Azzolini Riccardo 2019-10-28

Use case diagram

1 Requisiti

Il punto di partenza del processo di sviluppo è la definizione dei requisiti. Essa viene solitamente fatta in linguaggio naturale e in modo informale, ma UML permette di modellare i requisiti in modo semi-formale, mediante gli **use case diagram**.

2 Use case diagram

Gli use case diagram elencano le funzionalità fornite, a vari livelli (a seconda delle esigenze):

- dall'intero sistema;
- da un certo sottosistema;
- da una singola classe.

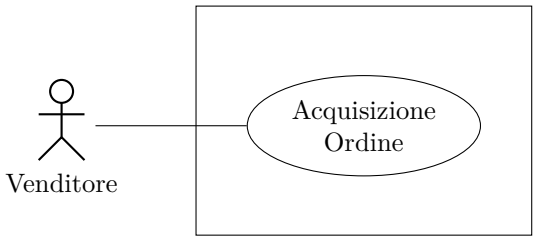
In particolare, definiscono il comportamento del sistema (/sottosistema), cioè le funzionalità principali e come il sistema agisce e reagisce.

Questi diagrammi evidenziano le relazioni esistenti tra gli **attori** (entità esterne, che vengono modellate, perché alcune loro caratteristiche potrebbero essere utili, ma non implementate) e gli **use case** (funzionalità del software usate dagli utenti). Più in generale, essi descrivono:

- il sistema;
- l'ambiente;
- le relazioni tra sistema e ambiente;
- le relazioni interne tra le varie funzionalità del sistema.

Lo use case diagram è un diagramma *statico*: mostra le funzionalità del sistema, ma non specifica quando esse vengono eseguite, in che ordine, ecc.

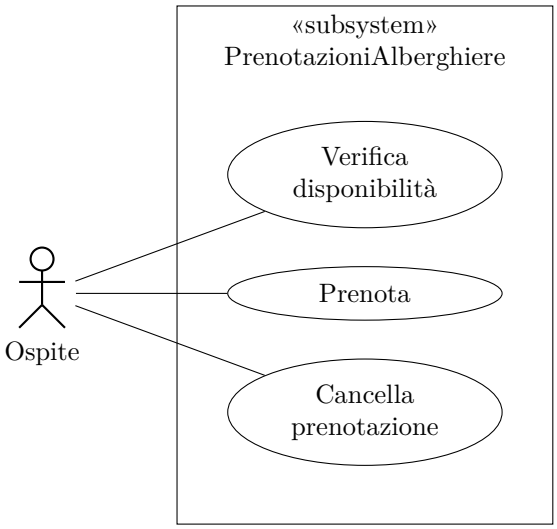
2.1 Esempi



In questo diagramma:

- "Venditore" è un attore;
- "Acquisizione Ordine" è uno use case;
- il rettangolo intorno allo use case indica il confine del sistema, ed è facoltativo (può essere lasciato implicito).

Per rappresentare un sottosistema, si usa lo stereotipo «subsystem»:



3 Use case

Uno **use case** rappresenta una funzione visibile dall'utente, mostrando una tipica interazione tra l'utente e il sistema informatico. Esso consente quindi di individuare il comportamento (in termini, appunto, di funzionalità offerte) di un'entità appartenente al sistema, senza entrare nel dettaglio della struttura interna di tale entità.

In prima approssimazione, la funzione rappresentata da uno use case deve essere una che dà valore aggiunto all'utente: ad esempio, "identifica utente", e non "inserisci nome", "inserisci cognome", ecc.

Gli use case possono essere di dimensioni diverse (per esempio, si potrebbero rappresentare sia la semplice identificazione di un utente che la gestione di una transazione complessa), e possono essere descritti con vari livelli di dettaglio.

Ogni use case deve essere collegato almeno a un attore (o a un altro use case, mediante apposite relazioni).

3.1 Acquisizione

Gli use case si acquisiscono solitamente parlando con gli utenti. A tale scopo, gli use case diagram sono pensati per essere abbastanza informali da poter essere presentati anche agli utenti (ma hanno alcune caratteristiche, come ad esempio la relazione di ereditarietà, che sono più orientate ai progettisti/sviluppatori).

