

Capitolo 9

Diagrammi di interazione

*Una interazione ben strutturata è come un algoritmo ben strutturato:
efficiente, adattabile e comprensibile*

GRADY BOOCH

Introduzione

Le parti che costituiscono un qualsivoglia sistema non sono assemblate per persistere semplicemente in una posizione stabile, bensì interagiscono tra loro scambiandosi messaggi al fine di raggiungere un preciso scopo. In ogni sistema ciascun componente svolge un determinato compito e interagisce con altri al fine di produrre un risultato globale, una funzionalità che va ben oltre la semplice somma dei risultati prodotti dai singoli oggetti. Quantunque ciò possa sembrare piuttosto scontato, negli ambienti di lavoro si assiste a una tendenza, verosimilmente non molto corretta, nel conferire particolare enfasi ai modelli statici. Se da un lato è abbastanza naturale e conveniente procedere con una iniziale produzione dei diagrammi di struttura statica (che fondamentalmente utilizzano il formalismo dei diagrammi delle classi), dall'altro la progettazione non si deve esaurire con questa proiezione. Lo studio del comportamento del sistema — dinamico per definizione — dovrebbe occupare un ruolo centrale. Ciò è particolarmente vero per sistemi basati sul paradigma OO, ove gran parte dell'attività di disegno è riconducibile all'individuazione, alla separazione e all'incapsulamento del comportamento da realizzare. I diagrammi dinamici, sebbene si configurino come una verifica accurata e formale di quanto modellato nella proiezione statica, a loro volta, da soli non sono sufficienti: non sono idonei a modellare informazioni di carattere strutturale e di gestione delle dipendenze. Le due proiezioni

devono quindi completarsi a vicenda; la descrizione di ogni comportamento deve essere modellata sia attraverso una prospettiva che mostri la struttura statica degli elementi cooperanti, sia per mezzo di un'altra che illustri il relativo comportamento dinamico. Nei capitoli precedenti si è visto come la struttura statica di un sistema, o di sue parti, sia esprimibile in UML attraverso il formalismo dei diagrammi delle classi e quello dei diagrammi degli oggetti — anche i diagrammi dei componenti e di dispiegamento permettono di evidenziare la struttura statica del sistema, ma l'attenzione è focalizzata più su aspetti di carattere fisico — mentre, per ciò che concerne la descrizione del comportamento dinamico, esistono diversi strumenti. Obiettivo del presente capitolo è avviare l'illustrazione dei formalismi forniti dallo UML per modellare comportamenti dinamici. In particolare, in questa sede l'attenzione è focalizzata sui diagrammi di interazione (*interaction diagrams*). Con questo nome si indicano i diagrammi di sequenza (*sequence diagrams*) e quelli di collaborazione (*collaboration diagrams*). La trattazione degli strumenti forniti dallo UML destinati alla descrizione del comportamento dinamico del sistema verrà conclusa nel Capitolo 10 con l'illustrazione dei diagrammi di attività (*activity diagrams*) e quelli di stato (*statechart diagrams*).

Lo UML fornisce pertanto ben quattro diverse tipologie di diagrammi per modellare comportamenti dinamici. Ciascuno di essi conferisce particolare importanza a un determinato aspetto e quindi risulta particolarmente idoneo in determinate situazioni e meno in altre. Per esempio, i diagrammi degli stati sono molto utili per mostrare l'evoluzione di un oggetto (anche composto) attraverso un insieme ben definito di stati, mentre è del tutto inefficace per descrivere la sequenza di messaggi scambiati da un gruppo di oggetti che prendono parte ad una specifica interazione. Uno dei compiti del modellatore consiste proprio nel saper individuare il formalismo che, di volta in volta, permette di evidenziare l'aspetto al quale si vuole conferire maggiore importanza, tenendo presente anche il pubblico dei fruitori.

I diagrammi di sequenza e di collaborazione sono forme diverse di diagrammi di interazione. Quantunque permettano di mostrare lo stesso contenuto informativo — è possibile passare automaticamente da una forma all'altra — si distinguono per l'aspetto dell'interazione cui è attribuita maggiore importanza. I diagrammi di sequenza enfatizzano lo scambio dei messaggi nel tempo di un insieme di oggetti cooperanti, mentre i diagrammi di collaborazione attribuiscono maggiore importanza all'organizzazione degli oggetti coinvolti nell'interazione modellata. Nei diagrammi di sequenza è presente un'apposita dimensione dedicata al fattore tempo, mentre l'organizzazione degli elementi che prendono parte all'interazione (collaborazione) è implicita: se un oggetto invia un messaggio a un altro, si evince che i due sono interconnessi per mezzo di una qualche forma di relazione. Nei diagrammi di collaborazione invece non è prevista un'apposita dimensione dedicata al fattore tempo, sebbene l'ordine dei messaggi sia esplicitamente dichiarato per mezzo di apposita numerazione degli stessi, mentre la collaborazione è resa evidente dalla presenza di espliciti legami tra gli oggetti.

Nonostante sia pratica comune affermare che gli elementi mostrati nei diagrammi di interazione sono oggetti, ciò non è del tutto preciso. Come si vedrà di seguito, spesso non è mostrato il comportamento di specifici oggetti bensì quello di istanze che esercitano ruoli ben definiti in una determinata iterazione.

Con il termine *interazione* si indica una descrizione comportamentale che include la definizione di una precisa sequenza di messaggi scambiati da un determinato insieme di oggetti al fine di produrre uno specifico risultato. Pertanto, un'interazione è costituita da una collaborazione (complesso degli oggetti cooperanti) con, in aggiunta, un insieme ordinato di flussi di messaggi resi possibili dai legami tra gli oggetti partecipanti.

In OO, quando degli oggetti collaborano si dice che si scambiano messaggi. Le collaborazioni (più precisamente interazioni) hanno luogo al fine di raggiungere uno specifico scopo, come per esempio l'erogazione di un determinato servizio agli utenti. Nella maggior parte dei casi, i messaggi consistono in semplici invocazioni (sincrone) di metodi; altre volte possono consistere in invio di segnali e veri e propri scambi asincroni di messaggi (pubblicazione di messaggi su un apposito sistema di messaggistica).

Elementi comuni

Prima di procedere con l'illustrazione dei diagrammi di sequenza e di collaborazione, si è ritenuto opportuno introdurre i concetti utilizzati da entrambe le notazioni. Chiaramente, trattandosi di due diverse istanze di una stessa tipologia di diagramma (quello di interazione), è lecito attendersi molte aree in comune.

Ruolo classificatore

Chiunque abbia avuto modo di utilizzare o leggere almeno un diagramma di interazione, indipendentemente dalla versione, si attende, verosimilmente, di imbattersi in notazioni aventi come elementi base oggetti (nell'accezione propria OO) opportunamente collegati con evidenziate specifiche comunicazioni. In quest'ottica, un diagramma di sequenza illustra un opportuno scambio di dati tra oggetti attuato al fine di conseguire un determinato risultato e si realizza conferendo particolare enfasi alla dimensione temporale. Un diagramma di collaborazione mostra invece la struttura di oggetti eventualmente corredata dalla rappresentazione di un comportamento dinamico, in cui si attribuisce particolare importanza all'organizzazione degli oggetti collaboranti.

Sebbene in questa convinzione non ci sia nulla di sbagliato, diagrammi di interazione così concepiti sarebbero troppo limitati. Nella pratica può rivelarsi utile realizzare diagrammi di interazione che coinvolgano altri elementi e non solo istanze. Per risolvere questo problema, il metamodello UML è stato arricchito introducendo nuovi elementi e prevedendo due versioni per ciascuno dei due diagrammi di interazione: una a livello di istanze — quella cui si è portati a pensare — e l'altra a un livello più astratto, di specifica.

Quest'ultima è meno intuitiva e rappresenta la formalizzazione di un artificio di modellazione resosi necessario per poter realizzare diagrammi di interazione con diversi elementi.



In questa seconda versione un elemento molto importante è il “ruolo classificatore” (*ClassifierRole*) che definisce una proiezione ristretta del *Classificatore* cui è associato (precisamente *ClassifierRole* eredita ed è associato all'elemento *Classificatore*), determinata dal comportamento richiesto dalla particolare collaborazione. In ogni collaborazione raramente è richiesto a tutti i classificatori inclusi di esercitare tutto il relativo comportamento; nella stragrande maggioranza dei casi è necessaria solo una sua parte: un ruolo appunto. Un medesimo classificatore può svolgere diversi ruoli così come uno stesso *ClassifierRole* può essere associato a svariati classificatori (classica relazione *n a n*). Pertanto, in un diagramma di interazione è possibile inserire ruoli di *Classificatori* (si ricordi che *Classificatore* è l'elemento “genitore” di molti elementi dello UML quali *Classe*, *Interfaccia*, *Nodo*, *Componente*, *Attore*, *Caso d'Uso*, ecc.), eccezion fatta per il *ClassifierRole* stesso, a condizione che il classificatore sia in grado di emettere e ricevere messaggi, ovviamente.

Logica conseguenza della presenza di due versioni di elementi base è la necessità di realizzare una doppia versione di tutti gli altri elementi. Si considerino per esempio le relazioni (fig. 9.1). Qualora il diagramma sia al livello di istanze, le connessioni (*Link*) tra istanze (*Instance*) servono per scambiare una sequenza parzialmente ordinata di stimoli (*Stimulus*) mentre, se è al livello di specifica, le connessioni (*AssociationRole*) tra ruoli classificatori (*ClassifierRole*) trasportano messaggi (*Message*).

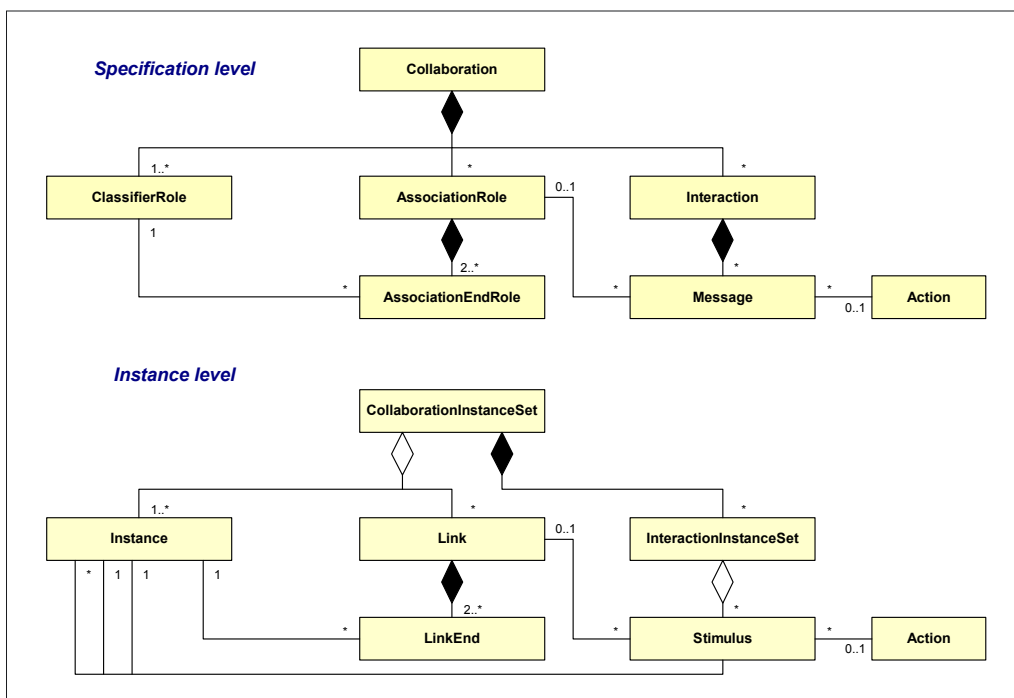
Ancora, nel caso di livello di istanze, l'interazione è definita *Interazione di insieme di istanze* (*InteractionInstanceSet*) invece che semplicemente *Interazione* (*Interaction*) come nel caso della versione di specifica, mentre la *Collaborazione* (*Collaboration*) diviene *Collaborazione di insieme di istanze* (*CollaborationInstanceSet*).

Gli elementi riportati tra parentesi sono le metaclassi del metamodello così come riportato nella fig. 9.1.

Stimoli e messaggi

Uno stimolo è una comunicazione tra due istanze che trasporta informazioni con l'aspettativa che precise azioni abbiano luogo come conseguenza della sua ricezione. Queste azioni possono consistere in un'invocazione di un'operazione, nella creazione o distruzione di un'istanza, nella generazione di un segnale ecc. Un messaggio è la specificazione di uno stimolo: precisa i ruoli cui il mittente e il destinatario si devono conformare e l'azione che, se eseguita, produce lo stimolo che si conforma al messaggio.




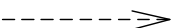
Figura 9.1 — *Panoramica degli elementi del metamodello coinvolti nell'utilizzo dei diagrammi di interazione (diagramma tratto dalle specifiche OMG UML 1.4 figura 3-58). La maggior parte di questi elementi appartiene al package Collaboration sottopackage di Behavioral Elements. Nel diagramma delle specifiche, ogni elemento è associato al suo rispettivo; per esempio, tra l'elemento Instance e ClassifierRole esiste una relazione molti a molti: un'istanza può recitare il ruolo di diversi ClassifierRole, così come un ClassifierRole può prevedere diverse istanze che si conformano ad esso.*



In UML stimoli e messaggi sono rappresentati graficamente per mezzo di frecce con forme diverse in funzione della relativa semantica come illustrato in tab. 1.

Le comunicazioni sono ornate da un'etichetta con significato che varia a seconda della particolare versione di diagramma: in quelli di sequenza rappresenta il nome dell'operazione da invocare o il nome del segnale che l'ha originata, mentre nei diagrammi di collaborazione indica il messaggio che viene inviato, gli eventuali argomenti e valore di ritorno. Spesso l'etichetta è fornita del numero di sequenza per mostrare l'ordine del messaggio nell'interazione di appartenenza, considerando invocazioni annidate, iterazioni, divisioni del flusso, concorrenze e sincronizzazioni.

Tabella 1 — Rappresentazioni grafiche dei diversi tipi di comunicazione.

Simbolo	Descrizione
Freccia piena con punta piena. 	Rappresenta un'invocazione di procedura o un flusso di controllo annidato. L'intero flusso annidato deve essere completamente concluso prima che la sequenza appartenente al livello esterno possa riprendere. Pertanto questa freccia può essere utilizzata per evidenziare ordinarie invocazioni di procedure, così come per denotare situazioni in cui un insieme di istanze sono attive concorrentemente ed una di esse invia un segnale ed attende che la corrispondente sequenza di comportamento annidato sia completata prima di poter proseguire (comunicazione sincrona).
Freccia piena con punta stilizzata. 	Rappresenta una comunicazione asincrona e quindi priva dell'annidamento del controllo. Il mittente spedisce lo stimolo e quindi prosegue immediatamente con il passo successivo previsto dalla relativa esecuzione.
Mezza freccia stilizzata 	Eliminata ed inglobata nella precedente a partire dalla versione 1.4 dello UML in quanto la distinzione delle due versioni era molto sottile e spesso fonte di confusione. Rappresentava una comunicazione asincrona il cui obiettivo era di mostrare esplicitamente lo scambio di un messaggio (asincrono) tra due oggetti in una sequenza procedurale.
Freccia tratteggiata con punta stilizzata. 	Indica il ritorno dall'invocazione di una procedura.



Nei diagrammi di sequenza, la numerazione dei messaggi può essere placidamente omessa: l'ordine è dato dalla posizione fisica delle frecce nel diagramma. Ciononostante è comunque utile per evidenziare iterazioni, flussi di controllo concorrenti, alternativi, ecc. Nei diagrammi di collaborazione tale numerazione è invece assolutamente indispensabile: rappresenta l'unico strumento per evidenziare la sequenza di messaggi in funzione del fattore tempo. Qualora si decida di visualizzare esplicitamente la numerazione degli stimoli anche nei diagrammi di sequenza, è necessario rispettare le regole grammaticali definite per le etichette (confrontare paragrafo *Etichette dei messaggi e stimoli*).

Diagrammi di sequenza

Definizione



I diagrammi di sequenza illustrano interazioni tra specifici “oggetti” conferendo particolare enfasi alla dimensione temporale. Con il termine interazione si indica un comportamento dinamico che include un insieme (parzialmente) ordinato di messaggi scambiati tra gli “oggetti” cooperanti (che prendono parte alla colla-

borazione) al fine di raggiungere un determinato risultato, come per esempio la fornitura di uno specifico servizio all'utente.

Dalla seconda parte della definizione è evidente che un'interazione è definita nel contesto di una collaborazione. Pertanto si specifica l'insieme degli elementi "collaboranti" considerati, magari implicitamente, e quindi la sequenza, parzialmente ordinata, dei messaggi che si scambiano per raggiungere il risultato voluto.

Da notare che, anche qualora si tracci un diagramma a livello di istanza, gli elementi mostrati non sono propriamente oggetti ma ciascuno di essi ne rappresenta una pluralità. A voler essere precisi, quindi, si tratta di ruoli che possono essere recitati da oggetti.

Gli elementi collaboranti sono generalmente indicati come oggetti. Ciò, sebbene semplifichi la trattazione — per questo motivo anche nel presente testo si ricorre sovente a tale semplificazione — non sempre è corretto. Nel metamodello UML la definizione di diagramma di sequenza prevede due "versioni":

1. quella più comune a cui si è abituati a pensare, detta *InteractionInstanceSet* (interazione di insieme di istanze) che coinvolge *Stimoli* e *Istanze* di *Classificatori* (nel metamodello, l'elemento *Instance* è associato ad almeno un elemento *Classificatore* che ne definisce le proprietà strutturali e comportamentali);
2. una meno intuitiva, definita in termini di *Interazione*, ossia scambi di *Messaggi* tra particolari entità modellate attraverso un apposito elemento denominato *ClassifierRole* (ruolo classificatore).

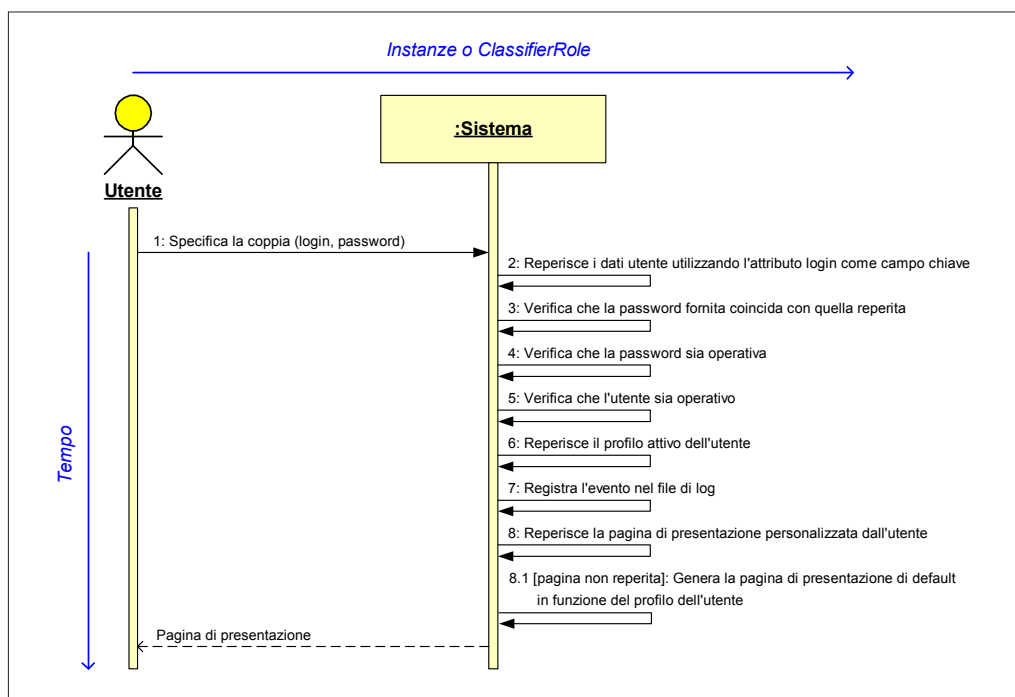
La seconda definizione è un artificio tecnico reso necessario per poter specificare nei diagrammi di sequenza elementi che non siano esclusivamente oggetti.

