

I diagrammi dei casi d'uso (use case diagram) rappresentano una vista statica delle funzionalità fornite dal sistema. Essi illustrano le relazioni tra singole funzionalità e tra una particolare funzionalità e una o più entità esterne al sistema (attori)

Diagrammi dei casi d'uso

Introduzione

La maggior parte dei testi che trattano la modellazione basata sui casi d'uso in UML pongono molta enfasi sui diagrammi, limitandosi a descrivere gli elementi sintattici che sono in essi rappresentati (sistema, attori, casi d'uso e relazioni), e trascurando quasi del tutto il livello di descrizione testuale. Seppure le descrizioni testuali non sono contemplate in UML, esse nondimeno costituiscono una parte fondamentale della modellazione di specifiche funzionali. In controtendenza con questi testi, quindi, iniziamo questo articolo affermando che i diagrammi sono – in questo particolare caso – la parte meno rilevante per un progettista: assolutamente più utili sono i casi d'uso espressi in forma narrativa, ossia le descrizioni testuali che illustrano ciascuno scenario di utilizzo di una funzionalità del sistema da parte di un utente. Per tale ragione, oltre a fornire i fondamenti di come si disegna un diagramma dei casi d'uso, cercheremo di inquadrare il ruolo dei diagrammi nella modellazione e chiuderemo l'articolo proponendo un template per la loro documentazione.

Casi d'uso

I casi d'uso costituiscono un modello funzionale di un sistema. Più in particolare, essi descrivono i requisiti funzionali che il sistema fornisce ai suoi utenti. La modellazione basata sui casi d'uso è stata proposta originariamente da Jacobson [Jacobson, 1992]. Secondo tale approccio, i requisiti funzionali di un sistema sono definiti in termini delle interazioni tra gli attori e il sistema stesso. Gli attori sono entità esterne al sistema che con esso interagiscono. Essi solitamente sono costituiti da utenti, ossia persone fisiche che utilizzano il sistema. Un attore nell'accezione più generale, tuttavia, può essere costituito anche da entità fisiche non umane (altri sistemi software o hardware collegati al sistema in esame), oppure da entità astratte (un timer che attiva una funzionalità del sistema allo scadere di un intervallo di tempo prestabilito, il verificarsi di un particolare evento, ecc.). Un caso d'uso definisce una sequenza di interazioni tra il sistema e uno o più attori, indipendentemente dalla loro natura [Gomaa, 2000]. Il modello dei casi d'uso viene elaborato mantenendo una prospettiva di tipo black-box: ogni interazione viene descritta sulla base della funzionalità fornita all'attore. Non viene svelato come tale funzionalità sia implementata. In altre parole, un caso d'uso definisce il comportamento di un qualche aspetto del sistema come osservato da un osservatore esterno, senza rivelarne la struttura interna.

In UML ogni caso d'uso è rappresentato mediante un'ellisse e un nome che caratterizza l'interazione stessa, mentre un attore è rappresentato attraverso l'icona di un omino stilizzato, come illustrato in figura 1. L'interazione tra l'attore e il caso d'uso (e, indirettamente, tra l'attore e il sistema che implementa il caso d'uso) è rappresentata mediante una linea che collega i due elementi, mettendoli in associazione fra loro.

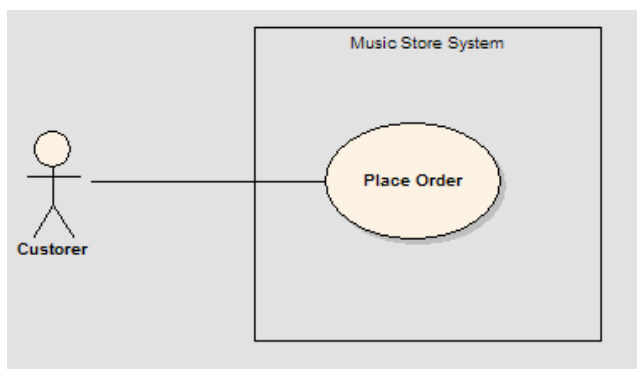


Figura 1 – Attore e caso d'uso

La scelta del nome di un caso d'uso è importante poiché ha un potere evocativo fondamentale: essa deve comunicare al lettore del diagramma la funzionalità che il caso d'uso rappresenta. Oltre questo succinto livello di descrizione testuale, infatti, il diagramma dei casi d'uso non fornisce alcun ulteriore contributo nel comprendere quali siano i passi principali di ciascuna interazione, quali siano le possibili cause di fallimento, quali le sequenze di passi alternativi rispetto allo scenario principale¹. Tutti questi aspetti vanno documentati nella descrizione testuale del caso d'uso, come descritto nel paragrafo "Documentare i casi d'uso" alla fine dell'articolo.