Ogni use case identificato viene:

- etichettato con un nome/titolo;
- descritto graficamente;
- descritto con del testo, per integrare la rappresentazione grafica.

3.2 Elementi caratteristici

Ciascuno use case è caratterizzato da:

- un *titolo*, che descrive sinteticamente lo scopo del caso d'uso;
- un *inizio*, l'azione che inizia l'interazione tra attore e sistema;
- uno *sviluppo intermedio*;
- una *fine*, individuata da delle specifiche condizioni di terminazione.

Mentre il nome di una classe è tipicamente un sostantivo singolare, il titolo di uno use case è solitamente una frase verbale (es. "identifica utente"), ma si possono usare anche i "nomi d'azione" (es. "identificazione utente"). In ogni caso, non sono accettabili i titoli eccessivamente generici, come ad esempio "gestisci sistema", perché essi non danno informazioni sufficienti.

3.3 Livello di dettaglio

Per loro natura, gli use case non sono ricchi di dettagli.

I dettagli possono sempre essere aggiunti dopo, ma è importante esplorare subito le ramificazioni dei cambiamenti che potrebbero derivare da tali aggiunte.

4 Attori

Un **attore** è un'entità esterna al sistema, che interagisce con il sistema, assumendo un particolare ruolo:

- controlla le funzionalità del sistema, fornendo input puntuali che permettono al sistema di prendere decisioni;
- fornisce input e/o riceve output dal sistema.

Siccome un attore rappresenta un ruolo, non c'è necessariamente una corrispondenza tra attori e individui precisi:

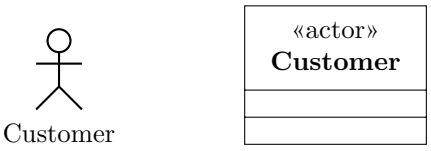
- possono esserci diversi utenti con lo stesso ruolo;
- uno stesso utente può ricoprire ruoli diversi;
- nonostante l'icona standard usata per raffigurarlo, un attore non corrisponde per forza a una o più persone, ma può invece essere una qualsiasi entità esterna (come anche, ad esempio, una base di dati, ecc.).

Inoltre, attori diversi possono esercitare lo stesso use case, e uno stesso attore può esercitare use case diversi.

Tra gli attori possono esistere relazioni di generalizzazione/specializzazione:

4.1 Rappresentazione grafica

Un attore non è altro che uno stereotipo di una classe. Esso può quindi essere rappresentato in due modi equivalenti:



È anche possibile definire stereotipi più specifici per particolari tipi di attori (ad esempio, una base di dati).

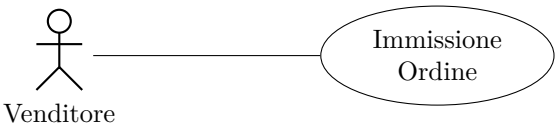
4.2 Funzione

Oltre a rappresentare i confini del sistema, gli attori sono utili, durante l'analisi, per individuare gli use case: spesso è più facile individuare una lista di attori, e poi capire

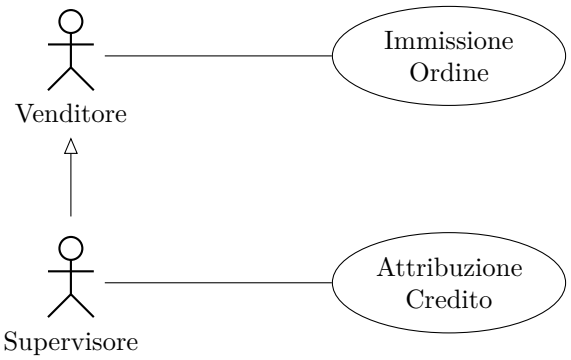
- quali siano i loro obiettivi;
- di quali interazioni col sistema essi debbano essere protagonisti.