Ricapitolando, un diagramma di sequenza può mostrare:

1. una collezione di **oggetti** e **frecce**, indicanti rispettivamente *Istanze* e *Stimoli*;
2. una collezione di **ruoli di classificatori** e di **frecce** rappresentanti rispettivamente *ClassifierRole* e *Messaggi* (nel metamodello UML si parla di *Collaboration* costituita da *Interaction*).

Gran parte del formalismo dei diagrammi di sequenza si deve alla notazione *Object Message Sequence Chart* (grafico della sequenza di messaggi tra oggetti) attribuibile a Buschmann, Rohert, Sommerlad e Stal la quale, a sua volta, deriva dall'antenata *Message Sequence Chart* (grafico della sequenza di messaggi).

Figura 9.2 — Rappresentazione grafica del main scenario di un servizio di autenticazione utente. Più precisamente, si dovrebbe trattare di uno scenario alternativo, infatti, sebbene permetta di raggiungere l'obiettivo del servizio, ciò avviene utilizzando un percorso alternativo: l'utente non dispone della pagina di presentazione personalizzata.



Formalismo

La rappresentazione grafica dei diagrammi di sequenza prevede due dimensioni (fig. 9.2):

- l'asse verticale che rappresenta il fattore tempo. Per default aumenta verso il basso (nulla vieterebbe di utilizzare una notazione speculare, anche se ciò potrebbe generare qualche complicazione);
- l'asse orizzontale destinato a ospitare le varie Istanze o ClassifierRole.

Dalla definizione dello UML, anche l'assegnazione canonica degli assi può essere invertita, sebbene siano poi ben pochi i tool che lo consentono.

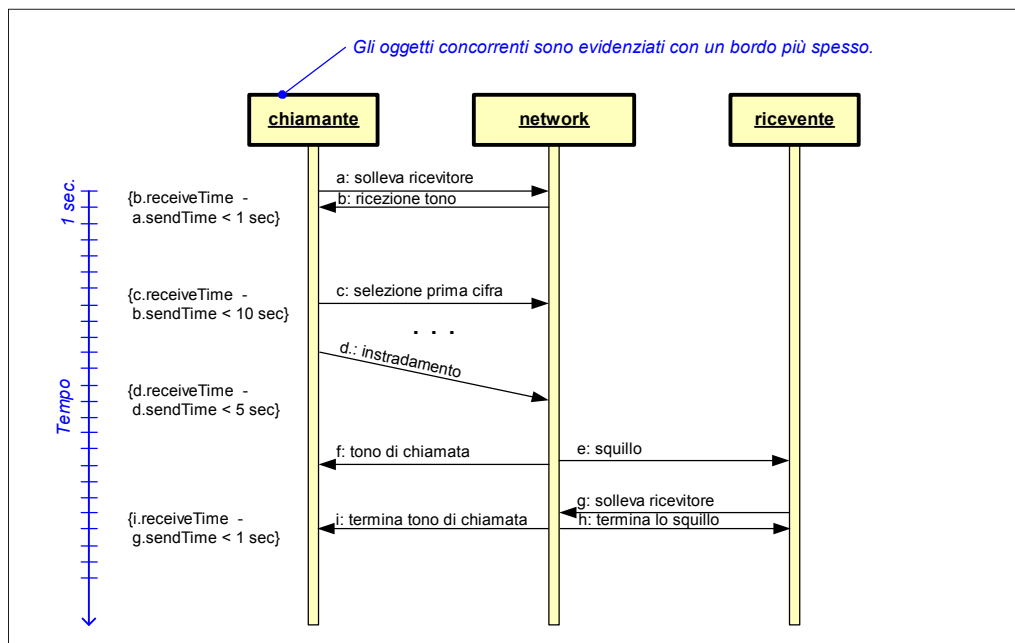
Gli elementi che prendono parte alla collaborazione sono visualizzati con il classico rettangolo proprio dei Classifieri. Nel caso si tratti di ClassifierRole si uti-

lizza il simbolo delle classi, mentre nel caso di istanze si ricorre alla variante utilizzata per gli oggetti.

Nella stragrande maggioranza dei casi, l'asse temporale non prevede alcuna gradazione, la distanza tra i vari messaggi presenti dipende esclusivamente da considerazioni relative all'aumento di leggibilità e linearità del diagramma stesso. In specifici contesti, tuttavia — come per esempio la modellazione di sistemi real-time — può risultare molto utile disporre di un asse graduato al fine di evidenziare il tempo richiesto da ciascun servizio e il dettaglio del modo in cui tale valore sia ottenuto (fig. 9.3).

Per quanto riguarda l'asse delle "istanze", l'ordine con cui sono presentati i vari elementi non ha alcun significato particolare, sebbene (come riportato nella sezione dedicata allo stile), si cerchi di inserirli rispettando l'organizzazione del sistema in strati. Ciò, tra l'altro, permette di razionalizzare la rappresentazione dello scambio di messaggi: è abba-

Figura 9.3 — *Semplice diagramma di sequenza con oggetti concorrenti (tratto dalle specifiche OMG UML 1.4). Dall'analisi del diagramma emerge che i tre oggetti sono sempre "attivi" anche se non utilizzati: tutta la linea di vita (line of life) è coperta dalla barra verticale che indica la localizzazione del controllo (flow of control). Ciò è dovuto al comportamento degli oggetti che prevede che siano in grado di rilevare — in qualsiasi momento! — un eventuale "messaggio" di loro competenza.*



stanza logico attendersi che istanze appartenenti a strati contigui diano luogo a un sostenuto scambio di messaggi, che invece è nullo tra istanze appartenenti a strati non adiacenti. L'autore è pur consapevole che si trova sempre il "genio" che incorpora in elementi della GUI accessi al database...

Nei diagrammi di sequenza le istanze partecipanti alla collaborazione sono mostrate attraverso il rettangolo (con notazione leggermente variante in funzione al tipo), con associato un segmento tratteggiato uscente dal rettangolo e normale al suo lato inferiore. Que-

Figura 9.4 — Diagramma di sequenza con la versione standard del pattern Service Locator. Il diagramma rappresenta una versione generale della strategia per l'architettura J2EE. Per esempio il ServiceFactory rappresenta l'EJBHome nel caso di componenti EJB, mentre nel caso di componenti JMS si può trattare di TopicConnectionFactory (publish/subscribe) o QueueConnectionFactory (point-to-point). Molto in breve (consultare il sito www.javasoft.com per maggiori informazioni), ogniqualvolta un oggetto client (per esempio un handler di una pagina JSP) deve eseguire un metodo business per espletare un servizio richiesto dall'utente (come per esempio autenticazione, visualizzazione estratto conto, ecc.), per prima cosa deve ottenere un riferimento al componente che eroga il servizio (in verità riceve il riferimento a un oggetto proxy, lo stesso che si occupa di rendere trasparenti le comunicazioni). Le direttive Sun prevedono di realizzare un apposito pattern: ServiceLocator, il quale, dato l'identificativo dell'oggetto, reperisca la definizione del ServiceFactory in un apposito repository (JNDI). Ciò permette di ottenere un'istanza del ServiceFactory da cui poi creare il proxy dell'oggetto che eroga il servizio (BusinessService) al quale la classe client può richiedere i vari servizi.

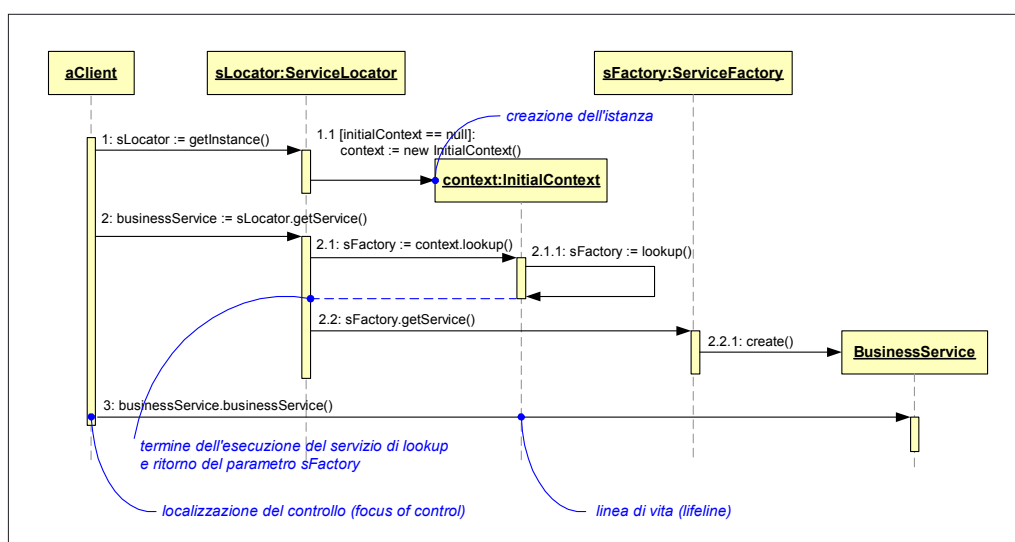
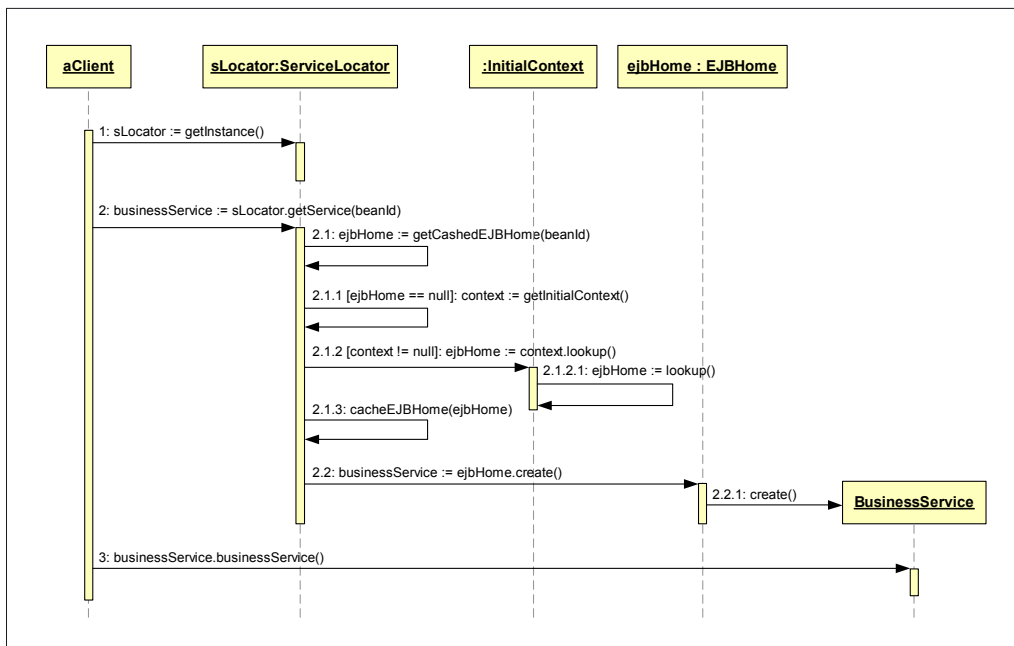


Figura 9.5 — Versione “avanzata” del pattern Service Locator. In questo caso si è limitata l’attenzione ai componenti EJB (si nota dal fatto che il “segnaposto” `ServiceFactory` è stato sostituito da `EJBHome`). Questa variante si differenzia dalla precedente in quanto l’oggetto `ServiceLocator` è dotato di un meccanismo di caching delle interfacce `EJBHome`. Tale meccanismo permette di evitare operazioni di lookup qualora l’interfaccia richiesta sia già presente nella cache. Nel diagramma è mostrato l’oggetto `InitialContext` che effettua il lookup (stimolo 2.1.2.1) al fine di reperire l’interfaccia `EJBHome` (più precisamente l’oggetto che si conforma a tale interfaccia). Tipicamente, quello che avviene nella realtà è che il container esegue una ricerca in un repository (quasi sempre di tipo LDAP) e, una volta reperito l’EJB richiesto, esegue un’operazione di deserializzazione. Pertanto nella realtà l’oggetto viene reperito e quindi creato. Tale comportamento però è specifico dei contenitori e non è assolutamente necessario riportarlo nel diagramma.



sto segmento, indicato con il nome di linea di vita (*lifeline*, fig. 9.4), rappresenta l’esistenza della relativa istanza nel particolare intervallo di tempo mostrato. Dato il canonico orientamento degli assi del diagramma di sequenza, la linea di vita è disegnata verticalmente.

Se una particolare istanza è creata e/o distrutta durante il periodo di tempo mostrato nel diagramma, allora la relativa linea di vita inizia e/o finisce negli appropriati punti (si consideri l’oggetto `DBConnManager` di fig. 9.7), altrimenti è tracciata dall’inizio alla fine

Figura 9.6 — Diagramma delle classi del pattern ServiceLocator.

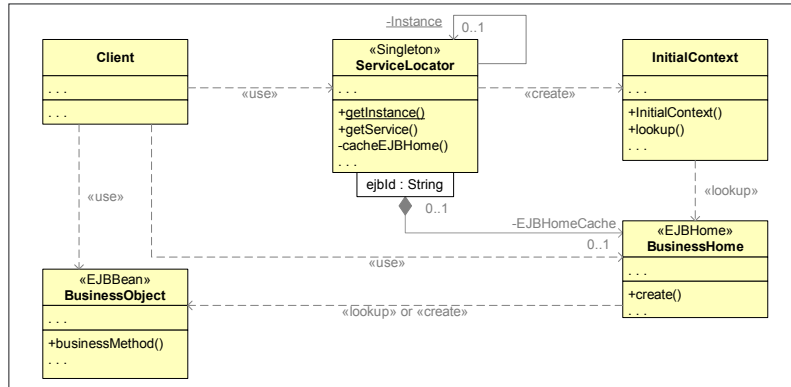
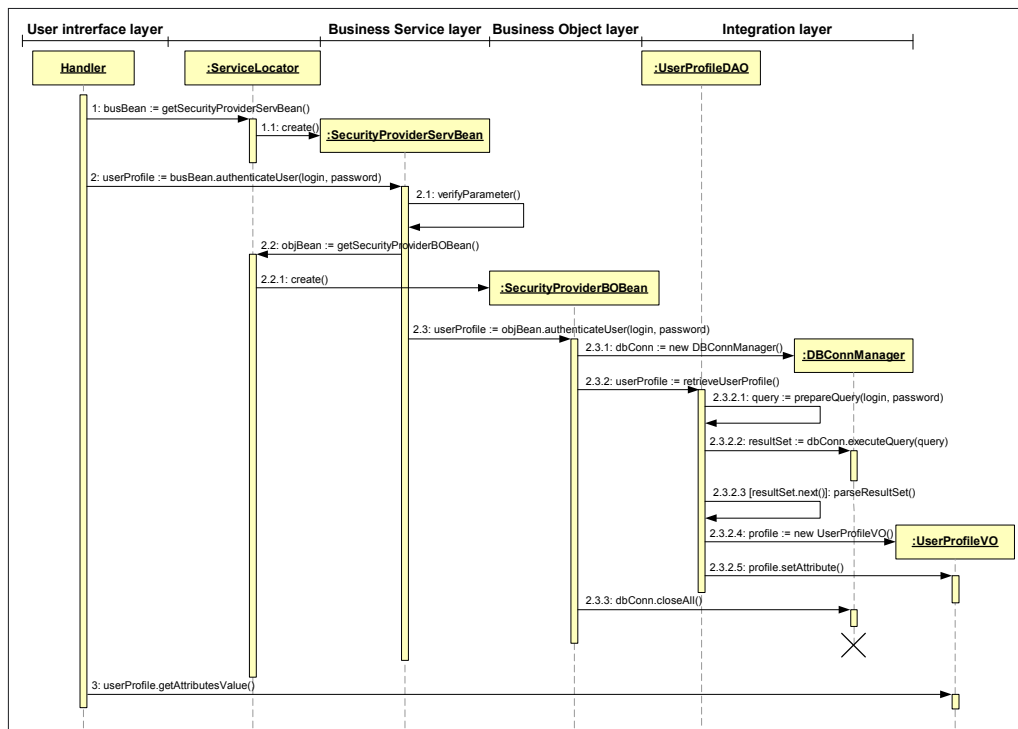


Figura 9.7 — Versione semplificata del comportamento dinamico del servizio di autenticazione utente in architettura J2EE. Il disegno prevede una struttura a strati (cfr Capitolo 8, Dipendenza del modello di analisi dall'architettura), l'uso di Stateless Session EJB, le strategie del Service Locator e del value object utilizzato per trasportare i dati attraverso i vari strati dell'architettura. Le classi ServiceLocator e UserProfileVO non appartengono a un singolo strato.



del diagramma. La creazione di un'istanza nel diagramma è mostrata attraverso una freccia (evento della relativa creazione) la cui punta giunge direttamente al simbolo dell'oggetto. Nelle figg. 9.4 e 9.5, rispettivamente, gli oggetti `ServiceFactory` e `EJBHome` sono creati durante l'esecuzione del servizio, pertanto i relativi simboli non sono mostrati sulla prima linea, bensì nel punto in cui avviene la loro creazione.

La distruzione è mostrata nei diagrammi di sequenza attraverso un simbolo a forma di "X" posto alla fine della linea di vita. La distruzione può essere conseguenza della ricezione di uno specifico messaggio, oppure può essere il risultato di una "autodistruzione" generata alla fine della freccia di ritorno.



Nella realizzazione di diagrammi di interazione appartenenti al modello di disegno relativi al comportamento dinamico di sistemi che sfruttano linguaggi di programmazione come Java, bisogna porre particolare attenzione alla visualizzazione esplicita della distruzione di un oggetto. Tale cautela si estende a tutti quei casi in cui si decida di adottare linguaggi di programmazione che dispongono di meccanismi automatici di gestione della memoria. In questi ambiti, la distruzione degli oggetti non è sotto il controllo del programma bensì è di pertinenza di opportuni processi in esecuzione "concorrente" sulla macchina virtuale. Logica conseguenza è che non potendo definire con precisione quando la distruzione avvenga, non dovrebbe essere mai mostrata esplicitamente. In questi linguaggi il programma si limita a rimuovere i riferimenti agli oggetti — impostare i relativi puntatori al valore *null* — candidandoli quindi alla distruzione. Questa ha luogo quando il processo di pulizia della memoria — il Garbage Collector in Java — ottiene l'assegnazione di cicli di CPU. Ciò nonostante, nella pratica, molto spesso la visualizzazione esplicita della distruzione di un oggetto è utile, anche se poi si tratta di una distruzione logica e non fisica. Chiaramente tale visualizzazione va seriamente ponderata qualora si utilizzi il formalismo dei diagrammi di interazione per illustrare particolari meccanismi, come la gestione di pool di oggetti (per esempio la creazione/distruzione di EJB da parte del container), algoritmi della gestione della memoria, quando sia presente del comportamento associato alla distruzione dell'oggetto, ecc.

