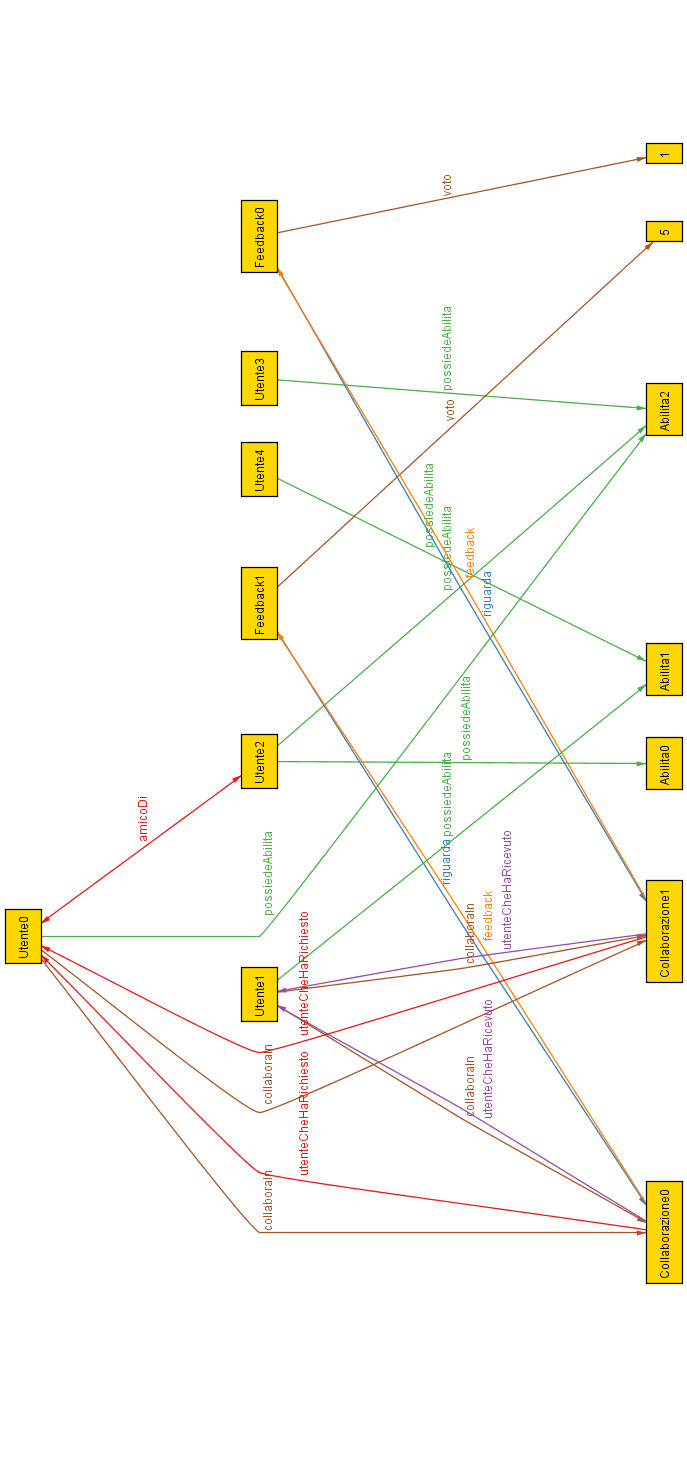


Fig. 5.1 – Mappa di un’istanza con Alloy Analyzer

# Indice delle figure

Fig. 1.1 – Diagramma di Jackson-Zave 4

Fig. 3.1 - Use Case Diagram 18

Fig. 3.2 - Sequence Diagram - Scenario 1 19

Fig. 3.3 - Sequence Diagram - Scenario 2 19

Fig. 3.4 - Sequence Diagram - Scenario 3 20

Fig. 3.5 - Sequence Diagram - Scenario 4 21

Fig. 3.6 - Sequence Diagram - Scenario 5 22

Fig. 3.7 - Sequence Diagram - Scenario 6 23

Fig. 3.8 - Sequence Diagram - Scenario 7 24

Fig. 3.9 - Sequence Diagram - Scenario 8 25

Fig. 3.10 - Sequence Diagram - Scenario 9 25

Fig. 3.11 – BMPN – Scenario 2 26

Fig. 3.12 – BMPN – Scenario 3 27

Fig. 3.13 – BMPN – Scenario 4 28

Fig. 3.14 – BMPN – Scenario 5 29

Fig. 3.15 – BMPN – Scenario 6 30

Fig. 3.16 – BMPN – Scenario 7 31

Fig. 3.17 – BMPN – Scenario 8 32

Fig. 3.18 – BMPN – Scenario 9 33

Fig. 3.19 - Class Diagram 34

Fig. 5.1 – Mappa di un’istanza con Alloy Analyzer 43