Attori, ruoli e utenti

Un attore in UML caratterizza un'utente o un insieme di utenti esterni al sistema che con esso interagiscono [Rumbaugh et al, 1999]. Una distinzione abbastanza comune consiste nel differenziare gli attori in primari e secondari, in base al loro livello di partecipazione al caso d'uso. Un *attore primario* attiva (o inizia) il caso d'uso e, quindi, è rivestito di un ruolo proattivo. Viceversa, un *attore secondario* ha un ruolo marginale poiché partecipa al caso d'uso solamente come sorgente di dati in input oppure come ricevitore di dati in output. L'attore secondario ha quindi un ruolo di supporto al caso d'uso, il quale è concepito per fornire una funzionalità il cui valore è percepito soprattutto dall'attore primario. Un database esterno al sistema è un esempio tipico di attore secondario che riceve o fornisce dati ma che non utilizza altrimenti il sistema. Un cliente di uno sportello ATM, come illustrato in figura 1, è chiaramente un attore primario poiché è colui che trae il principale valore dall'utilizzo del sistema, attivando direttamente ciascun caso d'uso. Non sempre, comunque, ricevere valore dall'utilizzo di una funzionalità e iniziare un caso d'uso si fondono in un unico attore. Nei sistemi embedded e nei sistemi real-time, ad esempio, la distinzione tra i due tipi di attore non è sempre così marcata, poiché tali sistemi sono caratterizzati da una scarsa partecipazione di attori umani. In essi un attore primario può essere rappresentato anche da un dispositivo di I/O o da un timer e, viceversa, un utente umano che si limita a ricevere alcune informazioni dal sistema può essere un attore secondario. In questi casi, non è tanto chi riceve il maggior valore dall'utilizzo del sistema, quanto chi attiva lo scenario di esecuzione che permette la distinzione tra attori primari e secondari.

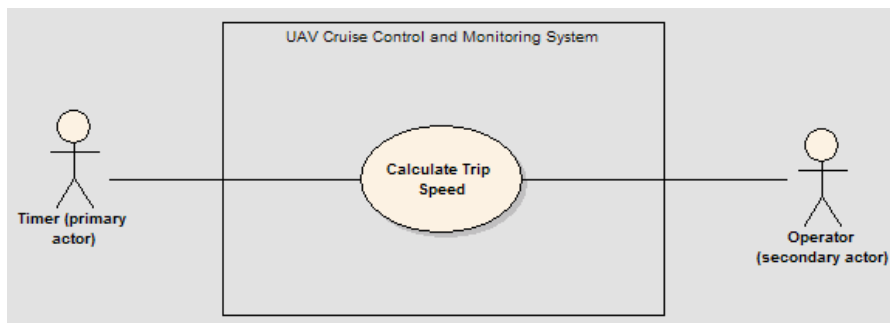


Figura 2 – Attore primario e attore secondario in un tipico sistema embedded

La figura 2 illustra un esempio tipico di attore primario rappresentato da un timer che, ad intervalli regolari, attiva il calcolo della velocità di crociera di un aereo teleguidato, il cui valore viene visualizzato ad un operatore alla base di controllo (i dettagli che specificano come questa informazione dal velivolo viene trasmessa all'operatore nella base sono aspetti interni del sistema, di cui il modello funzionale basato sui casi d'uso non deve tenere conto).

Concludiamo il paragrafo con un'ultima osservazione su attori e ruoli. Un attore rappresenta tipicamente un ruolo svolto da un utente del sistema nel dominio applicativo analizzato. Un utente, tuttavia, è anche un individuo fisico. Un errore abbastanza comune consiste nel confondere gli attori con le persone fisiche. Un attore rappresenta un ruolo svolto da *tutti* gli utenti di una determinata categoria. La differenza rispetto ad una persona fisica è che quest'ultima può svolgere le mansioni di diversi ruoli. Ad esempio uno stesso individuo può svolgere sia il ruolo di operatore, sia quello di pilota nell'esempio del sistema di controllo di un velivolo teleguidato. In questo caso, pur essendo sempre la stessa persona ad interagire con il sistema, i ruoli da essa ricoperti sono due e tali, quindi, devono essere anche gli attori. Passando da operatore a pilota, l'utente si toglie il cappello di operatore per mettersi quello di pilota, con il quale "vede" il sistema da una prospettiva diversa in termini di funzionalità (casi d'uso) che può esercitare.

¹ Si può obiettare che nella sintassi UML è comunque possibile inserire descrizioni testuali mediante delle note. Tuttavia la quantità di informazione necessaria per documentare adeguatamente un caso d'uso è generalmente di gran lunga superiore allo spazio che una nota potrebbe ragionevolmente occupare in un diagramma. La nota, dopotutto, dovrebbe essere un elemento marginale rispetto all'intero diagramma, e non il suo componente principale.