Un'istanza in vita prima dell'avvio del diagramma è mostrata riportando il relativo simbolo nella linea delle istanze in modo che preceda la prima freccia (avvio della transazione). Con riferimento a diagramma di fig. 9.5, le istanze delle classi `aClient`, `ServiceLocator` e `InitialContext`, esistono prima dell'esecuzione del servizio e quindi i relativi simboli sono riportati nella prima riga orizzontale. Un'istanza che resta in

vita alla conclusione del diagramma è mostrata attraverso una linea di vita che continua dopo l'ultima freccia. Sempre nel diagramma di fig. 9.5 si può notare che tutte le istanze restano in vita dopo l'esecuzione del servizio.

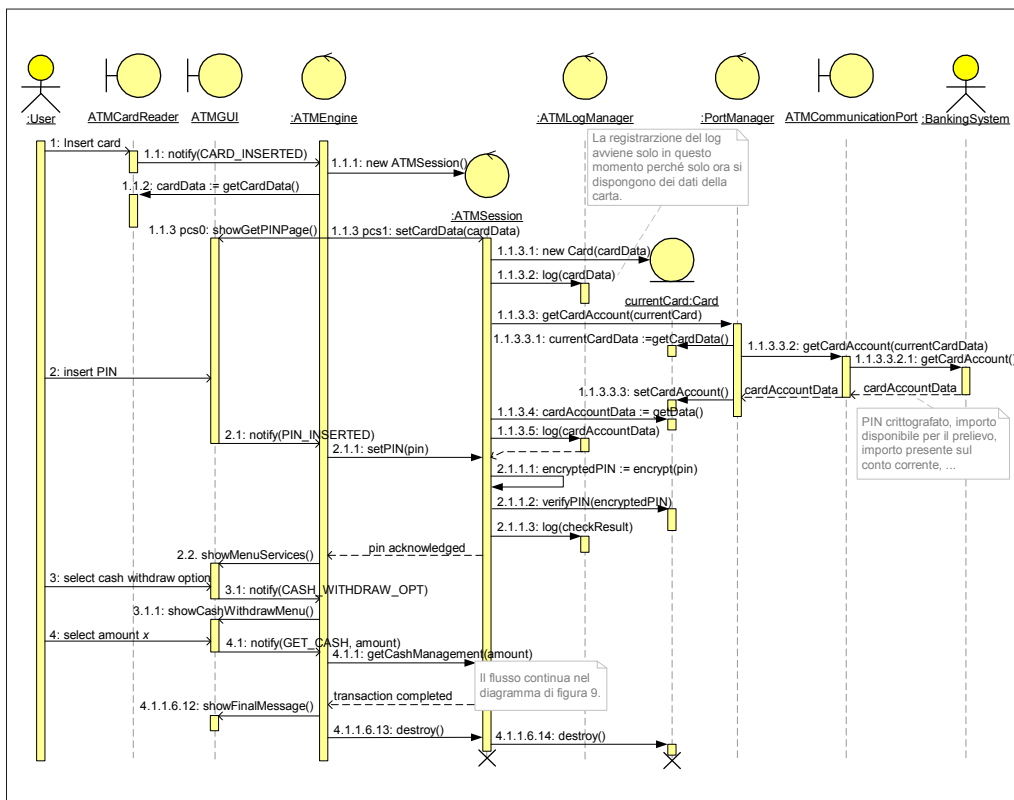
Nei diagrammi delle figg. 9.4 e 9.5 si sarebbe potuto indicare la distruzione dei vari bean e dei relativi `ServiceFactory`. Ciò però sarebbe stato corretto solo da un punto di vista logico poiché i container commerciali, per migliorare le prestazioni, cercano di gestire pool di componenti. Quindi, quando possibile, invece di distruggerli tentano di riutilizzarli.

Il diagramma in fig. 9.7 dovrebbe risultare abbastanza chiaro ma è opportuno spendere qualche parola su business object e integration layer. Una delle responsabilità dello strato di business object è quella di creare la classe che si occupa di automatizzare la connessione al database (`DBConnManager`) e, alla fine, di rilasciare opportunamente (chiudere) tutte le risorse utilizzate (`Connection`, `Statement` e `ResultSet`). Non vi è alcun riferimento alla gestione del pool di connessioni al database in quanto ormai disponibile in tutti gli application server (`DataSource`). Da notare che, trattandosi di ambiente Java, l'effettiva distruzione dell'oggetto `DBConnManager` non può essere gestita programmaticamente — gli adepti del linguaggio C++ sono ben consapevoli delle limitazioni che ciò comporta, come l'impossibilità della gestione del rilascio delle risorse nel distruttore dell'oggetto — ma avviene qualche tempo dopo che il riferimento viene rimosso (esecuzione del Garbage Collector).

Come si può notare, l'interrogazione del database avviene favorendo il caso in cui l'utente inserisca tutti i dati correttamente (fig. 9.2). Con una sola interrogazione si richiede di reperire l'utente in base alla stringa di login specificata, si effettua un join con la tabella della password aggiungendo il vincolo relativo al valore della password inserito, si inserisce la clausola che l'utente sia in uno stato operativo, si effettua un join con il profilo attivo, e così via. Nel caso in cui l'interrogazione non restituisca alcun risultato (autenticazione fallita), è necessario eseguire una serie di ulteriori query per identificare il motivo del fallimento ed eventualmente intraprendere opportune contromisure. Da notare che nel diagramma, per motivi di rappresentazione grafica, è stata omessa la gestione dello scenario relativo al fallimento dell'autenticazione, del numero massimo di tentativi consecutivi falliti e dell'intercettazione di tentativi di intrusione.

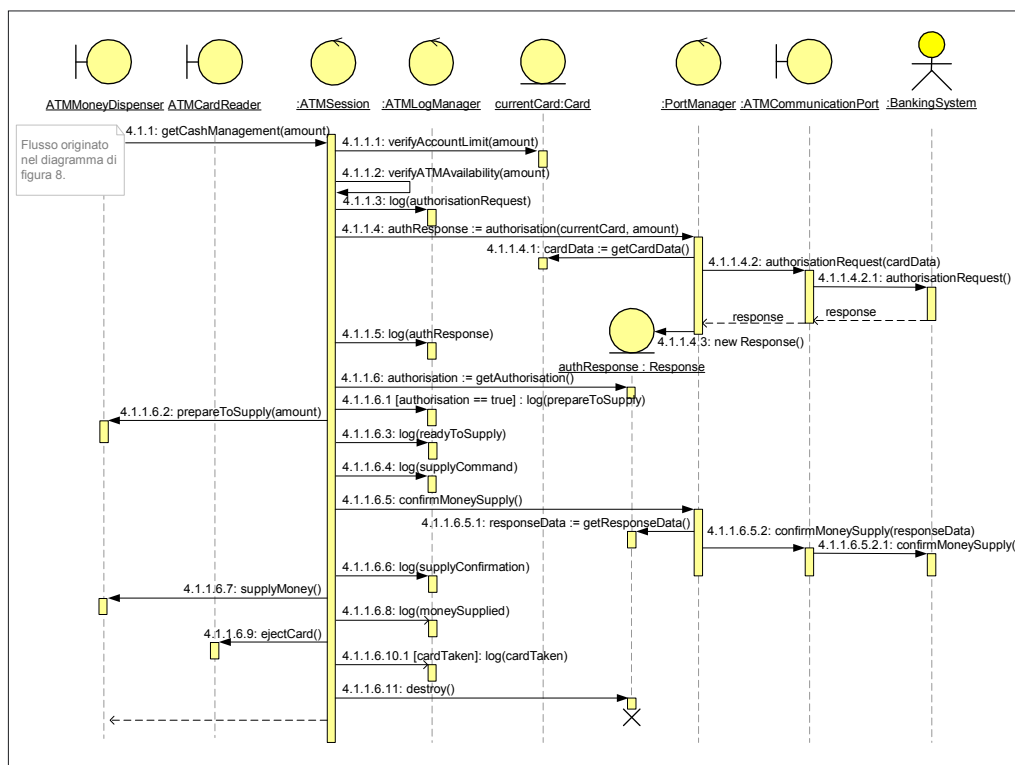
Durante la modellazione del comportamento dinamico, spesso accade di dover modellare interazioni abbastanza complesse tanto da non risultare facilmente esprimibili attraverso un solo diagramma, oppure di dover includere una medesima iterazione in diversi diagrammi di sequenza. In tutti questi casi è possibile realizzare un apposito diagramma dedicato alla sottosequenza da includere in altri diagrammi. In UML, il collegamento tra i diagrammi si ottiene mostrando, nel diagramma includente, uno stimolo o messaggio che parte da una determinata attivazione e non giunge a nessun'altra; nel diagramma incluso, invece, la situazione è speculare: uno stimolo o messaggio che giunge in un'apposita attivazione, ma non parte da nessun'altra. Qualora si decidesse di ricorrere a questa tecnica, è

Figura 9.8 — Esempio di diagramma di analisi relativo al servizio di prelievo contante da uno sportello Bancomat (ATM, Automated Telling Machine). In particolare in questa figura si è focalizzata l'attenzione sul servizio di autenticazione utente. Si è assunto che il controllo del PIN non avvenga in locale (rendendo non necessaria la memorizzazione del PIN sulla carta), bensì attraverso l'accesso a un opportuno server remoto. Ciò permette anche di memorizzare centralmente eventuali richieste di autorizzazione fallite. Si tratta di un accorgimento utile per evitare la possibilità di poter eseguire svariati tentativi in diversi Bancomat. Nel diagramma, la comunicazione con il server remoto avviene mentre l'utente è intento a impostare il PIN. Ciò è evidenziato per mezzo delle iterazioni sincrone: 1.1.3 pcs0 e 1.1.3 pcs1. Trattandosi di un diagramma al livello di analisi, sono stati inseriti diversi macrooggetti che, in fase di disegno, si prestano a essere opportunamente decomposti in oggetti più semplici, come per esempio l'astrazione della carta (Card) che contiene dettagli relativi al conto corrente, ai limiti previsti, ecc.



fortemente consigliato utilizzare appositi campi note da associare a questi messaggi o simboli di collegamento (*cfr* figg. 9.8 e 9.9).

Figura 9.9 — Diagramma di analisi mostrante in dettaglio il servizio del ritiro del denaro.



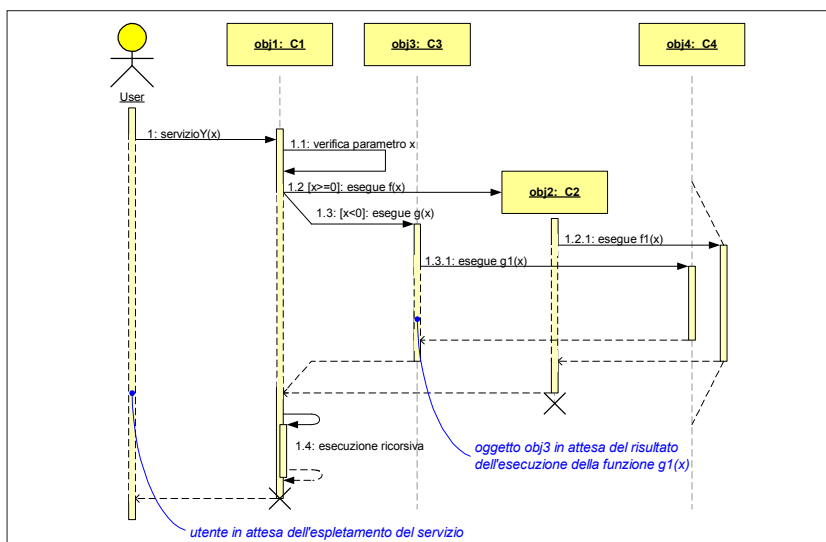
Nei diagrammi di sequenza è possibile illustrare comportamenti condizionali spezzando la linea di vita di un'istanza in due o più linee di vita concorrenti in cui ciascuna rappresenta uno specifico comportamento condizionale (fig. 9.10, stimoli 1.2 e 1.3). Queste linee di vita possono eventualmente ricongiungersi in un punto conveniente. I luoghi in cui avviene la diramazione sono mostrati attraverso diverse frecce uscenti dallo stesso punto. Queste potrebbero consistere in esecuzioni parallele o semplicemente alternative. Spesso è conveniente mostrare nelle etichette associate alle frecce le condizioni che danno luogo alla relativa diramazione.

Da notare che non sempre è necessario ripartire le linee di vita per mostrare comportamenti condizionali. Per esempio, nella fig. 9.5 è mostrato un comportamento condizionale in modo ordinario senza richiedere suddivisioni delle linee di vita. Le azioni appartenenti al punto 2.1 (2.1.1. [ejbHome == null] context := getInitialContext(), 2.1.2. [context != null] ejbHome := context.lookup(), 2.1.2.1. lookup() e 2.1.3. cacheEJBHome(ejbHome)) avvengono solo nel caso in cui la home interface dell'EJB richiesto non sia presente nella cache.



Visualizzare diversi scenari in uno stesso diagramma di sequenza non sempre produce una rappresentazione grafica gradevole e facilmente leggibile. Al fine di produrre diagrammi più chiari e facilmente comprensibili, spesso si preferisce realizzarne diverse versioni, ciascuna mostrante uno specifico scenario/condizione (cfr figg. 9.10-14). Il lato negativo di questo approccio è l'aumento del carico di lavoro necessario per realizzare e per mantenere aggiornati i diagrammi a seguito di variazioni (la modifica di un diagramma tende a ripercuotersi su diversi altri). Chiaramente questa tecnica non è utilizzabile qualora si vogliano mostrare comportamenti concorrenti. In tal caso non c'è alternativa: è necessario mostrare i diversi processi nello stesso diagramma. Da tener presente che per la visualizzazione di flussi di controllo concorrenti (thread), il formalismo dei diagrammi di sequenza non fornisce particolari costrutti e quindi non sempre risulta particolarmente idoneo a tal fine (si consideri la fig. 9.18). Qualora si abbia la necessità di mostrare comportamenti concorrenti non banali, probabilmente è il caso di considerare il ricorso ad altri formalismi, quali quello dei diagrammi di collaborazione e di attività. Quest'ultimi sono particolarmente indicati per la modellazione del comportamento parallelo grazie alla presenza di opportuni elementi grafici, mentre attraverso la notazione dei diagrammi di collaborazione si riesce comunque a evidenziare esecuzioni parallele attraverso la grammatica delle etichette associabili agli stimoli/messaggi.

Figura 9.10 — Diagramma di sequenza (dalle specifiche OMG UML) con l'utilizzo degli elementi: localizzazione del flusso, invocazione di un metodo dello stesso oggetto, esecuzione condizionale, ricorsione e distruzione.



Il periodo di tempo durante il quale un'istanza esegue delle azioni, sia direttamente sia attraverso procedure dipendenti, è evidenziato per mezzo di un'attivazione (*focus of control*, localizzazione del controllo). Le attivazioni sono mostrate graficamente per mezzo di rettangoli sovrapposti alle linee di vita. Il lato in alto è allineato con l'istante di tempo di avvio dell'attivazione (tipicamente corrispondente con la ricezione di un messaggio), mentre quello in basso con l'istante di tempo di completamento (cui spesso è associata la freccia di ritorno).

Le attivazioni possono essere corredate da apposite etichette — eventualmente inserendole a fianco del simbolo oppure sul lato sinistro — riportanti informazioni relative all'attivazione stessa o ai relativi obiettivi.



Sebbene lo standard UML preveda la possibilità di specificare delle etichette a fianco alle attivazioni, si tratta di un accorgimento inconsueto e poco pratico; ordinariamente si preferisce assegnare il compito di indicare l'obiettivo o l'azione di un'attivazione all'etichetta associata alla freccia entrante (il messaggio).

Spesso accade che una precisa interazione sia ripetuta più volte, ossia che "l'emissione" di un preciso insieme di stimoli possa avvenire diverse volte. In tali circostanze è possibile raggruppare l'insieme delle frecce che danno luogo alla "sottoiterazione" e evidenziarle — attraverso un opportuno rettangolo — come iterazione. La condizione dell'iterazione è riportata di solito alla fine della stessa.

Gli stimoli e i messaggi seguono le convenzioni indicate nella sezione *Messaggi e stimoli*. Nell'ambito dei diagrammi di sequenza sono rappresentati per mezzo di frecce, tipicamente orizzontali, aventi la coda nel rettangolo che mostra l'attivazione dell'istanza mittente e la testa nell'attivazione di quella ricevente. Qualora si voglia evidenziare un messaggio spedito da un'istanza a sé stessa, la freccia parte e arriva all'attivazione della stessa istanza (figura 9.10, 1.1. *verifica parametro x*). Da notare che, alla ricezione del messaggio, non è necessario evidenziare un'ulteriore attivazione sovrapposta a quella esistente. Questo accorgimento è invece necessario nell'ambito di invocazioni ricorsive (figura 9.10, 1.4. *esecuzione ricorsiva*).



Nelle precedenti versioni dello UML, sia l'invocazione da parte di un'istanza di propri metodi (*self-call*), sia le chiamate ricorsive potevano venir enfatizzate annidando la barretta dell'attivazione. Nella versione UML 1.4 non vi è menzione di particolari rappresentazioni grafiche da attuare per evidenziare autoinvocazioni, mentre è specificato che le chiamate ricorsive vadano rappresentate graficamente annidate, ossia riportando un'ulteriore barra di attivazio-



ne, allineata alla chiamata ricorsiva, leggermente spostata a destra, creando l'idea di una pila.

Da un punto di vista prettamente notazionale potrebbe essere una valida norma evidenziare esclusivamente chiamate ricorsive. In questo modo si crea la differenziazione tra autoinvocazioni (evidenziata attraverso una freccia che torna sull'attivazione invocante) e invocazioni ricorsive (mostrata per mezzo dell'attivazione annidata). Da un punto pratico, in tutti quei casi in cui è necessario mostrare del comportamento (scambio di messaggi/stimoli) all'interno di un'autoinvocazione, potrebbe risultare utile mostrare, anche in questo caso, un'ulteriore attivazione annidata. In effetti, in entrambe le situazioni l'attivazione chiamante deve mettersi in uno stato di attesa fin quando la procedura invocata termini la propria esecuzione per rientrare in possesso del controllo. Dal punto di vista della gestione dello stack non vi è molta differenza: in entrambi i casi è necessario memorizzare i parametri e il punto di ritorno, mentre nel caso di invocazioni ricorsive è necessario memorizzare anche lo stato degli attributi locali alla procedura presente nell'istante precedente l'invocazione.