5 Relazioni tra attori e use case

La partecipazione di un attore a uno use case è identificata da un'**associazione**:

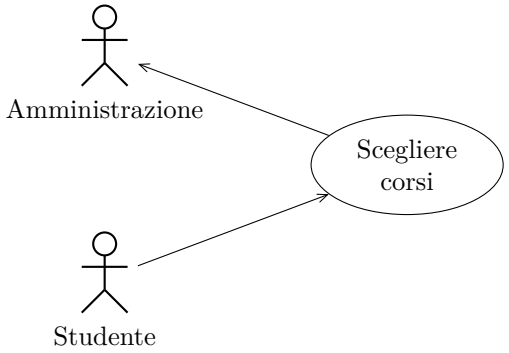


In presenza di generalizzazioni tra attori, l'attore specializzato può partecipare a tutti gli use case ai quali partecipa l'attore più generale. Ad esempio, nel diagramma

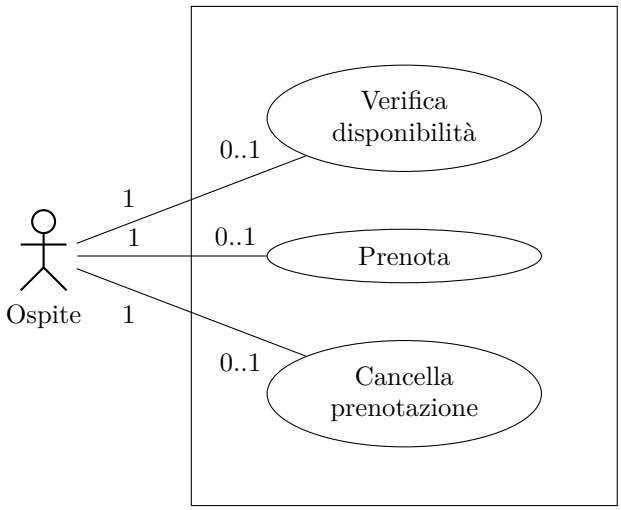


l'attore "Supervisore" può partecipare sia a "Immissione Ordine" che a "Attribuzione Credito".

Opzionalmente, si può usare una freccia per indicare la direzione prevalente delle informazioni:



Infine, è anche possibile specificare le molteplicità delle associazioni:



6 Documentazione di uno use case

Siccome la rappresentazione di uno use case è priva di dettagli, esso deve essere accompagnato da una documentazione testuale, scritta in linguaggio naturale, ma organizzata preferibilmente in modo strutturato:

- Titolo.
- Autori (utili per sapere a chi chiedere chiarimenti riguardo allo use case).
- Descrizione: una breve descrizione testuale della funzionalità.
- Requisiti: descrive un "contratto" che lo use case deve rispettare quando viene "eseguito", effettuando un'azione o fornendo un valore al sistema.
- Attori: si descrivono quali sono gli attori con cui lo use case è in associazione.
- Vincoli: forniscono una specifica di cosa fa il sistema, agli effetti esterni, quando lo use case viene eseguito:
  - precondizioni: devono essere soddisfatte prima che lo use case possa essere eseguito;
  - postcondizioni: devono essere soddisfatte dopo l'esecuzione dello use case;
  - invarianti: devono essere soddisfatti durante l'esecuzione dello use case.
- Scenari: descrizioni delle sequenze di eventi che caratterizzano il caso generale (scenario base), ed eventualmente dei casi particolari[1] (scenari alternativi), come ad esempio le situazioni di errore. Per ogni scenario, bisogna descrivere:
  1. come inizia;
  2. i singoli passi intermedi;
  3. come termina.

6.1 Scenari

Uno use case individua una tipologia di "storie" d'uso del sistema. Lo stesso use case può quindi avere **istanze** (varianti) diverse, ciascuna delle quali corrisponde a una particolare storia d'uso, ed è caratterizzata da una specifica sequenza di azioni.

Uno **scenario** individua e descrive esplicitamente una particolare istanza di uno use case. Ogni use case è caratterizzato da uno scenario base (la sequenza tipica di eventi), e da un numero, imprecisato a priori, di varianti. Le varianti possono infatti essere aggiunte anche nei successivi raffinamenti dello use case.

[1] È bene specificare i casi particolari, ma, per una funzionalità corposa, non è realisticamente possibile elencarli tutti.