Relazioni tra casi d'uso

Dalla versione 1.4 di UML le relazioni standard tra casi d'uso sono le seguenti:

- Inclusione;
- Estensione;
- Generalizzazione.

La relazione di *inclusione* (stereotipata mediante «include») viene utilizzata quando si vuole riutilizzare un caso d'uso all'interno di un altro. Facendo un'analogia con i linguaggi di programmazione, tale relazione è simile all'invocazione di funzioni. La sua utilità più importante consiste nel fornire al progettista uno strumento di decomposizione di un caso d'uso complesso in casi d'uso più semplici (se state pensando di essere vicini alla decomposizione funzionale non vi sbagliate affatto...). Una caratteristica importante che contraddistingue la relazione di inclusione è che essa "è per sempre". Quando un caso d'uso (che chiamiamo caso d'uso base) include un altro caso d'uso (che chiamiamo caso d'uso incluso), l'inclusione avviene sempre, in ogni circostanza! Non esistono flussi alternativi dello stesso caso d'uso base nel quale non si verifichi l'inclusione del caso d'uso incluso. In figura 3 è illustrato il caso d'uso di identificazione di un velivolo da parte di una stazione di controllo di terra, sempre con riferimento al sistema descritto nel paragrafo precedente.

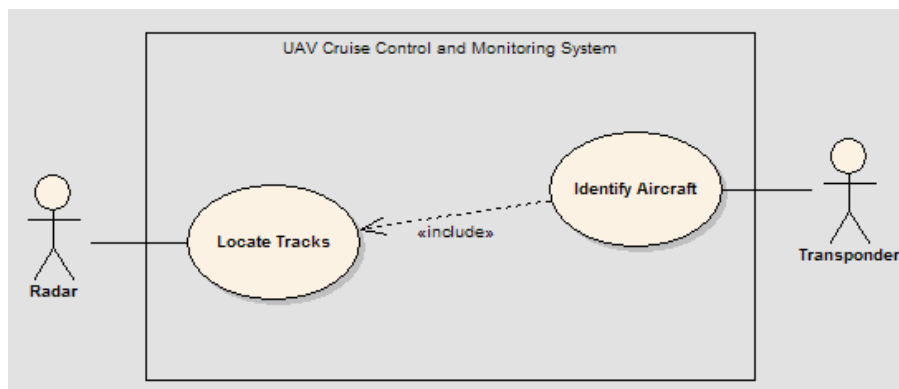


Figura 3 – Inclusione tra casi d'uso

Il diagramma afferma esplicitamente che ad ogni esecuzione di uno scenario di identificazione di un velivolo (*Identify Aircraft*) viene eseguito anche lo scenario di localizzazione della traccia di un aereo (*Locate Tracks*). Per identificare un velivolo il sistema deve interagire con un *transponder* collocato a bordo dell'aereo, mentre per localizzare una traccia è necessaria l'interazione con il *radar*. Senza la localizzazione di una traccia, tuttavia, non è possibile identificare il velivolo.

Un caso d'uso viene normalmente descritto (in forma testuale) mediante una sequenza di passi che realizzano il principale scenario d'esecuzione che termina con successo (successful business use case scenario). All'interno di un caso d'uso base, però, possono verificarsi delle eccezioni o, più in generale, delle estensioni che sono caratterizzate dall'aggiunta di ulteriori passi, qualora si verifica una particolare condizione. Il verificarsi di un'estensione provoca una momentanea "deviazione" dal flusso principale descritto nel caso d'uso base. Questa deviazione è chiamata *percorso alternativo* ed è rappresentata mediante la relazione di estensione (stereotipo «extend»). Il caso d'uso estensione aggiunge un nuovo comportamento al caso d'uso base, rappresentando quindi un'estensione della logica di base di quest'ultimo. In altre parole, il caso d'uso estensione continua il comportamento del caso d'uso base, inserendo azioni alternative. L'inserimento di tali azioni non è casuale, ma avviene in corrispondenza di un punto ben definito all'interno del caso d'uso base, chiamato *punto di estensione*. Ogni caso d'uso di base dichiara tutti i suoi punti di estensione. Un caso d'uso estensione può estendere più di un punto d'estensione. La relazione di estensione è simile alla gestione degli interrupt hardware:

- viene eseguito un caso d'uso base;
- il flusso d'esecuzione arriva ad un punto di estensione;
- si verifica una condizione che provoca una deviazione dal percorso base;
- il controllo viene ceduto al caso d'uso estensione che gestisce il nuovo percorso;
- viene completato il caso d'uso estensione;
- il controllo viene infine ritornato al caso d'uso base che riprende la sua esecuzione dal punto d'estensione in poi.