Le frecce sono tipicamente rappresentate orizzontalmente. Ciò indica sia che l'intervallo di tempo necessario per la trasmissione dello stimolo è del tutto trascurabile comparato con la durata dell'attivazione, sia che l'evento è atomico, ossia che, nell'ambito dello scenario, durante l'intervallo temporale in cui avviene la trasmissione, null'altro può accadere. Queste assunzioni sono valide nella maggior parte dei casi (come per esempio invocazione di metodi), mentre potrebbero non esserlo in particolari contesti. In tal caso, messaggi o stimoli sono illustrati attraverso una freccia inclinata verso il basso (la testa si trova verticalmente più in basso rispetto alla coda, fig. 9.3, d. *instradamento*).



I valori prodotti dall'esecuzione di un'attivazione possono essere specificati tramite opportuna freccia di ritorno, rappresentata con tratteggio discontinuo che può essere omessa — ciò aiuta a rendere i diagrammi più leggibili — in quanto implicita nella fine dell'attivazione. In particolare, si assume che a ogni chiamata sia associato un valore di ritorno eventualmente specificato della freccia indicante l'avvio. La freccia di ritorno invece va necessariamente visualizzata nei casi di scambio asincrono di messaggi o in scenari relativi a calcoli paralleli.

In particolari occasioni potrebbe aver importanza distinguere il periodo di tempo durante il quale una particolare istanza è attiva (*focus of control*) ed esegue delle operazioni da quello in cui, pur essendo attiva, si trova in una sorta di stato di attesa (per esempio

perché l'espletamento del proprio servizio dipende dai risultati prodotti dall'esecuzione di operazioni fornite da altre istanze). Il primo caso si mostra con il classico rettangolo dell'attivazione, mentre per il secondo è possibile rappresentare il rettangolo tratteggiato durante il lasso di tempo di "attesa" (figura 9.10). Da tener presente che in caso di una collaborazione sincrona è implicito (nella natura della stessa del tipo di interazione) che l'oggetto chiamante resti in uno stato di attesa per tutto il tempo necessario all'oggetto chiamato per terminare la propria elaborazione.

Nel contesto di un messaggio, soprattutto in ambiti di scenari di sistemi real-time, può risultare necessario dover specificare diversi istanti di tempo. Gli esempi più evidenti sono quello del momento in cui avviene l'invio e quello della ricezione, quello del time-out, ecc. In genere ha senso specificare gli istanti di tempo nei casi in cui sia necessario dichiarare dei vincoli a questi relativi. Per esempio, nel diagramma di sequenza relativo a una connessione telefonica (fig. 9.3), lo stimolo della richiesta della linea è denominato `sollevaRicevitore` e quello di ritorno è indicato con il nome `ricezioneTono`; potrebbe aver senso specificare il vincolo: $b.receiveTime - a.sendTime < 1 \text{ sec}$, dove `a` è istanza dello stimolo `sollevaRicevitore` e `b` è istanza di `ricezioneTono`. In altre parole, si sancisce che l'intervallo di tempo che va dalla richiesta del tono "libero" alla relativa ricezione, debba essere inferiore al secondo. Questi tipi di vincolo, nei diagrammi di sequenza dovrebbero essere visualizzati nel margine sinistro, allineati con la freccia.

Utilizzo

Il formalismo dei diagrammi di sequenza è uno strumento dello UML molto apprezzato e utilizzato nella progettazione dei sistemi informatici e non solo. Grazie alla sua semplicità, immediatezza e duttilità, si presta a essere impiegato in diverse fasi del processo di sviluppo del software: dall'analisi dei requisiti, dove è efficace per illustrare o analizzare l'interazione tra gli utenti e il sistema da realizzare, alla fase di disegno, in cui è utilizzato per mostrare il comportamento dinamico di alcune parti del sistema, per provarne la correttezza, e così via.

Precisamente, il formalismo dei diagrammi di sequenza, nel contesto dei processi di sviluppo del software, si presta ad essere impiegato nelle fasi di analisi dei requisiti, e analisi e disegno: se ne esaminerà l'uso di seguito.

Analisi dei requisiti

In questo stadio, i diagrammi di sequenza possono essere adottati per illustrare graficamente il comportamento dinamico dei casi d'uso; più precisamente dei corrispondenti *scenario*. Data la loro semplicità, si prestano a essere facilmente compresi anche da un pubblico con limitate conoscenze informatiche (quale è, tipicamente, quello degli utenti).



Qualora si decida di adottare i diagrammi di sequenza come supporto alla fase di analisi dei requisiti, si consiglia di utilizzarli per illustrare esclusivamente i requisiti funzionali del sistema e i modi di interagire con essi, evitando quanto più possibile dettagli tecnici appartenenti allo spazio delle soluzioni. È opportuno inoltre limitare la collaborazione a poche macroentità e utilizzare l'etichetta dei messaggi per rappresentare una descrizione, quanto più possibile in linguaggio naturale, delle informazioni scambiate, evitando etichette simili all'invocazione dei metodi di un oggetto (cfr figg. 9.2, 9.11-15).

In questa fase i principali benefici derivanti dall'utilizzo dei diagrammi di sequenza sono essenzialmente legati all'adozione di un formalismo grafico: risultano più accattivanti di uno "spento" testo, aumentano il livello di chiarezza e immediatezza, accrescono la capacità di comprensione e memorizzazione di quanto illustrato, agevolano il dialogo con l'utente, ecc. Gli svantaggi sono legati essenzialmente all'elevata quantità di tempo necessaria per produrre e mantenere i vari diagrammi. Questo, tipicamente, è di un ordine di grandezza superiore a quello richiesto dalla manutenzione di un documento testuale. Si consideri l'esempio illustrato: per un servizio non eccessivamente complesso, quale l'autenticazione utente, è stato necessario realizzare ben sei diagrammi: uno per lo scenario principale (fig. 9.2) e cinque per quelli di eccezione (figg. 9.11-15). Inoltre, una modifica a un diagramma (per esempio quello di fig. 9.11) tende a generare la necessità di modificare tutti i diagrammi successivi.

Figura 9.11 — Servizio autenticazione utente. Scenario di eccezione relativo al fallimento del riconoscimento della stringa di login.

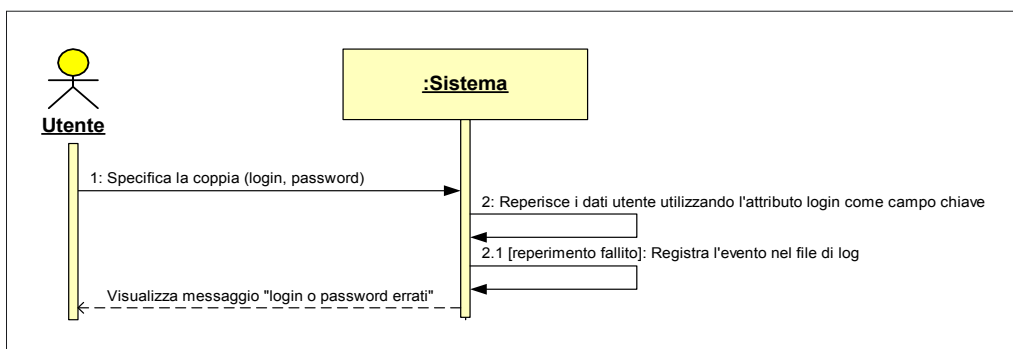


Figura 9.12 — Servizio autenticazione utente. Scenario di eccezione relativo al fallimento del riconoscimento della password. Si può notare che, limitatamente allo scenario riproposto, ha senso bloccare l'utente qualora il numero massimo consentito di tentativi fallimentari consecutivi sia stato raggiunto.

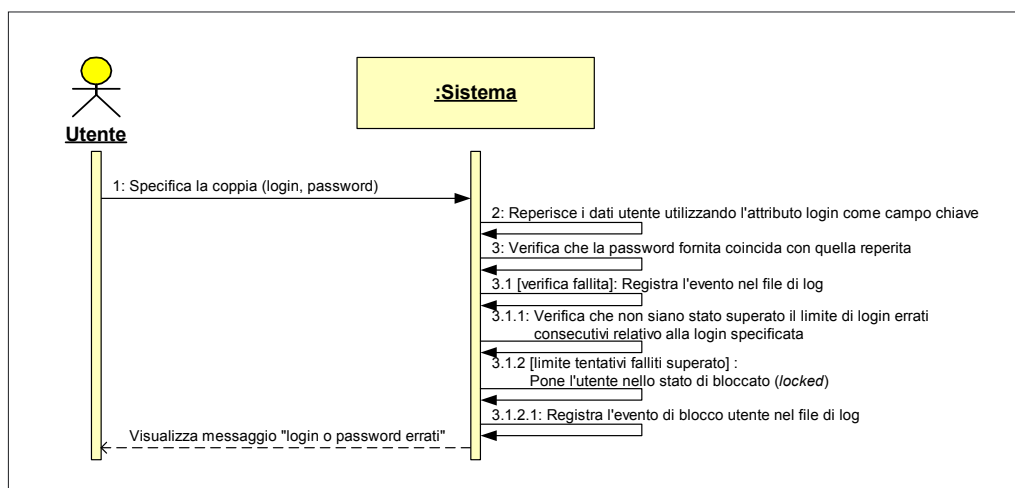


Figura 9.13 — Servizio autenticazione utente. Scenario di eccezione relativo all'identificazione di uno stato non valido della password (per esempio expired)

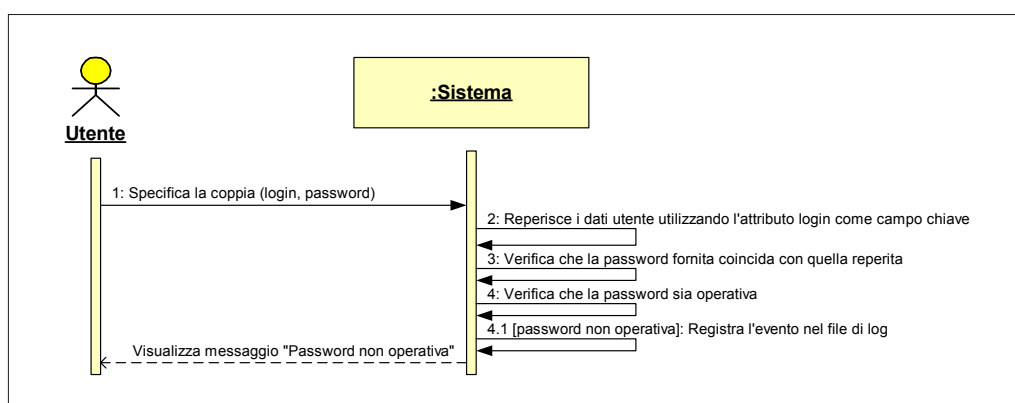


Figura 9.14 — Servizio autenticazione utente. Scenario di eccezione relativo all'identificazione di uno stato non valido dell'utente (per esempio locked).

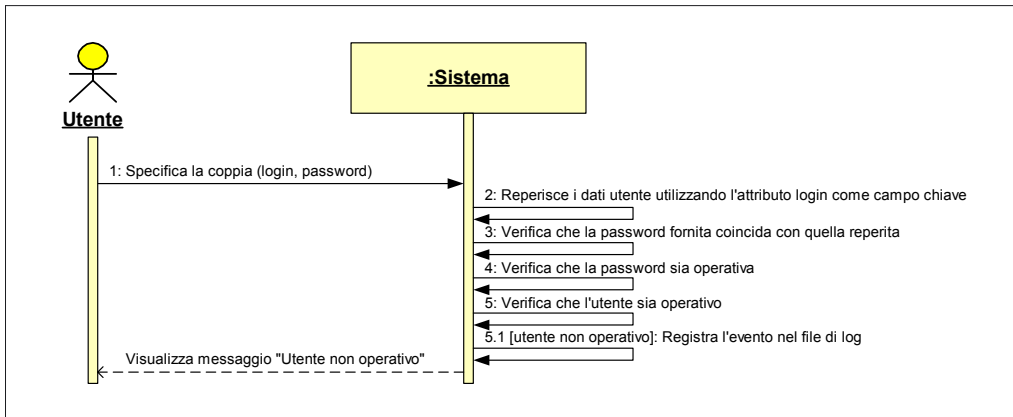
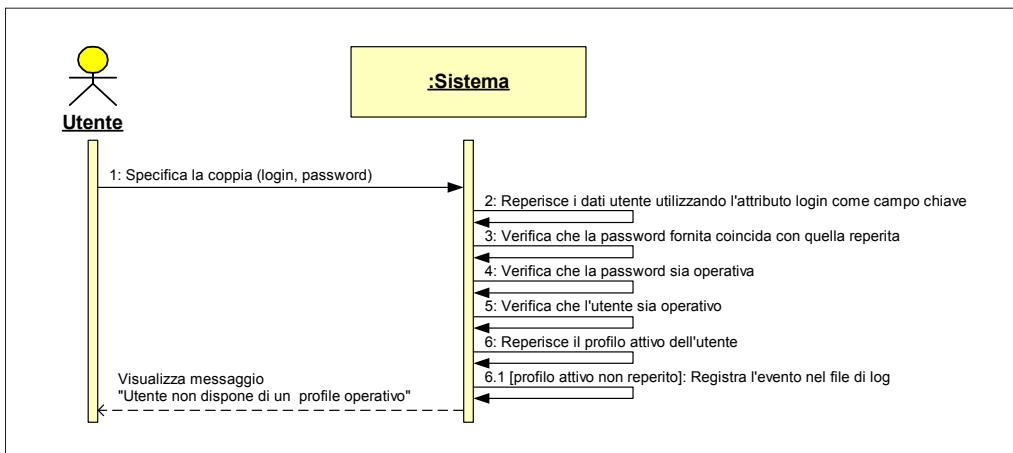


Figura 9.15 — Servizio autenticazione utente. Scenario di eccezione relativo all'identificazione della mancanza di un profilo attivo per l'utente.





Visti i problemi legati all'utilizzo dei diagrammi di sequenza nel processo di analisi dei requisiti, probabilmente, in tale fase, è opportuno evitarne l'impiego sistematico. È tuttavia consigliabile utilizzarli come strumento di supporto, magari per mostrare scenari particolarmente importanti o controversi, oppure per visualizzare dinamiche che coinvolgono diversi casi d'uso, al fine di evidenziare più chiaramente il quadro di insieme di servizi particolarmente complessi. Non è infrequente poi il caso in cui siano impiegati come strumento di indagine per investigare e quindi definire con gli utenti il funzionamento del sistema (requisiti funzionali), quantunque per questa attività i diagrammi di attività risultino più efficaci: evidenziano chiaramente le responsabilità tra i vari attori ed il sistema.

Analisi e disegno

In queste fasi è possibile avvalersi del formalismo dei diagrammi di sequenza per molteplici motivi (cfr figg. 9.4-5, 9.7-9; per quasi tutti gli utilizzi riportati in questo punto è possibile utilizzare anche la notazione dei diagrammi di collaborazione):

- navigare formalmente i modelli di struttura statica, definiti attraverso appositi diagrammi come quello delle classi, al fine di verificare che questi siano effettivamente in grado di erogare i servizi richiesti dal sistema. Spesso, realizzato un modello di struttura statica, non si è completamente sicuri che questo sia effettivamente in grado di realizzare i servizi per i quali è stato progettato. Un buon metodo di verifica consiste nel realizzare un diagramma di interazione — eventualmente anche solo su lavagna o su un foglio di carta — che renda esplicito il comportamento dinamico, ossia la collaborazione che deve instaurarsi tra le varie componenti al fine di realizzare i servizi richiesti. La realizzazione di diagrammi dinamici permette anche di evidenziare eventuali grovigli del modello — per esempio esorbitante scambio di messaggi tra alcune parti, oppure collaborazioni smisuratamente popolate — o aree non ben curate che quindi si prestano ad essere migliorate. Verosimilmente, è più opportuno spendere qualche ora tracciando dei diagrammi di sequenza, piuttosto che trovarsi nella situazione di dover buttare via il codice prodotto perché magari il disegno iniziale non era corretto.
- fornire ai programmatori la descrizione del comportamento dinamico da implementare, ovvero il modo in cui le istanze delle classi interagiscono tra loro per fornire i servizi richiesti. Per esempio, anche un diagramma delle classi non complesso come quello di fig. 9.16, sebbene utilissimo, da solo potrebbe non chiarire esattamente il meccanismo di funzionamento del pattern Observer descritto (cfr [BIB04]).

Figura 9.16 — Diagramma delle classi rappresentante il pattern dell'Observer [BIB04].