Nella documentazione dello use case, è necessario descrivere ogni scenario rilevante. Tale descrizione:

- deve essere effettuata dal punto di vista degli attori, dettagliando ciò che il sistema deve fornire quando lo use case viene eseguito;
- può essere informale (in linguaggio naturale), o mista (cioè composta sia da una descrizione testuale, che da una descrizione formale, realizzata con altri linguaggi).

In particolare, per descrivere i singoli scenari è possibile utilizzare gli *interaction diagram* di UML:

- sequence diagram;
- communication diagram.

6.1.1 Ruolo degli scenari

L'individuazione di uno use case definisce un requisiti ad alto livello, dal punto di vista dell'utente.

Raccogliendo poi gli scenari, si identificano i requisiti a un maggior livello di dettaglio. Solitamente, in questo passo si introduce anche la descrizione degli elementi del dominio del problema.

Infine, con l'analisi degli scenari allo scopo di generalizzazione e sintesi, si ottiene la **specifica** dei requisiti del sistema software: cosa deve fare il sistema, tenendo conto dei requisiti utente e delle caratteristiche del dominio applicativo.

Mediante gli scenari, le specifiche dei requisiti vengono espresse indicando delle sequenze di azioni. Esiste quindi il rischio di fissare "troppo presto" come funziona il sistema (invece di limitarsi al "cosa"), ma esso è facilmente evitabile se si rispettano i principi fondamentali degli use case, descrivendo solo i comportamenti che l'utente desidera, cioè quelli che hanno un effetto desiderato su uno o più oggetti del dominio del problema: tali comportamenti non sono soggetti a scelte progettuali, in quanto necessari per soddisfare i requisiti.

7 Relazioni tra use case

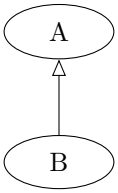
Tra gli use case possono essere definite relazioni di:

generalizzazione: simile alla generalizzazione tra classi;

inclusione: il comportamento individuato dallo use case target viene incluso nella sequenza di azioni corrispondente allo use case base;

estensione: le funzionalità individuate da uno use case base vengono estese, al verificarsi di determinate condizioni, con funzionalità definite in un altro use case.

7.1 Generalizzazione



Lo use case figlio (B) eredita tutti gli

- attributi
- scenari
- extension point

definiti nello use case padre (A), e partecipa a tutte le relazioni in cui è coinvolto il padre. In più, esso può

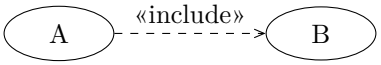
- partecipare anche ad altre relazioni;
- prevedere nuovi scenari, non definiti nel padre;
- ridefinire scenari presenti nel padre.

Uno use case:

- può ereditare da n altri use case (cioè è ammessa l'ereditarietà multipla);
- può essere il padre di n altri use case.

7.2 Inclusione

Possono esistere alcune funzionalità del sistema che servono, vengono "richiamate", per lo svolgimento di varie altre funzionalità. Per evitare di ripeterne la descrizione in ogni use case che le usa, tali funzionalità ricorrenti possono essere messe a fattor comune, indicando che vengono "incluse" negli use case più complessi che le utilizzano.



Questa relazione indica che la funzionalità individuata dallo use case target (B) viene inclusa in un punto specifico, e nella sua interezza, nella sequenza di azioni che caratterizza lo use case base (A), in modo analogo a una chiamata di procedura:

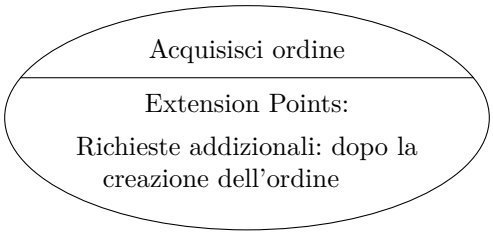
1. quando un'istanza dello use case base raggiunge il punto di inclusione, vengono svolte tutte le azioni previste dallo use case incluso;
2. successivamente, si riprende lo svolgimento delle azioni previste dallo use case base.

UML non fornisce un modo particolare per indicare il punto di inclusione, quindi questo deve essere indicato nella descrizione testuale degli scenari associati allo use case.

Gli use case che rappresentano funzionalità ricorrenti possono anche non essere direttamente in relazione con alcun attore.