In figura 4 è illustrato un esempio di relazione di estensione, tratto dalla modellazione di un sistema di controllo del traffico aereo. Lo scopo di tale sistema consiste nel fornire ai controllori di volo della stazione di controllo tutte le informazioni necessarie per coordinare il traffico aereo durante gli atterraggi e i decolli. Una delle funzionalità principali del sistema è la visualizzazione della traiettoria di volo di ciascun velivolo (*Display Flight Path*). Se l'aereo può atterrare senza alcun pericolo di collisione con altri velivoli in atterraggio o in decollo, il caso d'uso base viene eseguito dall'inizio alla fine, senza deviazioni dal percorso principale. Se, invece, una traiettoria di collisione con un altro velivolo viene individuata dal sistema, viene attivato il caso d'uso estensione che fa scattare degli allarmi, a seconda della gravità dell'evento. Un primo livello di allarme, descritto nella figura 4 dal caso d'uso estensione *Display Collision Trajectory* visualizza mediante un colore rosso tale traiettoria sullo schermo del controllore di volo. Un secondo livello di allarme, non illustrato in figura, può essere l'attivazione di allarmi sonori.

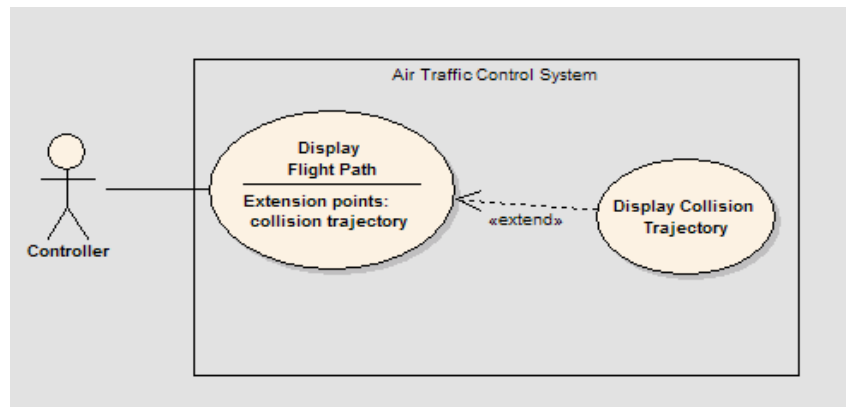


Figura 4 – Estensione di un caso d'uso

Come si nota dalla figura, il caso d'uso base dichiara esplicitamente i punti di estensione. Esistono due stili piuttosto comuni di dichiarazione di un punto di estensione:

- mediante espressioni condizionali (Extension points: if collision trajectory has been detected);
- mediante sostantivi (Extension points: collision trajectory)

Il primo stile è più dettagliato, ma ha lo svantaggio di rendere il diagramma maggiormente ingombrante; il secondo stile è più evocativo, lasciando la responsabilità al progettista di identificare nomi che implicitamente facciano intuire lo scenario alternativo. L'autore predilige proprio quest'ultimo stile poiché rende i diagrammi più compatti e semplici. Va inoltre considerato che per descrivere adeguatamente un caso d'uso è comunque necessario far ricorso alle descrizioni testuali (o narrative). Questo tipo di informazione, di conseguenza, trova una sua collocazione più efficace nel testo.

«Include» vs. «extend»	
Aumenta il comportamento di un caso d'uso base	Aumenta il comportamento di un caso d'uso base
Il caso d'uso incluso è <i>sempre</i> utilizzato per aumentare il comportamento del caso d'uso base	Il caso d'uso estensione <i>può</i> essere utilizzato per aumentare il comportamento del caso d'uso base
Il <i>caso d'uso base</i> decide quando invocare il caso d'uso incluso. Il caso d'uso incluso non è a conoscenza del caso d'uso base	Il <i>caso d'uso estensione</i> decide quando inserire il proprio comportamento nel caso d'uso base. Il caso d'uso base non è a conoscenza del caso d'uso estensione
La dipendenza rappresentata da una linea tratteggiata adornata dallo stereotipo «include» è orientata dal caso d'uso base a quello incluso, richiamando l'analogia di un programma che invoca una procedura	La dipendenza rappresentata da una linea tratteggiata adornata dallo stereotipo «extend» è orientata dal caso d'uso estensione a quello base, richiamando l'analogia di un interrupt software o di una callback function

Tabella 1 – Confronto tra le relazioni di inclusione e di estensione

I principianti che si avvicinano per la prima volta alla modellazione dei casi d'uso mediante UML talvolta confondono le relazioni di inclusione e di estensione, in particolare se non hanno familiarità con la gestione delle eccezioni (che rappresenta un'analogia molto forte con la relazione di estensione). In tabella 1 è proposto un confronto tra inclusione ed estensione [Pender, 2003] per favorire la padronanza di questi due concetti diversi.