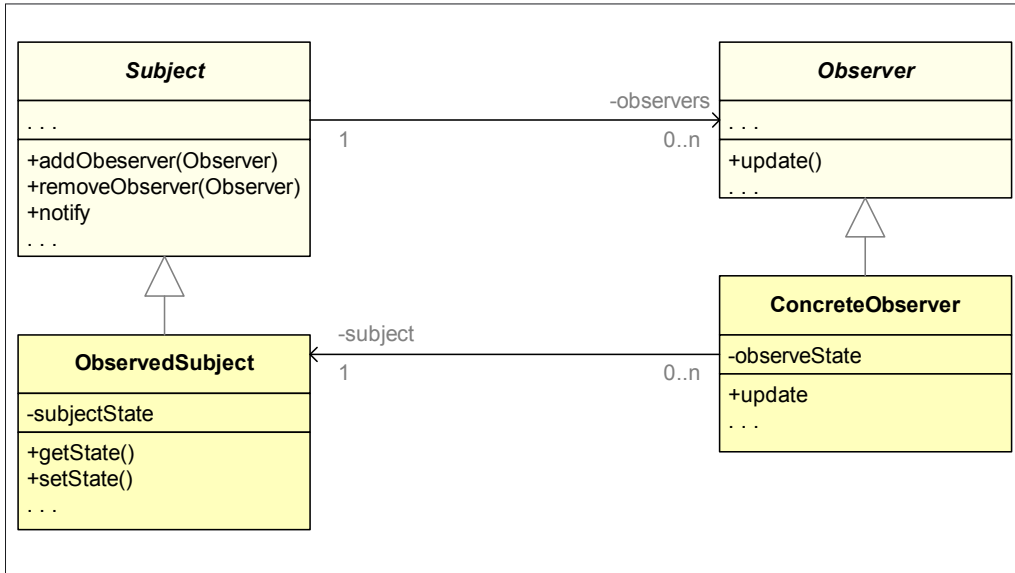
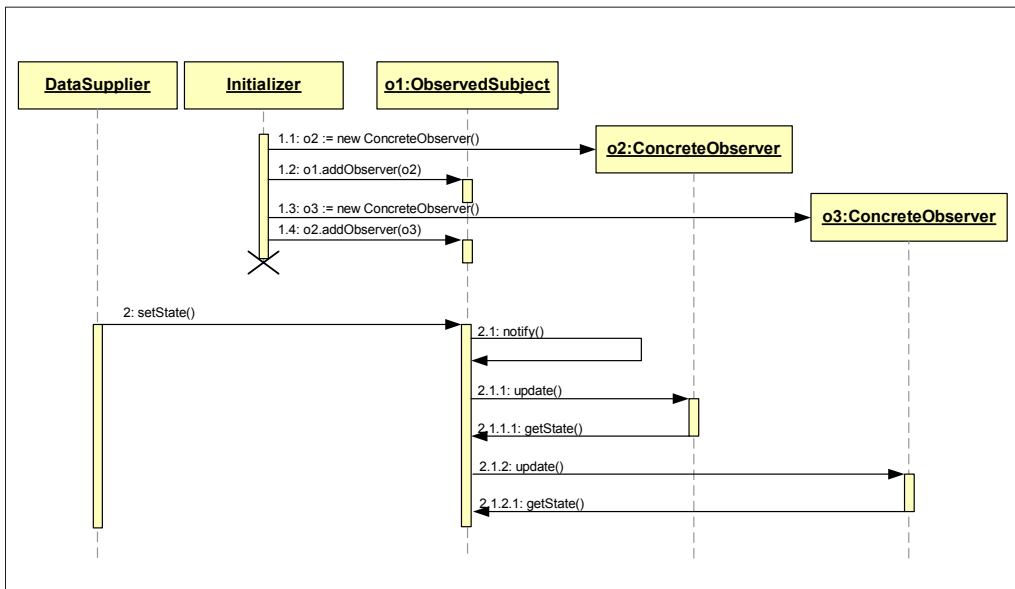


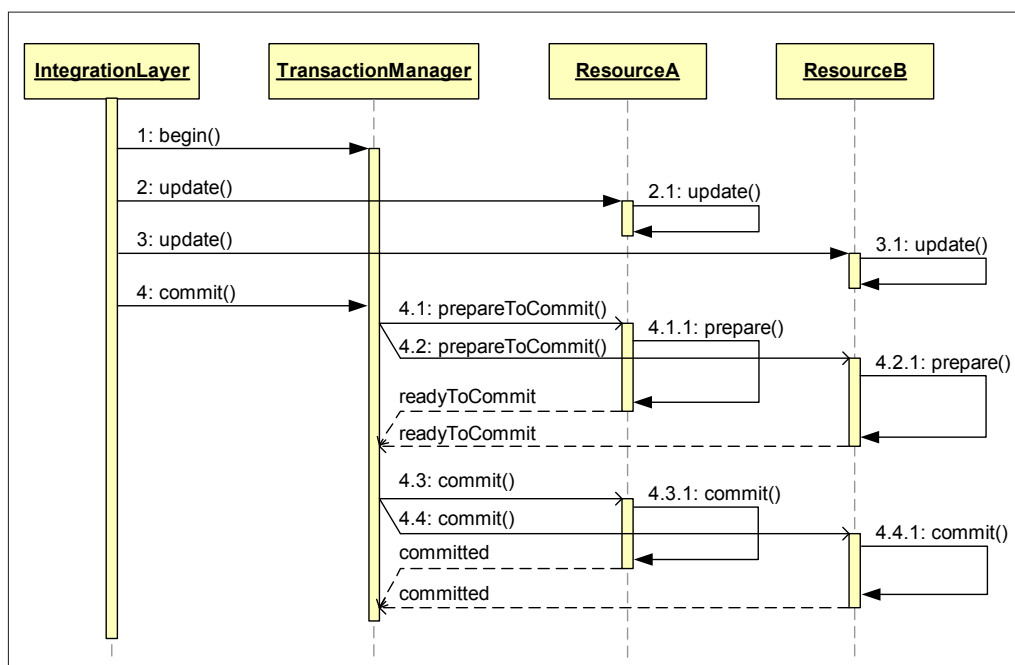
Figura 9.17 — Diagramma di sequenza illustrante il funzionamento del pattern dell'Observer.



Si può notare che è sufficiente disporre di un conciso diagramma dinamico, come quello di fig. 9.17, per rendere tutto immediatamente chiaro. Più in generale i diagrammi di sequenza, grazie al loro grado di intuitività, si dimostrano un ottimo strumento di supporto alla didattica. Tipicamente la realizzazione di qualche diagramma di sequenza risulta un ottimo espediente per rendere più comprensibili certe nozioni.

- realizzare un'ottima documentazione del sistema. Disporre dei diagrammi di struttura è decisamente molto importante, ma questi da soli non sono sufficienti giacché non forniscono alcuna indicazione circa le dinamiche del sistema, per le quali è

Figura 9.18 — Utilizzo del formalismo dei diagrammi di sequenza per illustrare, molto generalmente, il funzionamento del meccanismo del two-phases-commit. Nel diagramma si è cercato di mostrare del comportamento parallelo (nella fase di prepare to commit e in quella di commit). Sebbene si tratti di un diagramma abbastanza semplice, il risultato evidenzia le limitazioni della notazione dei diagrammi di sequenza nell'illustrare esecuzioni parallele.



necessario utilizzare i diagrammi di interazione. Da tener presente però che, se si avesse l'esigenza di enfatizzare l'esecuzione parallela di particolari processi, sarebbe opportuno valutare la possibilità di ricorrere ai diagrammi di attività.

- agevolare il processo di individuazione dei “colli di bottiglia”. Specificando nei diagrammi di sequenza le stime dei tempi richiesti per eseguire ogni singola funzione associata alla ricezione di un messaggio, è possibile apprezzare il tempo necessario per i servizi illustrati. Qualora questo sia eccessivo, è ancora possibile evidenziare quale sia il punto che richiede maggiori risorse in termini di tempo e quindi verificare la possibilità di ottenere lo stesso risultato con strutture o interazioni diverse.
- verificare la complessità del sistema e la relativa qualità in generale. Qualora, nel tracciamento dei diagrammi di interazione, si evidenzia che una determinata istanza di una classe riceve molte invocazioni, magari eterogenee, potrebbe essere un'indicazione che la relativa classe possiede troppe responsabilità (scarsa coesione), che la complessità è eccessiva, che determinati servizi sono richiesti troppo frequentemente, ecc.
- implementare una tecnica per la realizzazione di modelli di analisi e disegno di struttura statica. Alcuni autori suggeriscono di utilizzare i diagrammi di interazione — e in particolare quelli di sequenza — per dar luogo a prime versioni dei modelli di analisi e/o disegno. In sostanza si scoprono le varie classi secondo cui organizzare il sistema attraverso l'analisi dei messaggi scambiati, utilizzando un approccio rigorosamente top-down. Si realizza una prima versione del diagramma dinamico visualizzando esclusivamente poche entità fondamentali che costituiranno il sistema. Si comincia con il tracciare lo scambio di messaggi tra gli attori e il sistema, e quindi si procede con una seconda versione in cui le prime macroentità sono analizzate e scomposte in ulteriori unità di maggior dettaglio, e via discorrendo. Questo processo di raffinamento per passi successivi termina quando si ritiene di aver raggiunto un opportuno livello di dettaglio. Chiaramente, non è necessario ripartire ogni volta da zero, qualora si siano già realizzati alcuni diagrammi (magari anche attingendoli da sistemi precedenti) è possibile cercare di riutilizzarne i contenuti (le classi o almeno la struttura), integrando così un approccio di tipo bottom-up. Sebbene questa tecnica offra il grosso vantaggio di disporre di un'eccezionale documentazione del sistema a processo ultimato, gli svantaggi sono diversi. In primo luogo esiste il ricorrente problema di un adeguato utilizzo della risorsa tempo (produrre tutti questi diagrammi e soprattutto mantenerli aggiornati richiede un impiego di energie non indifferente), poi ben pochi tecnici posseggono un livello di esperienza tale da renderli in grado di utilizzare proficuamente questo approccio, ecc.

Per via di queste considerazioni, l'autore del testo conserva non poche perplessità sull'efficacia di questa strategia.



Lo stile

Si passerà ora all'illustrazione di una serie di linee guida che permettono di realizzare diagrammi di sequenza di maggiore qualità e quindi più immediatamente e semplicemente comprensibili.

Stabilire un'unica direzione

Nei limiti del possibile è conveniente organizzare i diagrammi di sequenza affinché emerga una sola direzione del flusso dei messaggi. Ora, vista e considerata la convenzione di scrittura e lettura del mondo occidentale — da sinistra verso destra e dall'alto verso il basso — è conveniente rispettare tali norme. Qualora poi si lavori per organizzazioni appartenenti alla cultura araba o orientale è possibile adeguare lo stile alle convenzioni locali (scrittura da destra a sinistra). Restando in Occidente, è opportuno disegnare i diagrammi di sequenza iniziando dall'angolo in alto a sinistra proseguendo verso quello in basso a destra, cercando di organizzare le varie entità in modo tale che se un'istanza invia un messaggio da un'altra, il mittente preceda il destinatario (si trovi alla sua sinistra).

È opportuno tenere a mente e rispettare l'ordinamento delle comunicazioni: un messaggio che, nello stesso diagramma, compare più in basso rispetto a un altro, indica che avviene in un istante di tempo successivo rispetto al messaggio più in alto. Con riferimento alla fig. 9.17, le varie istanze sono state posizionate in modo tale che tutti i messaggi posseggano un verso da sinistra verso destra. Inoltre è stato rispettato l'ordinamento dei messaggi: lo stimolo 1.2. `o1.addObserver(o2)` avviene in un tempo successivo allo stimolo 1.1. `o2 = new ConcreteObserver()` localizzato più in alto nel diagramma.

Organizzare i diagrammi di sequenza secondo questa direttiva li rende più naturali, chiari, meno ingarbugliati, più facilmente comprensibili e memorizzabili. Chiaramente non sempre ciò è possibile. Per esempio, in fig. 9.7, lo stimolo 2.2 `bean = getSecurityProviderBOBean()` è disegnato da destra verso sinistra, mentre nelle figg. 9.8 e 9.9 si è creato un conflitto tra la presente direttiva e quella successiva. In particolare l'evidenziazione della struttura multistrato dell'architettura ha generato la necessità di non rispettare l'ordine del flusso dei messaggi. Ciò nonostante si è preferito conferire maggiore importanza alla struttura dell'architettura in quanto, nel disegno globale, permette di raggiungere un maggiore livello di chiarezza e concorre a rinforzare le linee guide architettoniche.

Evidenziare la struttura multistrato

Come visto approfonditamente nel Capitolo 8 (cfr *Dipendenza del modello di analisi dall'architettura*), le architetture moderne prevedono un'organizzazione multistrato (*mul-*

ti-tiered), caratterizzata dalla suddivisione logica delle responsabilità del sistema in livelli, in modo che ciascuno si faccia carico di una sola tipologia di responsabilità. Il sistema pertanto è il risultato di una sequenza di strati, in cui ognuno fornisce servizi allo strato di livello superiore, eventualmente utilizzando altri servizi forniti dal livello inferiore. È importante che anche i diagrammi di sequenza siano organizzati enfatizzando questa struttura. Pertanto si inizia con le istanze degli attori posti a sinistra, poi si prosegue con le istanze dei moduli di interfaccia utente (le *boundary*), lo strato dei servizi business, quello degli oggetti business e lo strato di integrazione. In virtù di questa organizzazione, è logico attendersi che istanze appartenenti a strati adiacenti diano luogo a un certo scambio di messaggi, che invece è nullo tra elementi appartenenti a strati non adiacenti. Pertanto, realizzare diagrammi di sequenza organizzati enfatizzando la struttura a strati dell'architettura, oltre a rinforzare il concetto di organizzazione multi-tiered del sistema, permette di conferire una migliore organizzazione dei diagrammi, di renderli più lineari, ecc. (cfr figg. 9.8 e 9.9).



L'applicazione di queste regole potrebbe risultare problematica nella realizzazione dei diagrammi appartenenti al modello dei requisiti utente. In questa fase, aspetti appartenenti al dominio della soluzione non devono fare la loro comparsa e quindi non è opportuno introdurre concetti di architettura e di strutture a strati. Ciononostante, è possibile disegnare i diagrammi di sequenza, inserendo gli attori a sinistra, il sistema o le sue parti — che per forza di cose devono restare molto generiche — a destra, scegliendo le posizioni in funzione dei messaggi scambiati con le altre entità.

Alle regole succitate esiste una sorta di eccezione. Qualora un'interazione preveda la partecipazione di sistemi esterni che non avviano la transazione, è opportuno riportare questi ultimi all'estrema destra del diagramma. Per esempio, nei diagrammi delle figg. 9.8 e 9.9, l'attore `BankingSystem` è stato riportato all'estrema destra. Ciò inoltre ha generato la necessità di riportare un'apposita *boundary* (`ATMCommunicationPort`) rappresentante parte dello strato di integrazione. Allo stesso modo, nella realizzazione di diagrammi di sequenza relativi a scenari di casi d'uso (fase di analisi dei requisiti) che coinvolgono attori secondari (che quindi ricevono unicamente notifiche dal sistema) è consigliabile riportarli alla destra del diagramma.

Utilizzare i nomi in modo consistente

I diagrammi di sequenza sono utilizzati in diverse fasi dei processi di sviluppo del software con diverse finalità e quindi le “istanze” che vi compaiono provengono da manufatti di diversa natura. Molto importante è che le varie entità siano utilizzate in modo consistente

e che gli elementi siano prelevati, di volta in volta, dal relativo diagramma di struttura statica. Per esempio, nel realizzare diagrammi di sequenza per la descrizione degli scenari, gli attori inseriti devono essere gli stessi presenti nel modello dei casi d'uso (fig. 9.2); nella fase di analisi le entità illustrate devono essere prelevate dal modello di struttura statica di analisi, mentre, nella fase di disegno le entità devono provenire dal modello di struttura statica di disegno. Alla fine del processo, tutti i modelli devono essere consistenti tra loro e bilanciati. Qualora, disegnando un diagramma di sequenza, ci si dovesse accorgere della necessità di introdurre un elemento non presente nei diagrammi di struttura statica, è opportuno verificare se vi sia effettivamente la necessità di questo nuovo elemento (i diagrammi di sequenza sono anche un utile strumento di verifica) o se si stia cercando di creare un sinonimo di un elemento esistente o addirittura di stravolgere la struttura disegnata dal modello di struttura statica.

Mostrare un adeguato livello di dettaglio

La selezione del “giusto” livello di dettaglio da mostrare è un problema ricorrente nella realizzazione di tutti i diagrammi UML e non solo. Qualora il livello risulti eccessivamente astratto si corre il rischio di produrre diagrammi incomprensibili o che non forniscono alcuna informazione; d'altro canto, se il livello di dettaglio è eccessivo, si rischia di produrre diagrammi complessi, poco leggibili, difficili da produrre e mantenere, con il pericolo di utilizzare in modo non appropriato il fattore tempo e quindi il budget allocato al progetto.



Per quanto riguarda la produzione di diagrammi con uno scarso livello di dettaglio, ahimè, c'è poco da fare. L'unico consiglio può essere quello di sottoporre i diagrammi all'attenzione di altre persone estranee alla relativa produzione e quindi verificare con loro la comprensibilità di quanto modellato. Per quanto riguarda un eccessivo livello di dettaglio, è possibile tenere presenti alcuni consigli. Per esempio, qualora sia necessario impostare i dati di un particolare oggetto, invece di dar luogo ad una lunga lista di messaggi atti ad impostare il valore di ogni singolo attributo (`setXX`), è consigliabile mostrare un unico messaggio con la descrizione del tipo `setAttributes` o `setData` o `setInfo`. Lo stesso discorso vale qualora si desideri prelevare i valori degli attributi: specularmente al caso precedente è possibile mostrare un unico messaggio del tipo `getAttributes` o `getData` o `getInfo` (fig. 9.8, stimolo 1.1.3: `setCardData(cardData)`). Evitare dettagli arcinoti al lettore o di interesse non particolare per il contesto oggetto di studio. Qualora un messaggio consista nell'invocazione di un metodo che restituisce un valore, probabilmente non è il caso di mostrare tale parametro attraverso un apposito messaggio di ritorno (stereotipo della freccia tratteggiata). Qualora sia veramente necessario mo-



strare il parametro di ritorno, magari perché utilizzato in un futuro messaggio, è possibile specificarlo direttamente nell'etichetta relativa all'invocazione del metodo (fig. 9.7, 2.3: `userProfile = bean.authenticateUser(login, password)`). Ancora, dovendo mostrare un'interazione di una serie di componenti di un'architettura basata sulla piattaforma EJB probabilmente non è necessario mostrare nei vari diagrammi l'interazione con l'`InitialContext`, il `ServiceFactory`, ecc. (se necessario, questi dettagli possono essere rappresentati in un apposito diagramma). È sufficiente mostrare la richiesta dello specifico EJB, a un'opportuna classe del tipo `ServiceLocator`, e quindi mostrare i messaggi scambiati direttamente con il bean, ignorando (qualora non aggiungano particolare comportamento) l'invocazione della home e della remote interface (fig. 9.7).

Inserire le descrizioni dei flussi di messaggi

Spesso l'analisi dei diagrammi di sequenza, specie di quelli relativi a fasi a carattere tecnico, non risulta agevole. Ciò può essere imputabile a diversi fattori, come per esempio l'introduzione di un eccessivo numero di istanze, la visualizzazione di un eccessivo livello di dettaglio e così via. In ogni modo, la comprensione dei diagrammi di sequenza, specie di quelli appartenenti alle fasi di analisi e disegno, può essere semplificata inserendo opportuni commenti; un po' come si fa anche per il codice. Questi commenti dovrebbero essere inseriti all'estrema sinistra, e dovrebbero illustrare la logica della sequenza, specificando, di volta in volta, l'obiettivo che si intende raggiungere con una determinata sottosequenza di messaggi. Queste descrizioni, oltre a favorire la comprensione della logica dei diagrammi, dovrebbero fornire delle importanti informazioni di riferimento ai requisiti che li hanno generati (*traceability*), ossia alle sezioni dei casi d'uso che intendono risolvere.

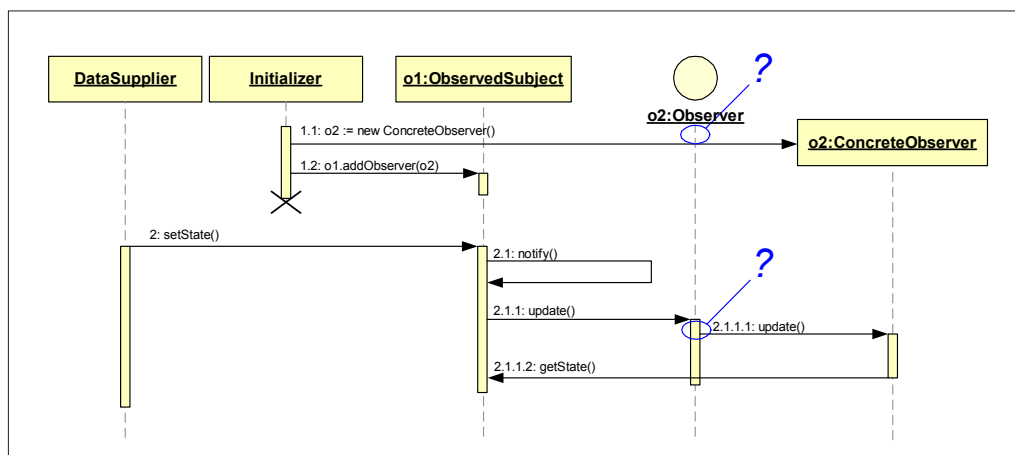
L'introduzione dei commenti, spesso preziosissimi nelle fasi di analisi e disegno, dovrebbe possedere una validità piuttosto relativa nelle fasi di analisi dei requisiti, in cui la descrizione degli stessi messaggi dovrebbe risultare più che sufficiente.