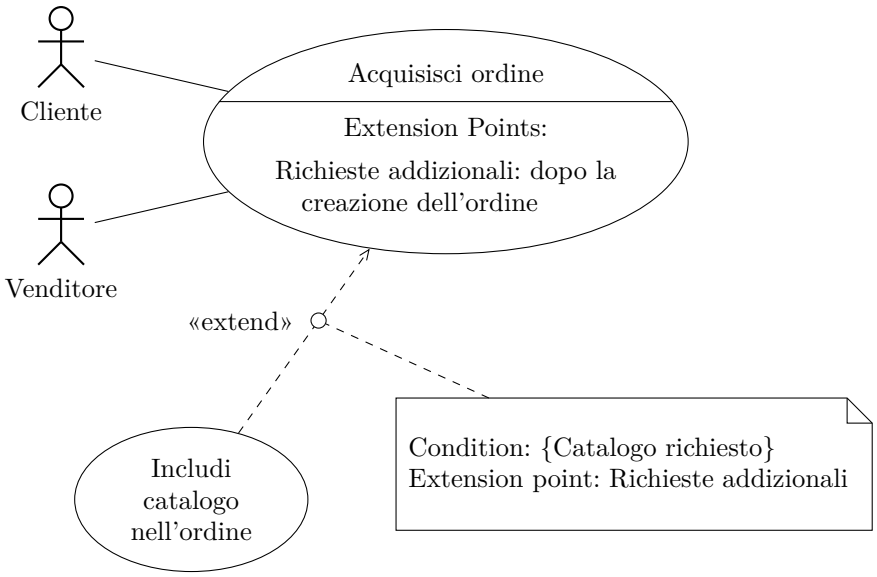
7.3 Estensione

Oltre al nome di uno use case, è possibile elencare degli **extension point**: dei punti in corrispondenza dei quali è possibile inserire attività aggiuntive, estendendo così lo use case base. Ad esempio:



La relazione «extend» connette uno use case estensione allo use case base, specificando:

- una condizione che deve essere verificata affinché abbia luogo l'estensione;
- l'extension point nel quale si inseriscono le attività aggiuntive specificate dallo use case estensione.

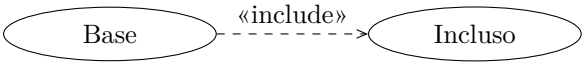


Quando uno scenario associato allo use case base raggiunge l'extension point specificato dalla relazione «extend», e la condizione per l'estensione è verificata, lo use case estensione "si inserisce" nell'esecuzione dello use case base, quindi viene inglobato nel comportamento base quello previsto dall'estensione.

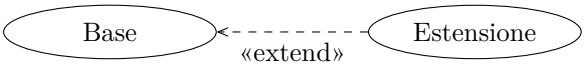
Lo use case estensione può essere uno use case vero e proprio, oppure solo un "frammento" di use case, che non è autonomamente istanziabile (cioè non può essere eseguito come use case autonomo, ma solo come estensione).

Nota: Graficamente, la freccia della relazione di estensione punta dallo use case esteso a quello base, ovvero è orientata nella direzione opposta rispetto a quella dell'inclusione. In generale, la direzione delle frecce di queste due relazioni corrisponde all'ordine soggetto–verbo–oggetto, dove il verbo è «include» o «extend»:

- lo use case base *include* («include») lo use case incluso;