L'ultima relazione esaminata in questo articolo è la *generalizzazione*. La abbiamo lasciata per ultima poiché è la relazione sulla quale esiste un minor consenso unanime sul suo significato e sulla sua reale utilità. Alcuni esperti ritengono che la generalizzazione sia semplicemente un'altra relazione per descrivere percorsi alternativi [Fowler&Kendall, 1999], favorendo a mio giudizio la confusione che può allora essere fatta con l'estensione, modellata proprio in termini di percorsi alternativi. Per tale ragione, altri esperti [Gomaa, 2000] evitano del tutto l'utilizzo di entrambe le relazioni, preferendone una ed eliminando dal proprio vocabolario l'altra. La maggiore confusione nasce dal fatto che entrambe le relazioni implicano una variazione dal comportamento definito nel caso d'uso base. Per chiarire il concetto, diciamo che la generalizzazione è simile all'ereditarietà tra classi, laddove esiste un metodo generico definito nella classe base, ed una sua particolare implementazione (specializzazione) in una classe derivata. Nella modellazione dei casi d'uso esiste analogamente un caso d'uso generico che viene specializzato da un caso d'uso derivato. Nel caso d'uso derivato viene estesa o specializzata la sequenza di attività che definisce il caso d'uso base. Un elemento essenziale che permette di distinguere la generalizzazione dall'estensione è l'assenza nella prima forma di relazione dei punti di estensione. Un caso d'uso base nella generalizzazione non viene mai interrotto a causa dell'invocazione di un caso d'uso derivato. Non solo, ma a tempo d'esecuzione non coesistono mai caso d'uso derivato e caso d'uso base: fin dall'inizio viene eseguito l'uno oppure l'altro (a differenza dell'estensione che prevede, nelle circostanze in cui si attiva il caso d'uso estensione, la coesistenza di entrambi i casi d'uso). In base all'esperienza personale, tuttavia, l'utilizzo della generalizzazione dovrebbe essere fatto con prudenza, poiché a fronte delle possibili confusioni che può generare, non fornisce un sostanziale apporto alla definizione delle specifiche funzionali del sistema. Essa ha un valore organizzativo, raggruppando all'interno di una gerarchia casi d'uso simili. L'utilizzo che risulta più chiaro di tutti è quello in cui il caso d'uso base è astratto, mentre i casi d'uso derivati sono forme concrete di specializzazioni del caso d'uso base. La figura 5 illustra questo tipo di specializzazione, riproponendo il caso d'uso base *Place Order* descritto in figura 1 nella modellazione di un negozio on-line di articoli musicali.

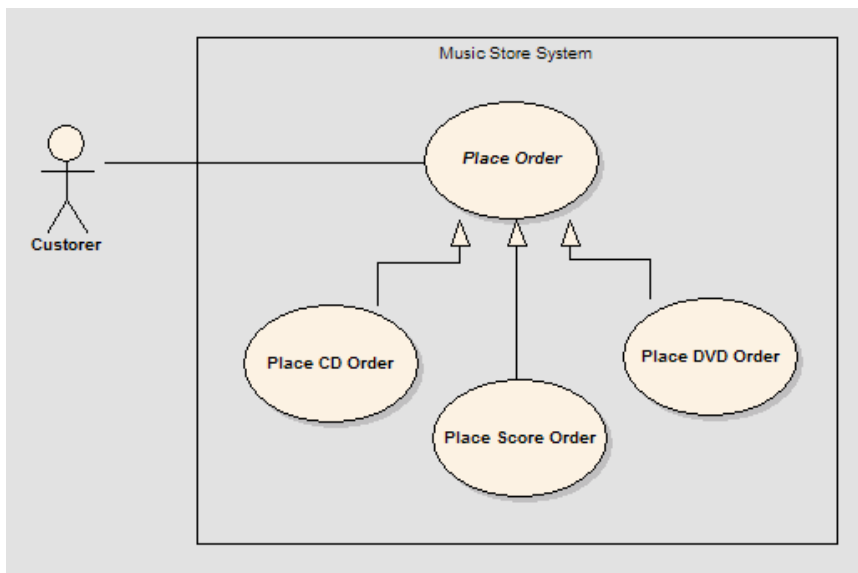


Figura 5 – generalizzazione tra casi d'uso

Il caso d'uso base in questo caso non contiene alcuna sequenza di azioni poiché è astratto (in UML è modellato formattando il suo nome in *italico*). L'unico scopo è quello di evidenziare che esistono diverse tipologie di ordini che un utente del negozio di articoli musicali può effettuare. Ogni caso d'uso derivato descrive una specifica tipologia (è possibile, ad esempio, acquistare CD, DVD musicali oppure spartiti). Solamente i casi d'uso derivati vengono documentati per mezzo di una descrizione testuale. Questo modo di utilizzare la generalizzazione è coerente sia da un punto di vista formale, sia da quello concettuale. Formalmente, infatti, un caso d'uso derivato "eredita" il comportamento di un caso d'uso base. Nell'esempio di figura 5 il caso d'usa base è vuoto, ed ogni caso d'uso derivato sarà sempre visto come un'estensione del primo. Da un punto di vista concettuale, inoltre, possiamo dire che nessun cliente effettua concretamente un