Rappresentazione del polimorfismo

Molto spesso, durante la modellazione delle dinamiche di un sistema, si rende necessario rappresentare del comportamento polimorfico nei diagrammi di interazione ed in particolare in quelli di sequenza. Una situazione tipica è di dover mostrare esplicitamente la creazione di oggetti i cui servizi, in un secondo momento, sono invocati attraverso il tipo esposto da un'opportuna interfaccia implementata dalla classe di cui l'oggetto è istanza (si consideri il diagramma di fig. 9.17).

In letteratura è possibile individuare alcune soluzioni degne di nota. Una delle più utilizzate prevede di mostrare due occorrenze dello stesso oggetto (identificato per mezzo

Figura 9.19 — Tecnica utilizzata per mostrare comportamento polimorfo.



del nome) attraverso altrettanti tipi: quello di un'opportuna interfaccia e quello della classe di cui è istanza (fig. 9.19).

Per quanto ciò sia lecito e possa offrire dei vantaggi (illustrare esplicitamente l'utilizzo di oggetti in maniera astratta, mostrare molti dettagli attraverso un solo diagramma, ecc.), da un punto di vista concettuale e semantico vi sono diversi problemi. Per esempio, nella prima occorrenza dell'oggetto *o2* (quella in cui il tipo è dato dall'interfaccia *Observer*), la relativa linea di vita asserisce che l'oggetto è in vita prima dell'inizio del diagramma, mentre la seconda occorrenza evidenzia la creazione dell'oggetto nel contesto del diagramma stesso (stimolo 1.1), quindi in un secondo momento. Ancora, l'invocazione del metodo `update()` (stimolo 2.1.1) provoca un'autoinvocazione o una delegazione (l'istanza *o2* di tipo *Observer* invoca sé stessa di tipo *ConcreteObserver*) del tutto artificiale. Entrambi gli inconvenienti potrebbero essere attenuati con opportune note aggiuntive. Si tratta comunque di una soluzione atta ad alleviare le anomalie, ma che di certo non le risolve.

A questo punto bisognerebbe porsi la domanda se è veramente il caso di procedere con l'utilizzo di questa tecnica o, meglio ancora, se è necessario mostrare nei diagrammi di interazione dettagli di questo tipo. Si tratta chiaramente di un'interrogazione retorica. Verosimilmente, nella stragrande maggioranza dei casi, non è necessario mostrare gli oggetti contemporaneamente sia attraverso il proprio tipo, sia attraverso quello di un'eventuale interfaccia. La visualizzazione del comportamento polimorfo si presta a essere risolto utilizzando congiuntamente i diagrammi di struttura statica. Come per tutte le altre tipologie di diagrammi, anche quelli di interazione forniscono una proiezione del sistema, dettagliata quanto si vuole, ma di certo non in grado di mostrare tutti i dettagli.

Pertanto è possibile ricorrere a soluzioni come quelle adottate nelle figure 9.16 e 9.17. Nel diagramma di struttura statica è mostrato che agli osservatori viene propagato lo stimolo di `update()` attraverso l'apposita interfaccia (`Observer`) di cui però non viene fatta assolutamente menzione nel relativo diagramma di sequenza, il quale si limita a mostrare i tipi concreti. L'utilizzo congiunto dei due diagrammi permette pertanto di illustrare chiaramente il comportamento polimorfico di oggetti senza dover ricorrere a particolari artifici.

Diagrammi di collaborazione

Definizione

I diagrammi di collaborazione mostrano grafici formati da entità interconnesse (tipicamente oggetti) eventualmente corredati dalla rappresentazione di comportamento dinamico. Tale comportamento include un insieme parzialmente ordinato di messaggi scambiati tra le entità cooperanti al fine di raggiungere un determinato risultato come, per esempio, la fornitura di un servizio all'utente, l'elaborazione di un algoritmo, e così via. Tipicamente i diagrammi di collaborazione costituiscono frammenti di un modello più grande e completo realizzati per uno scopo preciso. Ciò chiaramente è vero per la quasi totalità dei diagrammi realizzati durante un qualsiasi processo di sviluppo del software.

Il formalismo dei diagrammi di collaborazione prevede due aspetti: la struttura statica delle entità partecipanti ed — eventualmente — la descrizione della relativa rete di comunicazioni. Il primo aspetto in UML è definito *Collaborazione* e rappresenta la struttura delle entità partecipanti che recitano un preciso ruolo nell'espletamento di un servizio, corredate dalle relative relazioni, mentre la rete di comunicazioni messa in atto per realizzare il servizio è detta *Interazione* (consultare fig. 9.1). Precisamente, un'interazione contiene un insieme parzialmente ordinato di messaggi, ognuno dei quali specifica una comunicazione tra ruoli ben definiti che sono mittenti e/o destinatari dei vari messaggi.

Ogniqualevolta si realizza un diagramma di collaborazione, si definisce una *Interazione* nel contesto di una collaborazione. Precisamente, la realizzazione di un diagramma di collaborazione, prevede la seguente sequenza di attività:



1. definizione della collaborazione: si rappresenta l'insieme delle entità collaboranti corredate dalle relative relazioni;
2. modellazione dell'interazione: si riporta nel diagramma la sequenza parzialmente ordinata dei messaggi o stimoli che le varie entità si scambiano al fine di produrre il risultato desiderato;
3. specificazione delle etichette dei messaggi/stimoli: si utilizza la relativa



grammatica al fine di fornire maggiori dettagli circa l'interazione tra le varie entità (si visualizzano flussi di controllo condizionali, iterativi, esecuzioni parallele, ecc.).

La definizione formale di diagramma di collaborazione — in modo del tutto consistente al caso dei diagrammi di sequenza — prevede due versioni (fig. 9.1):

- la prima sancisce che un diagramma di collaborazione rappresenta una collaborazione (*Collaboration*) con eventualmente specificata una *Interazione*. Pertanto è fornito un insieme di ruoli (*ClassifierRole*) recitati da opportune istanze, corredati dai relativi legami instaurati nell'ambito dello scenario considerato, con eventualmente specificato l'insieme dei *Messaggi* che stabiliscono l'interazione che ha luogo tra queste istanze per raggiungere un determinato risultato;
- la seconda afferma che un diagramma di collaborazione rappresenta un *CollaborationInstanceSet* (insieme di istanze collaboranti) con eventualmente specificato un *InteractionInstanceSet* (insieme di istanze interagenti). Pertanto è costituito da una collezione di istanze (*Instance*), corredate delle relative relazioni e con eventualmente specificato l'insieme di *Stimoli* che specificano l'interazione che ha luogo per l'espletamento del task specificato.

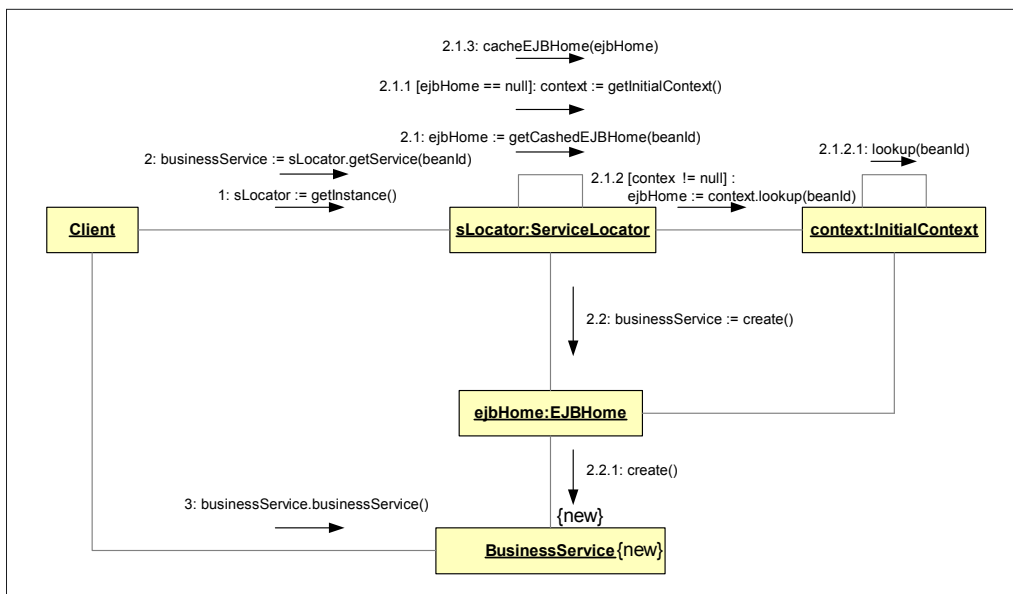
Da notare che la seconda definizione (detta al *livello di istanze*) è quella cui si è più comunemente portati a pensare: un diagramma di collaborazione rappresenta un grafo di oggetti con eventualmente specificati gli stimoli scambiati tra questi, necessari per realizzare il servizio. La prima (detta al *livello di specificazione*) invece, è meno intuitiva e rappresenta fondamentalmente un artificio tecnico elaborato al fine di poter includere nei diagrammi di collaborazione ulteriori elementi (specializzazioni dell'elemento *Classifier* del metamodello) che non siano unicamente oggetti (*Instance*).

Dalle definizioni suddette segue che un diagramma di collaborazione non deve necessariamente includere un'interazione. In tale evenienza il diagramma mostra esclusivamente il contesto nel quale le interazioni possono aver luogo, e quindi, nel caso di un diagramma al livello di istanze, si ha un diagramma affine a uno ad oggetti (confrontare Capitolo 7, "I veri diagrammi degli oggetti").

Formalismo

I diagrammi di collaborazione sono costituiti da un grafo di nodi connessi da opportuni archi (*cfr* fig. 9.20). I nodi rappresentano istanze o ruoli classificatori (fig. 9.1) e sono rappresentati graficamente, rispettivamente, attraverso il rettangolo degli oggetti e delle classi, mentre gli archi rappresentano le connessioni tra questi elementi (*Link* o *AssociationRole*).

Figura 9.20 — Diagramma di collaborazione del pattern ServiceLocator dotato di meccanismo di cache. Si tratta della trasposizione del diagramma di fig. 9.5 (a cui si rimanda per la descrizione dettagliata). Come si può notare, le principali differenze sono le seguenti: nei diagrammi di collaborazione la sequenza dei messaggi è fondamentale in quanto rappresenta l'unico sistema per poter evidenziare l'ordinamento dei messaggi e la struttura fisica delle istanze è evidenziata esplicitamente.



Al fine di fornire un maggiore quantitativo di informazioni, oltre a questi elementi base è possibile specificarne altri, come diversi tipi di classificatori, generalizzazioni e vincoli. Gli elementi aggiuntivi, chiaramente, mantengono la propria rappresentazione grafica (tool di disegno permettendo).

Un ruolo classificatore è mostrato utilizzando lo stesso simbolo del rettangolo utilizzato per rappresentare le classi, in cui, tipicamente, è riportato esclusivamente il compartimento dedicato al nome. Nulla vieta in caso di necessità di riportare anche quelli degli attributi e delle operazioni (*cf* fig. 9.21).

Il comportamento dedicato al nome è utilizzato per specificare una stringa conforme alla seguente struttura:

```
[ '/' ClassifierRoleName ] [ ':' ClassifierName [ ',' ClassifierName ]* ]
```

Sia il nome del ruolo, sia quello del o dei classificatori sono opzionali. Chiaramente almeno uno dei due deve essere presente. Il nome del ruolo è sempre preceduto dal carat-

Figura 9.21 — Diagramma di collaborazione di livello di specifica. In questo caso gli elementi presenti sono i ruoli dei classificatori (ClassifierRole) e i ruoli associazione (AssociationRole).

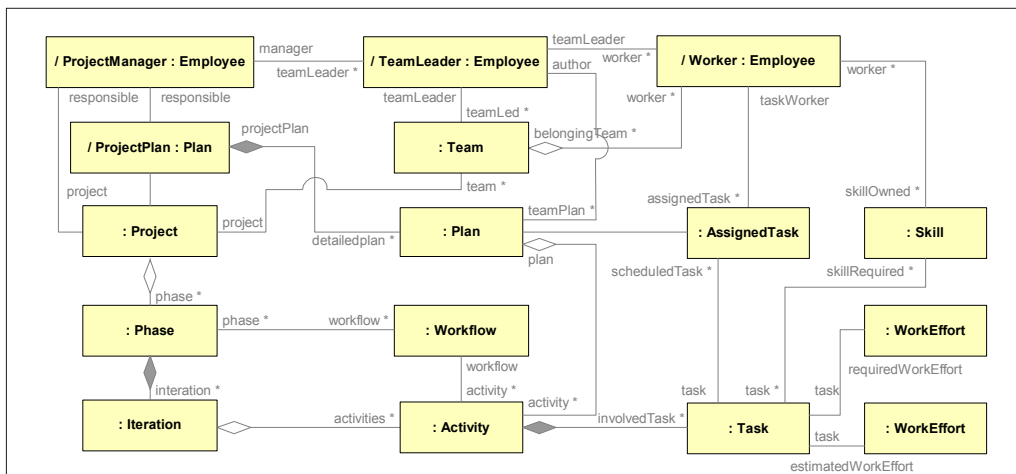


Figura 9.22 — Diagramma mostrante una porzione di una collaborazione munita di iterazione. Come si può notare non è necessario riportare, in ogni diagramma, tutte le entità, le relazioni, ecc. bensì la selezione degli elementi da visualizzare deve essere eseguita in funzione dell'obiettivo che si intende conseguire con la realizzazione del modello stesso.

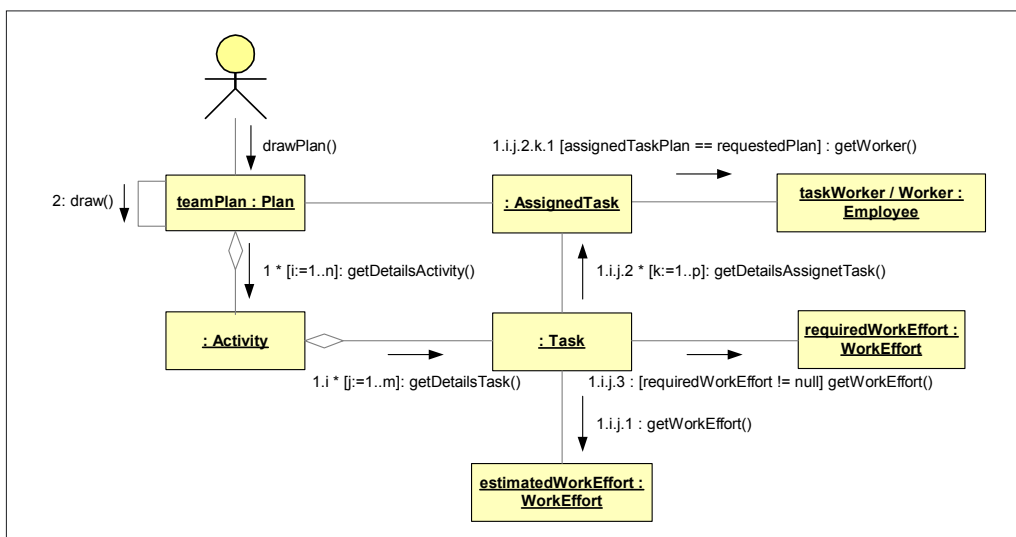


Tabella 9.2 — *Sommario della sintassi del nome del ruolo.*

SINTASSI	SPIEGAZIONE
/ R	Indica un ruolo denominato R.
: C	Il ruolo non viene fornito, mentre C è il classificatore base.
/ R : C	Indica un ruolo R di un classificatore C.

Tabella 9.3 — *Sommario della sintassi dei nomi degli oggetti.*

SINTASSI	SPIEGAZIONE
: C	Istanza il cui nome non è specificato di un classificatore C.
/ R	Istanza il cui nome non è specificato recitante il ruolo R.
/ R : C	Istanza il cui nome non è specificato di un classificatore C recitante il ruolo R.
O / R	Istanza di nome O recitante il ruolo R.
O : C	Istanza di nome O originata da un classificatore C.
O / R : C	Istanza di nome O originata da un classificatore C recitante il ruolo R.
O	Istanza di nome O.

tere slash (/), mentre quello del classificatore è introdotto dal carattere “due punti” (:) (tab. 9.2). Il nome del classificatore può essere fornito nella sua forma completa, ossia riportante il percorso costituito dall’elenco dei package che lo contengono (ricordando che nello UML il separatore dei vari elementi del percorso è il carattere due punti e non il singolo punto). L’eventuale sequenza dei classificatori (specificati attraverso il nome) rappresenta i nomi dei classificatori base.

Qualora il classificatore sia uno stereotipo, come da prassi UML, è possibile visualizzarlo ricorrendo alla relativa icona, e/o riportando il nome dello stereotipo stesso prima della stringa rappresentante il nome del classificatore. Il diagramma di fig. 9.23, per esempio, mostra un comportamento dinamico di un frammento di modello appartenente alla fase di analisi e pertanto le varie entità sono mostrate ricorrendo ai relativi stereotipi.

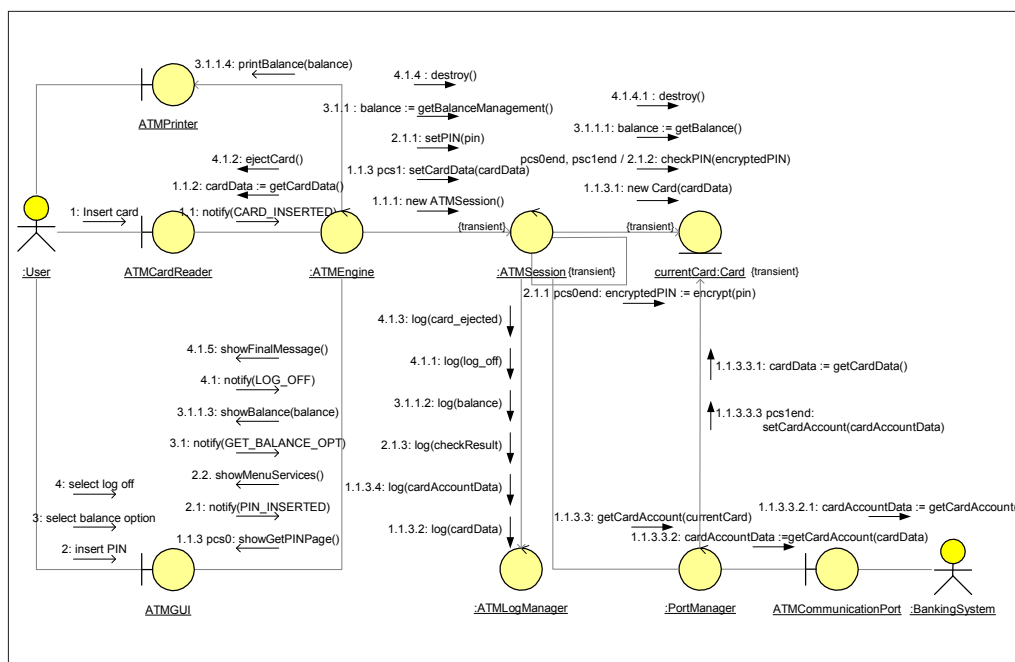
Nel caso in cui un ruolo classificatore rappresenti un insieme di istanze è possibile specificarne la molteplicità nell’angolo in alto a destra del rettangolo dedicato al nome.