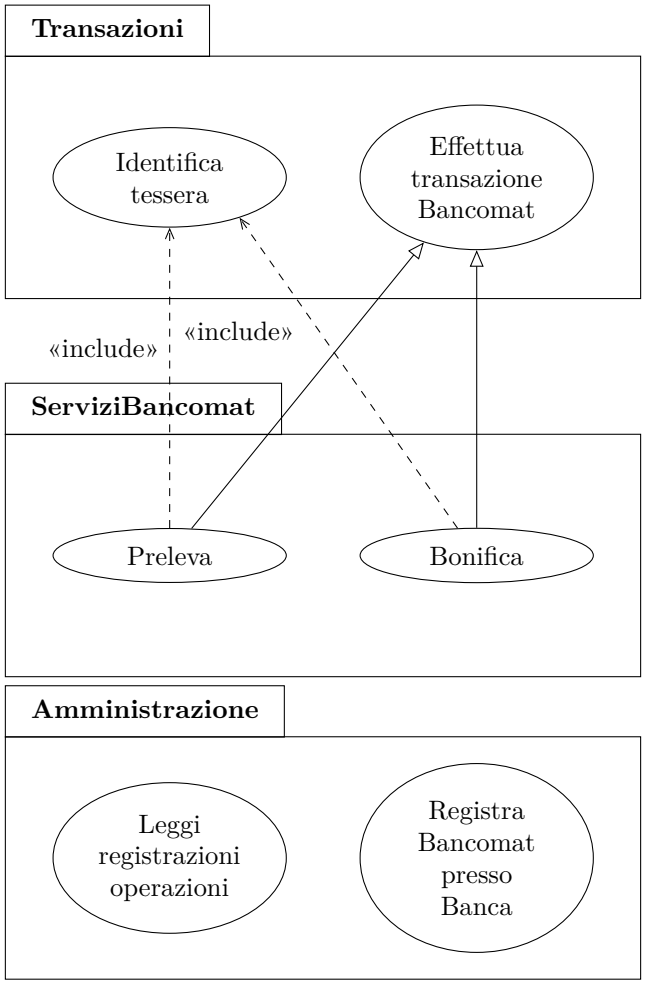


- lo use case estensione *estende* («extend») lo use case base.



8 Organizzazione in package

Gli use case possono essere raggruppati in package. Di fatto, così facendo, si divide la specifica dei requisiti in "capitoli". Ad esempio:



Nota: In questo diagramma, gli attori sono stati omessi solo per semplificare l'esempio. In un diagramma reale, sarebbe necessario rappresentarli.

Osservazione: Tutte le parti di un sistema devono essere collegate, ma ciò non è vero per gli use case diagram: siccome essi rappresentano dei requisiti, l'importante è che i vari use case siano presenti, e che ciascuno di essi sia collegato almeno a un attore (o a un altro use case, ad esempio nel caso dell'inclusione).

9 Quando usare gli use case

L'identificazione degli use case è la prima cosa da fare, dato che essi:

- rappresentano i requisiti di sistema;
- forniscono una base per la pianificazione e il controllo dei progetti.

È però normale continuare a identificare nuovi use case durante tutta la modellazione concettuale del sistema.

10 Granularità degli use case

Una granularità fine facilita la pianificazione del lavoro, rendendo più chiaro "cosa c'è da fare". Infatti, nel processo di sviluppo basato su UML, gli use case (o meglio, i gruppi di use case logicamente correlati) costituiscono le unità di lavoro su cui si può basare il piano di sviluppo.

Un livello di dettaglio eccessivo, però, rende difficile avere una visione d'insieme, sintetica, del sistema.

11 Definizione degli use case

La scrittura degli use case richiede una buona comprensione del dominio del problema. L'analisi del dominio, l'analisi dei requisiti, e la documentazione attraverso use case e testo associato, devono quindi andare di pari passo.

Il processo di definizione degli use case è iterativo:

1. si identificano i comportamenti principali e più semplici;
2. si descrivono i comportamenti alternativi e più complessi.

Esso termina al raggiungimento di un livello di dettaglio buono (che facilita tutte le attività successive), ma non eccessivo, altrimenti:

- si complicherebbe inutilmente la descrizione;
- si introdurrebbero prematuramente delle scelte progettuali;
- risulterebbe difficile avere una visione d'insieme.

12 Requisiti non funzionali

Gli use case *non sono adatti* a specificare i requisiti non funzionali. Siccome questi sono comunque di fondamentale importanza, è necessario descriverli in un altro modo:

- attraverso un'apposita specifica testuale, allegata agli use case diagram;
- mediante diagrammi di tipi diversi.

13 Altre funzioni degli use case

Oltre all'analisi dei requisiti, gli use case sono utili anche per:

- la *convalida del sistema*: permettono di ricavare facilmente dei casi di test;
- la *gestione del progetto*: possono essere usati per stimare la complessità (un numero più elevato di use case corrisponde, in media, a un maggiore effort di sviluppo, anche se la dimensione di ciascuno use case è variabile) e per organizzare il progetto.