ordine generico – anzi, possiamo ritenere che l'ordine generico stesso sia un'entità astratta, non reale, analogamente all'attività di preparare un primo piatto. Nessun cuoco prepara concretamente un primo piatto: si preparano pastasciutte, minestre, lasagne, che sono tutte specializzazioni concrete del concetto generale "primo piatto". Utilizzata in questo modo, raramente la generalizzazione porta a confusioni di sorta con l'estensione.

Il ruolo dei diagrammi dei casi d'uso nella modellazione object-oriented

I diagrammi dei casi d'uso descrivono un modello funzionale del sistema. Va notato subito che stiamo parlando di *sistema* e non di *problema* oppure di *dominio applicativo*. La scelta di utilizzare questo termine non è casuale. Molti testi sulle metodologie ad oggetti identificano i diagrammi dei casi d'uso come un artefatto di analisi. L'analisi, tuttavia, modella il dominio e il problema: l'analisi è sicuramente indipendente dalle scelte e dai dettagli implementativi che caratterizzano un sistema. Possiamo concludere quindi che i diagrammi dei casi d'uso sono un artefatto del design e non dell'analisi! Sono anche un artefatto del design object-oriented? Assolutamente no! Il più importante contributo fornito dal metodo ad oggetti nella modellazione di un sistema è di tipo strutturale, per meglio gestire le dipendenze tra le classi e la complessità del software [Meyer, 1997] [Riel, 1996] [Martin, 2000]. Allora che dignità possiamo attribuire alla modellazione basata sui casi d'uso (e, più in generale, alla modellazione funzionale)? La principale utilità consiste nel fornire una documentazione facilmente comprensibile all'utente, ai committenti e alle diverse figure coinvolte nel progetto (manager, esperti di dominio, personale di assistenza clienti, ecc.). Dopotutto, un utente quando interagisce con un sistema, lo fa avendo una chiara percezione di quelle che sono le sue funzionalità, e non di quella che è la sua struttura interna. Un autista può guidare un veicolo ignorando cosa accada nella centralina elettronica oppure all'interno del motore ad iniezione. In questa prospettiva, i diagrammi, ma soprattutto i casi d'uso, diventano un veicolo importante di comunicazione per chiarire funzionalità e specifiche del sistema da costruire. Se ben documentato, un modello dei casi d'uso può diventare un contratto effettivo con il committente. Più in particolare, la descrizione narrativa di un caso d'uso è parte integrante del contratto, mentre il diagramma dei casi d'uso può essere visto come una tabella dei contenuti, un sommario, che organizza i casi d'uso. Il lettore riceve quindi una visione globale dell'organizzazione del documento dei requisiti funzionali guardando i diagrammi, per poi localizzare più agevolmente ed entrare nel dettaglio nella sezione che gli interessa. Alcuni CASE tool permettono inoltre l'inserimento di link ipertestuali sia all'interno dei diagrammi, sia all'interno della descrizione narrativa, simulando una completa navigazione.

Documentare i casi d'uso

In conclusione di articolo proponiamo un template per la documentazione dei casi d'uso. Il template è articolato in due sezioni: una sezione di base che contiene tutti gli elementi essenziali, e una sezione alternativa che identifica elementi opzionali.

TEMPLATE DI BASE

- *Nome caso d'uso* – Ogni caso d'uso deve avere un nome; il nome esprime il goal dell'utente nell'utilizzo del sistema.
- *Goal (summary description)* – descrizione della funzionalità fornita dal sistema e che soddisfa una necessità dell'utente, ossia che è percepita dallo stesso utente come "valore".
- *Attori* – persona, dispositivo o altra entità esterna al sistema che interagisce con il sistema. Per ogni caso d'uso esiste sempre un attore primario che è colui che inizia il caso d'uso stesso.
- *Precondizioni* – Condizioni che devono essere soddisfatte all'inizio del caso d'uso. Rappresentano le "garanzie minime" che devono essere soddisfatte per poter attivare lo scenario di utilizzo del sistema (Garanzie fornite dagli attori al sistema).
- *Trigger* – evento trigger che attiva il caso d'uso
- *Descrizione (main success scenario o scenario principale)* – descrizione della sequenza di interazioni più comune tra gli attori e il sistema. In particolare viene descritta la sequenza principale che porta alla conclusione del caso d'uso con successo. La descrizione è definita in termini di input forniti dall'attore e di risposta del sistema. Il sistema è trattato secondo un modello di tipo black-box, concentrandosi su cosa esso fa in risposta agli input, e non su come internamente queste risposte vengano prodotte.
- *Alternative (estensioni)* – descrizioni delle variazioni dalla sequenza di passi tipica del main success scenario. Tali alternative estendono lo scenario principale. La gestione delle eccezioni è un esempio tipico di tali estensioni. Non tutte le alternative portano necessariamente ad un fallimento del caso d'uso.