Un’istanza che svolge il compito di un ruolo classificatore è mostrata graficamente attraverso il classico rettangolo degli oggetti. Il nome dell’istanza rispetta la seguente convenzione (fig. 9.21):

```
[ObjectName] [ '/' ClassifierRoleName ] [ ':' ClassifierName [ ',' ClassifierName ] * ]
```

Rispetto allo standard utilizzato per il ClassifierRole, le differenze sono la presenza del nome dell’oggetto e la sottolineatura dell’intera stringa (tab.9.3).

Figura 9.23 — Diagramma di collaborazione, al livello di analisi (si riconosce dall'utilizzo dei relativi stereotipi e dalla presenza di "macro" oggetti), per l'ottenimento del saldo del conto corrente associato alla carta Bancomat. La possibilità di posizionare gli elementi su tutta la superficie piana ha permesso di modellare, attraverso un solo diagramma, un servizio molto complesso, sebbene poi la leggibilità non ne tragga beneficio. In particolare si può notare che l'analisi dell'evoluzione dell'interazione richiede un certo impegno; non è quindi così immediato come nel caso dei diagrammi di sequenza. Ciò è dovuto alla mancanza di una dimensione esplicita dedicata al fattore tempo. Il diagramma è stato disegnato dando risalto alla struttura a strati dell'architettura.



Nel metamodello UML, il segmento che unisce due ruoli classificatori, è AssociationRole (ruolo associazione) mentre quello che connette istanze è Link (collegamento). Il nome di questi elementi segue le stesse regole riportate per il nome del ruolo classificatore (fig. 9.25).

I diagrammi di collaborazione si caratterizzano per conferire particolare enfasi all'aspetto strutturale della collaborazione, pertanto viene mostrata esplicitamente l'organizzazione strutturale (elementi e relazioni) delle varie entità cooperanti.

Nel contesto di un processo di sviluppo del software, il formalismo dei diagrammi di collaborazione è utilizzato principalmente per modellare procedure appartenenti al mo-

dello di analisi e disegno. Pertanto, non di rado, le relazioni esistenti tra le varie istanze sono corredate dai vincoli di navigabilità, mostrati graficamente per mezzo di apposita freccia posta a una estremità del segmento della relazione. Per esempio, nel diagramma di fig. 9.23, si può notare che un'istanza `ATMSession` può navigare, e quindi utilizzare l'interfaccia pubblica di, istanze `Card`, mentre non è vero il contrario.



Mostrare vincoli di navigabilità nei diagrammi di collaborazione, sebbene conforme alle direttive UML, raramente rappresenta una buona norma. La loro visualizzazione non aggiunge particolari chiarimenti — i vincoli possono essere desunti dall'analisi dell'interazione — e tende a creare confusione circa la direzione dei vari messaggi/stimoli. Pertanto è consigliabile relegare la visualizzazione dei vincoli di navigabilità ai diagrammi di struttura.

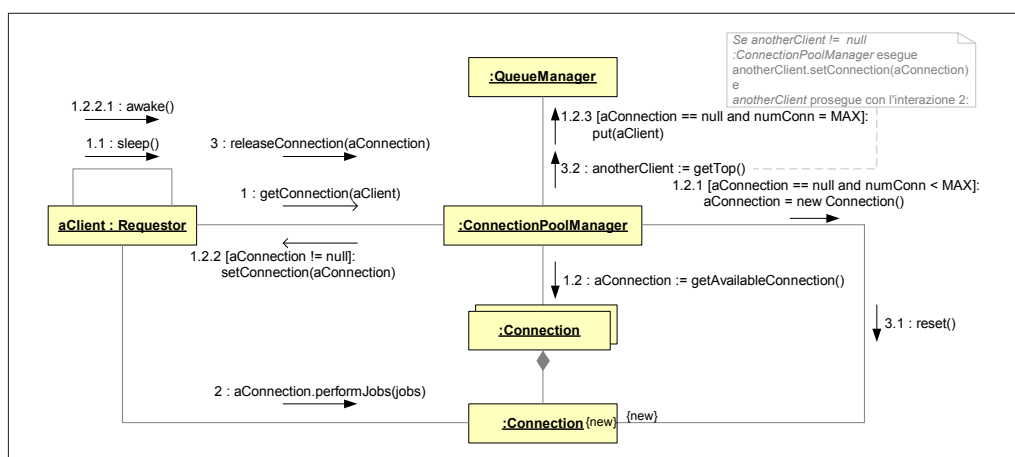
Come illustrato nel capitolo 7 (paragrafo “Navigabilità”), qualora in una relazione di associazione le istanze di una classe non possano “vedere” quelle dell'altra (essenzialmente invocarne i metodi), nella rappresentazione della relazione si inserisce una freccia indicante il verso di percorrenza. Pertanto gli oggetti istanza della classe posta nella coda dell'associazione (`navigabilità = false`) possono invocare i metodi ed accedere agli attributi pubblici delle istanze della classe puntata dalla freccia (`navigabilità = true`), mentre non è possibile il contrario.

La notazione dei diagrammi di collaborazione, al contrario di quella dei diagrammi di sequenza, non prevede un apposito asse dedicato al fattore tempo. Per sopperire a tale lacuna, l'ordinamento temporale dei messaggi è indicato da un'esplicita numerazione riportata nell'etichetta associata ai messaggi stessi (tipicamente, si inizia con il numero 1). Tale numerazione è anche utilizzata per evidenziare sottosequenze di messaggi associate a invocazioni annidate, per cicli e comportamenti alternativi. Per esempio, con riferimento alla fig. 9.22, lo stimolo 1 (`getDetailsActivity()`) viene eseguito n volte per reperire tutte le attività di cui è composto un piano a livello di singolo team. Al fine di evidenziare il comportamento iterativo, lo stimolo successivo (`getDetailsTask()`), necessario per reperire i dettagli di ciascun task, è etichettato con la sequenza $1.i$, dove i è l'indice della precedente iterazione.

Per via dell'impossibilità di ricorrere a convenzioni grafiche specifiche per mostrare il ciclo di vita degli elementi (cosa invece presente nei diagrammi di sequenza), il formalismo dei diagrammi di collaborazione prevede il seguente elenco di vincoli standard:

- `{new}` è utilizzato per mostrare istanze e relazioni create nel corso della interazione disegnata;

Figura 9.24 — Esempio di utilizzo del *multiobject* per rappresentare il comportamento dinamico di un *ConnectionPool*.



- `{destroyed}`, specularmente al caso precedente, è utilizzato per mostrare istanze e relazioni distrutte nel corso dell'interazione disegnata;
- `{transient}`, unione dei due casi precedenti, è utilizzato per mostrare istanze e relazioni create e distrutte nel corso della interazione disegnata.

Queste informazioni dovrebbero poter essere desunte dall'analisi dell'interazione che si instaura tra le varie istanze, tuttavia è molto comodo conferire loro particolare enfasi mostrandole direttamente nel diagramma di collaborazione.

Con riferimento al diagramma di fig. 9.23, si può notare che le istanze *ATMSession* e *Card* (e i relativi collegamenti, *link*), sono create e distrutte durante l'interazione visualizzata. Pertanto, al fine di evidenziare il relativo ciclo di vita, sono state corredate dai vincoli `{transient}`. Si tratta chiaramente di informazioni desumibili dall'analisi del diagramma stesso (sono presenti i vincoli `1.1.1: new ATMSession` e `1.1.3.1: new Card(cardData)`) che però conviene sempre evidenziare.

Alcuni autori non incoraggiano l'utilizzo del vincolo `{transient}`, consigliando invece il ricorso alla coppia `{new}` e `{destroyed}`. Si tratta di una scelta assolutamente legittima, sebbene discutibile.

Oggetti multipli

In una collaborazione è possibile mostrare oggetti multipli (*multiobject*). Ciascuno di essi rappresenta un insieme di istanze presenti alla terminazione di una relazione *molti* di un'as-

sociazione. La rappresentazione grafica si ottiene creando un “effetto ombra” come mostrato in fig. 9.24: un primo rettangolo è disegnato spostato leggermente sia lungo l’asse verticale, sia lungo quello orizzontale per dare l’idea di una pila.

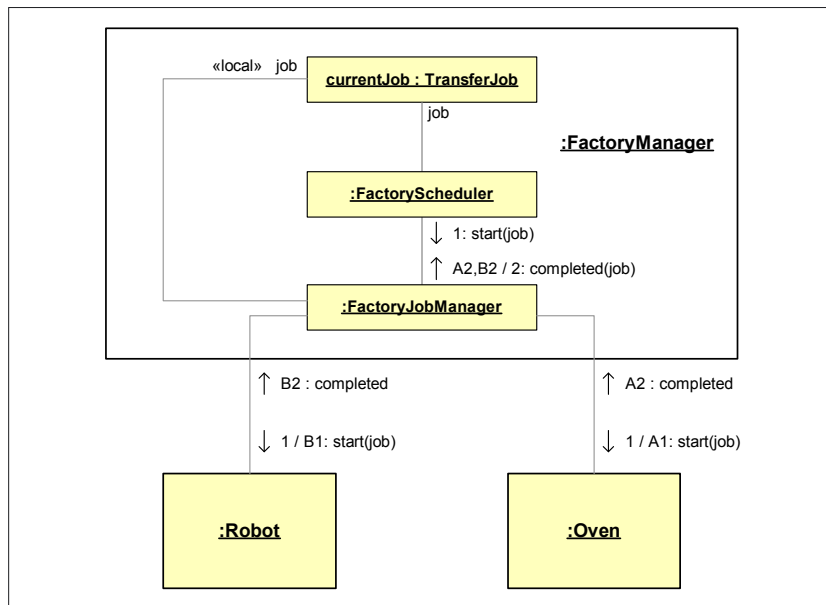
Una comunicazione con il *multiobject* rappresenta uno stimolo inviato all’insieme di istanze. L’esecuzione di una determinata operazione su tutte le istanze di un oggetto multiplo viene mostrata attraverso un pattern ben definito formato da due stimoli:

- il primo diretto al `multiobject` stesso al fine di ottenere il riferimento alle singole istanze;
- il secondo inviato a ciascuna di esse attraverso un collegamento temporaneo.

Questo pattern prevede un’alternativa: è possibile inglobare i due collegamenti in un uno solo, che mostra un’interazione e un’applicazione a ciascuna istanza. Importante è riportare l’indicatore di molti (*) nel nome del ruolo del destinatario, al fine di evidenziare che molti collegamenti individuali sono impliciti.

Ciascuna istanza dell’insieme è mostrata attraverso il canonico simbolo destinato agli oggetti. Tipicamente si preferisce associare le singole istanze al multioggetto di appartenenza attraverso una relazione di composizione (cfr Capitolo 7, “Aggregazione e composizione”).

Figura 9.25 — Esempio di oggetto attivo composto (fig. 3-71 specifiche UML 1.4).



Oggetto attivo

Un oggetto attivo (*active object*) è un'istanza che detiene un flusso di controllo (*thread of control*) e può dar luogo a un'attività di controllo. Viceversa, un oggetto passivo mantiene dei dati, ma non ha la facoltà di avviare un controllo. Ciò nonostante, può inviare stimoli e processare quelli ricevuti nell'ambito del processo in cui è inserito. In maniera corrispondente agli oggetti attivi è possibile definire i ruoli classificatori attivi, ossia classi attive le cui istanze sono appunto oggetti attivi.

La notazione grafica utilizzabile per evidenziare oggetti attivi prevede diverse alternative. In primo luogo è possibile disegnare la relativa icona (il rettangolo) utilizzando un bordo più spesso (stessa notazione dei diagrammi di sequenza). Eventualmente è anche possibile aggiungere l'apposita parola chiave {active} (in UML si tratta di un *tagged-value*, valore etichettato, il cui valore true viene sottointeso, *cfr* Capitolo 2, "Meccanismi di estensione"). Un'alternativa consiste nel rappresentare gli oggetti attivi come oggetti composti con all'interno le varie parti.

Come specificato nel capitolo dedicato ai diagrammi degli oggetti (in quel caso si trattava di stereotipi dell'elemento di fine associazione `AssociationEnd`), alla terminazione dei vari collegamenti (`Link`) è possibile associare opportuni stereotipi atti a specificare l'implementazione delle associazioni. In particolare gli stereotipi predefiniti sono illustrati nella tab. 9.4.

Sebbene riportare questi stereotipi concorra a rendere i diagrammi di collaborazione più accattivanti, si tratta di informazioni assolutamente deducibili dai diagrammi di struttura e quindi dovrebbe essere quello il luogo più naturale ove specificare informazioni di questo tipo.

Tabella 9.4 — *Stereotipi dell'elemento fine collegamento (`LinkEnd`). Da notare che le icone mostrate non sono standard, ma si tratta di quelle utilizzate da alcuni tool.*

Stereotipo	Descrizione	Spiegazioni
«association» A	associazione	Indica che il <i>Link</i> è ottenuto attraverso un'associazione vera e propria. Si tratta della situazione di default è pertanto inutile specificare questo stereotipo.
«parameter» P	parametro di un metodo.	Il riferimento all'oggetto è realizzato attraverso un parametro di un metodo.
«local» L	variabile locale di un metodo.	Il riferimento all'oggetto è ottenuto per mezzo di una variabile locale.
«global» G	variabile globale	Il riferimento all'oggetto di destinazione realizzato attraverso un valore globale e quindi conosciuto da tutti gli elementi.
«self» S	auto collegamento	Indica che l'istanza è visibile in quanto è la stessa che effettua la richiesta. Si tratta di uno stereotipo abbastanza inutile in quanto desumibile immediatamente dall'analisi visiva del diagramma.

Etichette dei messaggi e stimoli

L'etichetta specificabile ai collegamenti tra le entità di un diagramma di collaborazione permette di dichiarare il messaggio/stimolo spedito, gli eventuali argomenti compreso il valore di ritorno e la sequenza dei messaggi all'interno di una interazione più ampia, includendo chiamate annidate, iterazioni, suddivisioni del flusso, esecuzioni concorrenti e sincronizzazioni.

Più precisamente, la sintassi prevede la seguente struttura:

```
[predecessor] sequence-expression signature
```

Il campo *predecessor* è una lista di numeri, separati da virgola, conclusa dal carattere *slash* (/). Qualora la lista risulti vuota può essere omessa.

```
predecessor := sequence-number ',' ... '/'
```

Ogni elemento di questa lista rappresenta un'espressione di sequenza (*sequence-expression*) senza ripetizioni e deve necessariamente corrispondere al numero di sequenza di un altro messaggio. Questo meccanismo serve a indicare che un determinato messaggio o stimolo non può avvenire fintantoché non siano stati emessi tutti i messaggi/stimoli presenti nella lista *predecessor*. In poche parole, la lista rappresenta un meccanismo di sincronizzazione di flussi di controllo (thread). Per esempio, nella fig. 9.25, affinché lo stimolo 2 (A2, B2 / 2: completed(job)) possa aver luogo è necessario che siano stati ricevuti dall'istanza FactoryJobManager gli stimoli A2 e B2 (ossia della relativa sincronizzazione).

Tutti i messaggi (stimoli) corrispondenti a numeri di sequenza precedenti a quello analizzato ne sono implicitamente predecessori e quindi non necessitano di essere dichiarati nella lista dei predecessori del messaggio stesso.

L'espressione di sequenza (*sequence expression*) è una lista di termini di sequenza (*sequence terms*), separati dal carattere "punto" (.), terminata dal carattere "due punti" (:)

```
sequence-expression := sequence-term '.' ... ':'
```

Ogni *sequence term* rappresenta un livello di annidamento procedurale all'interno dell'intera interazione. La relativa sintassi prevede:

```
sequence-term := [integer | name] [recurrence]
```

L'intero rappresenta l'ordine sequenziale dei messaggi (stimoli) all'interno del livello superiore di invocazione procedurale. Per esempio un messaggio (stimolo) identificato dal

numero 3.1.4 segue quello identificato dal numero 3.1.3 ed entrambi hanno luogo all'interno dell'attivazione 3.1. Il nome rappresenta un controllo di flusso concorrente. Messaggi (stimoli) che differiscono per il nome sono concorrenti al relativo livello di annidamento. Per esempio i messaggi (stimoli) 3.1a e 3.1b avvengono in maniera concorrente all'interno dell'attivazione 3.1.

La *recurrence* rappresenta un'esecuzione o condizionale o iterativa. Le alternative sono:

```
'*' [||] '[' interaction-clause ']'
```

utilizzata per indicare un'iterazione;

```
'[' condition-clause ']'
```

utilizzata per indicare una ramificazione del flusso di controllo

Un'iterazione rappresenta una sequenza di messaggi o stimoli appartenenti allo stesso livello di annidamento. La clausola di iterazione (*iteration-clause*) può essere specificata in pseudocodice o utilizzando un opportuno linguaggio di programmazione. Un esempio molto semplice è

```
*[i := 1..n]
```

Per default viene assunto che le iterazioni abbiano luogo sequenzialmente. Qualora non sia questo il caso (quindi esecuzione parallela) la notazione prevede l'aggiunta delle doppie barrette.

Una condizione rappresenta un messaggio (o stimolo) la cui esecuzione è subordinata al verificarsi della condizione espressa nella clausola. Come nel caso precedente, tale condizione si presta a essere espressa o attraverso un appropriato pseudocodice oppure utilizzando un opportuno linguaggio di programmazione. Semplici esempi sono

```
[overdraft = 0]
```

```
[aConnection != null and numConnection < maxConnection]
```

La sintassi delle due versioni è molto simile (nel caso dell'iterazione è prevista la presenza del carattere asterisco ed eventualmente la doppia barretta) e la versione 1.4 dello UML non prescrive alcun formato per le clausole.

La firma (*signature*) è una stringa che specifica il nome, il possibile valore restituito e l'eventuale lista di parametri dell'operazione invocata o della ricezione, secondo la sintassi:

```
signature := [return-value ::= ] message-name [argument-list]
```

Il valore di ritorno rappresenta una lista di nomi indicanti i valori restituiti alla fine dell'esecuzione dell'elaborazione generata dalla comunicazione. È quindi possibile specificare il ritorno di più valori, particolarmente utile nei diagrammi al livello di specifica. Alcuni esempi sono

```
aConnection := getAvailableConnection()
```

```
login, name, surname, e-mail := getUserCredentials()
```

Il nome del messaggio corrisponde o a quello dell'operazione che l'entità ricevente deve eseguire all'atto del ricevimento della comunicazione, oppure al nome del segnale inviato al ricevitore.

La lista degli argomenti è un elenco separato da virgola racchiuso tra parentesi (queste possono essere omesse qualora l'elenco sia vuoto). Come è lecito attendersi si tratta dei parametri attuali da sostituire a quelli formali previsti dall'operazione invocata. La modalità per specificare questi argomenti non è definita in UML e quindi si possono utilizzare le medesime convenzioni utilizzate nei linguaggi di programmazione.

Invece che mostrare stringhe di testo per specificare gli eventuali valori di ritorno e lista di argomenti, è possibile ricorrere a una notazione grafica alternativa. Questa prevede di rappresentare tali elementi per mezzo di piccole circonferenze collegate ad apposite frecce con indicato il nome del parametro di ritorno o del parametro attuale.