- *Postcondizioni* – Condizioni sempre soddisfatte al termine del caso d'uso (Garanzie fornite dal sistema agli attori)

ELEMENTI OPZIONALI DEL TEMPLATE

- *Priorità* – priorità del caso d'uso
- *Response time* – vincoli relativi al tempo di risposta che il sistema deve garantire nell'esecuzione del caso d'uso.
- *Scope* – identifica l'ambito di visibilità dello use case. I due ambiti tipici sono il sistema che si sta progettando (system use case model) oppure l'azienda (business use case model). Limitatamente al system use case model, se i casi d'uso sono organizzati in sottosistemi funzionali (utilizzando dei package), in questa sezione si riporta il sottosistema funzionale in cui lo use case è contenuto.
- *Dipendenze con altri casi d'uso* – riferimenti ad altri casi d'uso collegati (relazioni di inclusione o di estensione). Se non ci sono dipendenze, la voce può essere omessa dalla documentazione del caso d'uso.
- *Questioni aperte* – dubbi, aspetti non chiari o ancora da concordare relativamente ad uno o più comportamenti descritti dallo use case.

Di seguito descriviamo un esempio di documentazione di un caso d'uso ispirato dalla modellazione di un sistema ATM [Gomaa, 2000]. In figura 6 sono illustrate le funzionalità di base di un sistema ATM:

- Prelevare fondi (*Withdraw Funds*);
- Visualizzare ultimi movimenti (*Query Account*);
- Trasferire fondi (*Transfer Funds*).

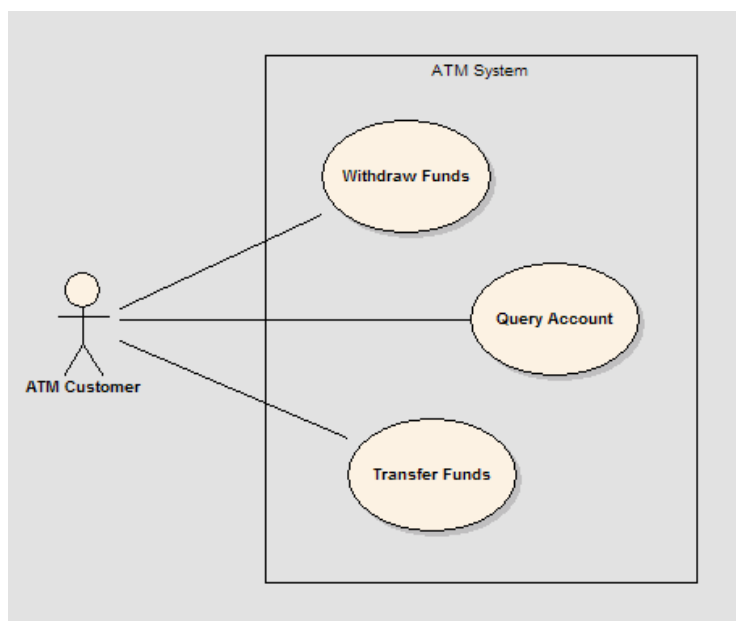


Figura 6 – Casi d'uso che rappresentano le funzionalità di base di un terminale ATM

Riproponiamo a titolo esemplificativo la documentazione del caso d'uso Prelevare Fondi (*Withdraw Funds*).

Nome caso d'uso: Withdraw Funds.

Scope: ATM system

Goal (summary): L'utente dell'ATM preleva una specifica somma di denaro da un conto corrente bancario valido.

Dipendenze con altri casi d'uso: nessuna

Attori: Cliente del terminale ATM.

Trigger: inserimento di una carta di credito nel lettore del terminale ATM

Precondizioni: Il terminale dell'ATM è in attesa (idle) e visualizza un messaggio di "Benvenuto".

Descrizione (main success scenario o scenario principale):

1. Il cliente inserisce nel lettore del terminale ATM la sua carta di credito.
2. Il sistema riconosce la carta e ne legge il numero (card number).
3. Il sistema visualizza a terminale la richiesta di inserimento del PIN utente.
4. Il cliente inserisce il suo PIN.
5. Il sistema verifica che la carta non è scaduta, né è stata rubata o risulta smarrita.
6. Il sistema verifica che il PIN inserito dal cliente corrisponda con quello della carta.
7. Il sistema controlla che il conto corrente sia accessibile mediante l'utilizzo della carta.
8. Il sistema visualizza il conto corrente del cliente e visualizza le possibili operazioni che il cliente può effettuare (Prelievo, Ultimi Movimenti, Saldo Disponibile, Altri Servizi).
9. Il cliente seleziona il conto corrente desiderato (nel caso il cliente ne abbia più di uno) e seleziona l'operazione Prelievo.
10. Il sistema visualizza la richiesta dell'ammontare da prelevare.
11. Il cliente inserisce l'ammontare da prelevare.
12. Il sistema verifica che il conto corrente contenga sufficienti fondi per consentire la conclusione dell'operazione e che non sia stato superato il limite massimo giornaliero per il prelievo.
13. Il sistema autorizza il prelievo.
14. Il sistema emette il denaro, registrando la transazione sul conto corrente.
15. Il sistema stampa la ricevuta mostrando il numero della transazione, il tipo di operazione effettuata, la quantità di denaro prelevata e il saldo del conto corrente.
16. Il sistema espelle la carta.
17. Il sistema ritorna nello stato iniziale di attesa (idle), visualizzando il messaggio di "Benvenuto"

Alternative (estensioni):

- 2a. Se il sistema non riconosce la carta dell'utente, quest'ultima viene espulsa dal lettore.

- 5a. Se la carta risulta scaduta, il sistema la confisca.
- 5b. Se la carta risulta smarrita, il sistema la confisca.
- 5c. Se la carta risulta rubata, il sistema la confisca.

- 6a. Se il cliente ha inserito un PIN che non corrisponde con quello della carta, il sistema visualizza una nuova richiesta di inserimento del PIN. Se il cliente inserisce per tre volte un PIN errato, il sistema confisca la carta.

- 7a. Se il sistema verifica che il numero di conto non è valido, viene visualizzato un messaggio di errore e la carta viene espulsa dal lettore.

- 12a. Se non ci sono sufficienti fondi nel conto corrente, il sistema visualizza un messaggio di errore ed espelle la carta. Il terminale ATM viene riportato allo stato di attesa (idle) e viene visualizzato il messaggio di "Benvenuto".
- 12b. Se il limite giornaliero massimo di prelievo è stato già raggiunto, il sistema visualizza un messaggio di errore ed espelle la carta. Il terminale ATM viene riportato allo stato di attesa (idle) e viene visualizzato il messaggio di "Benvenuto".

- 14a. Se il terminale ATM non ha sufficienti fondi per completare la transazione, il sistema visualizza un messaggio di errore, espelle la carta, il terminale ATM viene riportato allo stato di attesa (idle) e viene visualizzato il messaggio di "Benvenuto".
- 14b. Se il cliente seleziona da terminale il pulsante Cancel, il sistema cancella la transazione ed espelle la carta. Il terminale ATM viene riportato allo stato di attesa (idle) e viene visualizzato il messaggio di "Benvenuto".

Postcondizioni: L'utente ha ottenuto la somma di denaro che aveva richiesto di prelevare.

Questioni aperte: nessuna.

Le sezioni Dipendenze con altri casi d'uso e Questioni aperte potrebbero anche essere omesse dalla documentazione del caso d'uso, in quanto vuote, e aggiunte eventualmente in un secondo momento, all'occorrenza. Seguono le documentazioni degli altri casi d'uso nel diagramma UCD1.

Conclusioni

In questo articolo abbiamo esaminato che cosa sono i casi d'uso, quali sono le relazioni standard previste da UML e quale può essere il ruolo di un modello funzionale nell'ambito dello sviluppo di un progetto software.

Bibliografia

- [Fowler&Kendall, 1999] Fowler, Martin; Kendall, Scott – “*UML Distilled 2/E*”, Addison Wesley, 1999
- [Gomaa, 2000] Gomaa, Hassan – “*Designing Concurrent, Distributed, and Real-Time Applications with UML*”, Addison Wesley, 2000
- [Jacobson, 1995] Jacobson, Ivar – “*Object-Oriented Software Engineering: A Use Case Driven Approach*”, ACM Press, 1992
- [Martin, 2000] Martin, Robert C. – “*Design Principles and Design Patterns*”, www.objectmentor.com
- [Meyer, 1997] Meyer, Bertrand – “*Object-Oriented Software Construction 2/E*”, Prentice Hall, 1997
- [Pender, 2003] Pender, Tom – “*UML Bible*”, Wiley & Sons, 2003
- [Riel, 1996] Riel, Arthur J. – “*Object-Oriented Design Heuristics*”, Addison Wesley, 1996
- [Rumbaugh et al, 1999] Rumbaugh, James; Booch, Grady; Jacobson, Ivar – “*The Unified Modeling Language Reference Manual*”, Addison Wesley, 1999