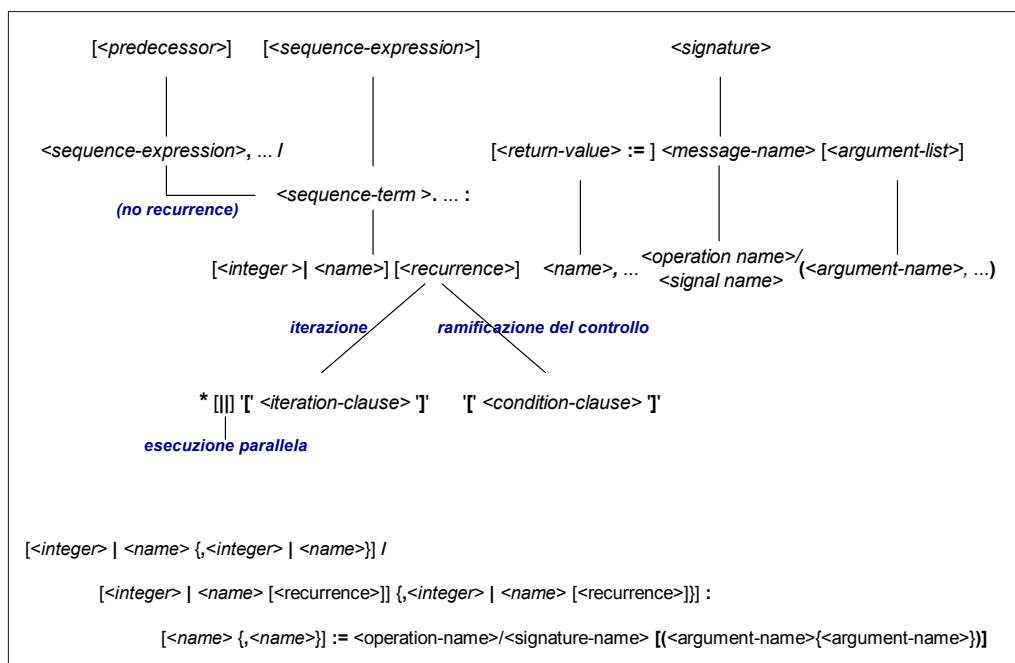


Sebbene la notazione grafica rappresenti un'alternativa molto accattivante, tende a generare diagrammi di collaborazione piuttosto confusi, specie in presenza di collaborazioni non semplici e di messaggi (stimoli) con un numero di parametri (tra quelli di ingresso e di uscita) superiore a qualche unità. Pertanto, a meno di casi semplici con interazioni limitate, si consiglia di ricorrere alle convenzioni classiche delle stringhe. Sebbene ben poco accattivanti e meno evidenti, forniscono il vantaggio di generare diagrammi più ordinati e più vicini alla relativa implementazione.

Come nel caso dei diagrammi di sequenza, il ritorno da un flusso può essere indicato esplicitamente attraverso una freccia tratteggiata.

Sebbene possibile, è sconsigliata la visualizzazione esplicita di questo flusso che finisce quasi sempre per rendere i diagrammi decisamente caotici.

Figura 9.26 — Diagramma riassuntivo illustrante la grammatica delle etichette associabili a messaggi e stimoli visualizzati nei diagrammi di collaborazione.



In riferimento a quanto riportato in fig. 9.26, la notazione utilizzata prevede che:

- gli elementi non appartenenti alla grammatica e quindi a disposizione dell'utente siano circondati da parentesi angolari e riportati in corsivo;
- gli elementi appartenenti alla grammatica siano mostrati in grassetto. Qualora fonte di possibile confusione, siano racchiusi tra apici singoli;
- gli elementi opzionali siano racchiusi da parentesi quadre;
- gli elementi la cui ricorrenza è del tipo *zero..molti* siano posti all'interno di parentesi graffe.

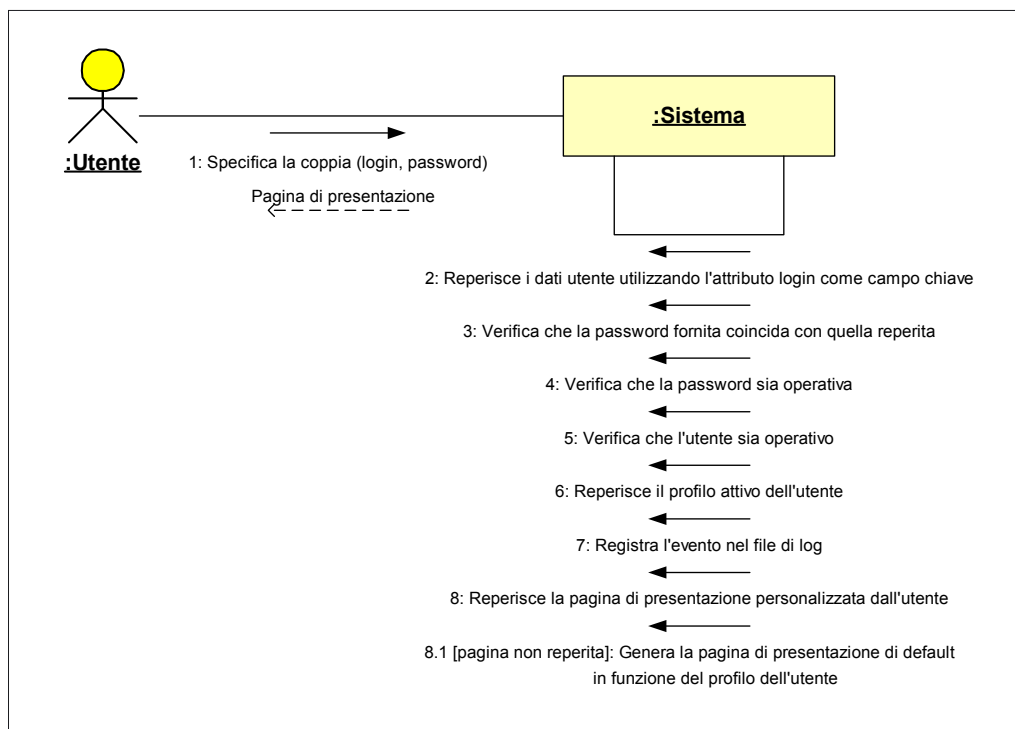
Utilizzo

I diagrammi di collaborazione, come è lecito attendersi, prevedono un utilizzo molto simile a quello dei diagrammi di sequenza. Più precisamente, nell'ambito dei processi di sviluppo del software, la relativa adozione è, per così dire, adiacente e in buona parte

sovrapposta a quello dei diagrammi di sequenza: mentre questi ultimi sono preferiti nelle fasi alte (analisi dei requisiti e analisi), quelli di collaborazione tendono ad essere privilegiati nelle fasi a maggiore contenuto tecnico (analisi e disegno).

Durante la fase di analisi dei requisiti, il ricorso ai diagrammi di collaborazione è abbastanza raro: il suo carattere tecnico, e la minore importanza conferita al fattore tempo, li rende meno fruibili a un pubblico non molto esperto come quello degli utenti/clienti. Inoltre, la caratteristica intrinseca dei diagrammi di comportamento dinamico prodotti nelle prime fasi del processo di sviluppo del SW (pochi “macro” elementi corredati da un gran numero di messaggi), ne rende proibitivo l’utilizzo (*cfr* fig. 9.27). Viceversa, nella fase

Figura 9.27 — *Esempio di utilizzo del formalismo dei diagrammi di collaborazione per modellare lo scenario di un caso d’uso. Si tratta della trasposizione del diagramma di fig. 9.2. La mancata rappresentazione esplicita della dimensione temporale, nonché la presenza di un gran numero di messaggi tra pochi elementi, rende la notazione dei diagrammi di collaborazione decisamente poco allettante rispetto a quella dei diagrammi di sequenza, nella fase di analisi dei requisiti. A poco poi serve la visualizzazione dell’organizzazione delle entità collaboranti, fin troppo scontata.*



di disegno, le caratteristiche che li rendono poco efficaci nelle fasi precedenti si trasformano in importanti vantaggi: è possibile evidenziare dettagli molto tecnici, mostrare collaborazioni tra un elevato numero di oggetti, e così via.

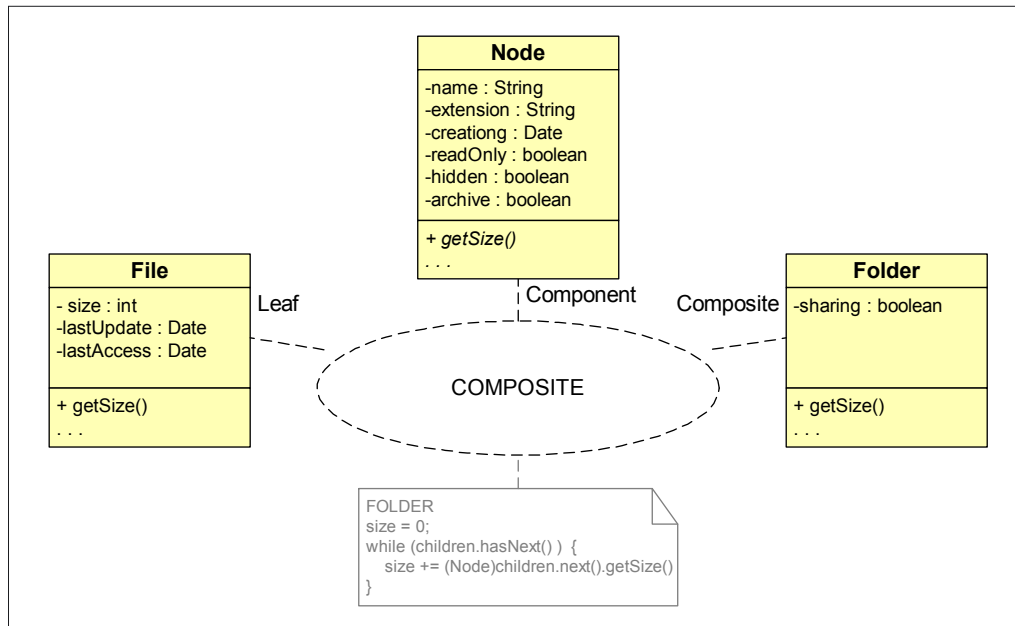
Di seguito è riportato l'elenco degli utilizzi che la notazione dei diagrammi di collaborazione condivide con quella dei diagrammi di sequenza (la famosa sezione sovrapposta). Per questo motivo è riportata esclusivamente una breve descrizione (per una trattazione più esaustiva si rimanda al paragrafo "Utilizzo dei diagrammi di sequenza").

- navigare formalmente i modelli di struttura statica;
- fornire ai programmatori la descrizione del comportamento dinamico da implementare. Tipicamente per questo utilizzo i diagrammi di collaborazione tendono ad essere preferiti a quelli di sequenza;
- realizzare un'ottima documentazione del sistema;
- verificare la complessità del sistema e la relativa qualità in generale;
- realizzare i modelli di analisi e disegno.

Gli utilizzi per i quali i diagrammi di collaborazione si dimostrano privilegiati sono:

- modellare l'implementazione logica di operazioni complesse, specie se coinvolgono un numero significativo di oggetti e comportamenti concorrenti. Potendo disporre gli elementi su tutta la superficie a disposizione è possibile dar luogo a un più razionale utilizzo dello spazio, che, in ultima analisi, si traduce nel realizzare diagrammi più facilmente leggibili e nel poter mostrare collaborazioni tra un maggiore numero di elementi. Inoltre, pur non prevedendo apposita notazione grafica, il formalismo dei diagrammi di collaborazione permette di evidenziare agevolmente operazioni da eseguire parallelamente grazie alla grammatica prevista dalle etichette di stimoli/messaggi (sebbene per questo utilizzo la notazione migliore resti quella dei diagrammi di attività).
- analizzare con maggior dettaglio la suddivisione del comportamento tra le varie classi per mezzo dell'esplorazione degli aspetti comportamentali del sistema (non a caso, per esempio, la metodologia CRC enfatizza la collaborazione tra le classi).
- realizzare un modello che permetta di fornire una migliore visione (sebbene non sempre di eccessivo dettaglio) della collezione degli oggetti collaboranti. Questo modello, tipicamente, risulta molto utile in sistemi *real-time*.

Figura 9.28 — Esempio di diagramma di collaborazione utilizzato per rappresentare la structure pattern. Questa applicazione del design pattern Composite è utilizzata per rappresentare un semplice file system.



Struttura pattern

Oltre agli utilizzi classici citati nel precedente paragrafo, il formalismo dei diagrammi di collaborazione ne prevede un altro abbastanza singolare, che consiste nella specifica dell'implementazione dei costrutti di disegno. Si tratta di un utilizzo particolarmente apprezzato per scopi espositivi/didattici ma che non sempre trova altrettanto successo nella pratica.

L'esempio classico è l'utilizzo del formalismo dei diagrammi di collaborazione per rappresentare la struttura dei design pattern (fig. 9.28). In questo caso si rappresenta una collaborazione parametrizzata (template). Ad ogni applicazione la struttura generale del pattern è specializzata per il particolare contesto di utilizzo. In sostanza si sostituiscono i classificatori generici con altri specifici. Il tutto è molto simile, sebbene con un diverso livello di astrazione, a quanto avviene durante l'invocazione di un metodo: i parametri formali sono sostituiti da quelli attuali.

Anche la notazione grafica utilizzata in questo contesto è abbastanza inconsueta: l'intera collaborazione — che dall'esterno può essere vista come un'unica entità — è rappresentata attraverso un'ellisse, disegnata con bordo tratteggiato e con il nome della collabora-

zione riportato all'interno. Dal perimetro dell'ellisse si fanno poi uscire tanti segmenti (sempre rappresentati con tratteggio) quanti sono i classificatori che prendono parte alla collaborazione. Ognuno di questi è etichettato con il ruolo del partecipante. Questo nomi sono importanti perché rappresentano i parametri che devono essere sostituiti da appositi classificatori per l'applicazione concreta della collaborazione parametrica.

Nel metamodello UML l'elemento `Collaboration` è un elemento generalizzabile (eredita dall'elemento `GeneralizableElement`), pertanto è possibile collegare tra loro opportune collaborazioni utilizzando la relazione di specializzazione.

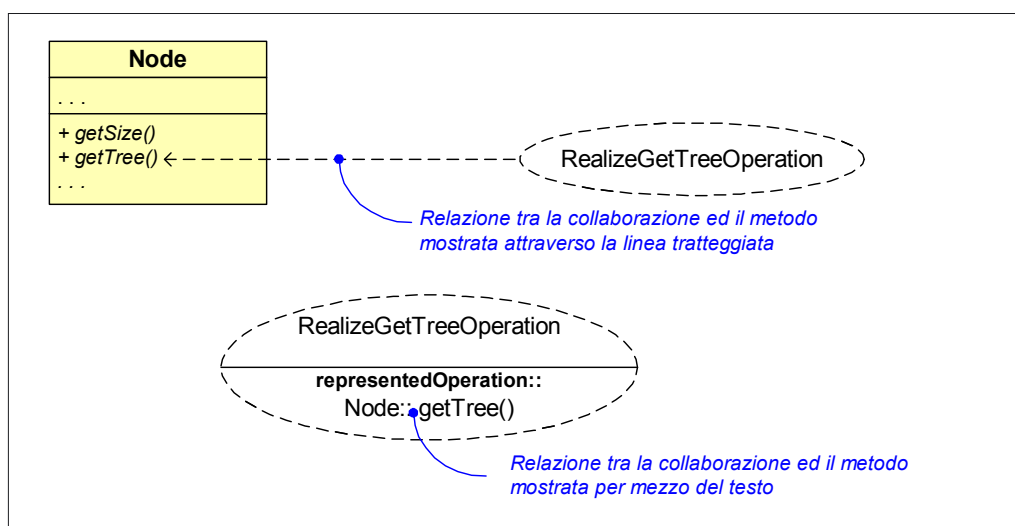
Una collaborazione utilizzata per realizzare un'operazione di un classificatore è rappresentata graficamente per mezzo di una linea tratteggiata con punta stilizzata che connette la collaborazione all'operazione implementata (fig. 9.29).

In alcuni casi è importante mostrare la struttura statica della collaborazione. Ciò è possibile sia riportando il dettaglio in un apposito diagramma, sia mostrando tale organizzazione direttamente all'interno dell'ellisse.

Qualora sia necessario definire una collaborazione in termini di un'altra, è possibile utilizzare due diverse notazioni:

- utilizzare l'ellisse tratteggiata mostrante la collaborazione e le relative relazioni;
- utilizzare la notazione classica dei diagrammi di collaborazione.

Figura 9.29 — Notazioni alternative per rappresentare una collaborazione atta a mostrare l'implementazione di un metodo.



La prima risulta più veloce e tende a enfatizzare i pattern applicati, mentre la seconda, più laboriosa, fornisce un maggiore livello di dettaglio e quindi un maggiore contenuto informativo.



Questo utilizzo dei diagrammi di collaborazione risulta utile per fini didattici/espositivi, mentre lo è molto meno nella progettazione reale. In teoria dovrebbe trattarsi di un ottimo meccanismo per produrre più rapidamente modelli di disegno al livello di specifica: ogni volta che si identifica un particolare costruito di disegno è possibile fornire semplicemente la specifica senza perdere troppo tempo nella fornitura dei dettagli, rimandando tale attività al modello di disegno di implementazione.

In pratica però, la maggior parte degli strumenti di disegno presenti sul mercato dispongono di un catalogo dei costrutti di disegno utilizzati più frequentemente. Quindi per la specifica del relativo dettaglio è sufficiente selezionare il pattern desiderato dal relativo repository e istruire il tool su come applicarlo, demandando al tool stesso la relativa applicazione. Ciò permette di ottenere direttamente la versione di implementazione del costruito senza dar luogo a inutili perdite di tempo.



Lo stile

Il presente paragrafo è dedicato all'illustrazione di una serie di linee guida che permettono di realizzare diagrammi di collaborazione di maggiore qualità e quindi più immediatamente e semplicemente comprensibili.

I criteri specificati per i diagrammi di sequenza che mantengono la loro validità anche nel dominio dei diagrammi di collaborazione, sono:

- evidenziare la struttura multistrato (*cf* fig. 9.23);
- utilizzare i nomi in modo consistente;
- selezionare un livello di dettaglio adeguato.

Mostrare la freccia per ogni messaggio

È buona pratica evidenziare ogni messaggio scambiato tra le varie entità per mezzo di apposita freccia, anche qualora le entità siano oggetto di un precedente scambio di messaggi. Ciò, oltre a rendere il diagramma più leggibile, permette di presentare una migliore visione della complessità sia dei singoli oggetti, sia della collaborazione generale. Qualora

un servizio richieda un eccessivo scambio di messaggi, potrebbe essere il caso di interrogarsi se le varie entità siano state correttamente disegnate (alcune potrebbero presentare un elevato accoppiamento) e se sia possibile rendere il tutto meno complesso, ricorrendo, al limite, a calcolate ridondanze

Evidenziare chiaramente i processi concorrenti

Qualora un comportamento dinamico si presti a essere modellato attraverso processi concorrenti è possibile e opportuno visualizzarli esplicitamente. Le direttive UML specificano di utilizzare la grammatica prevista per le etichette da associare agli stimoli/messaggi. In particolare è possibile evidenziare interazioni parallele assegnando ai messaggi/stimoli originati da thread concorrenti la medesima sequenza numerica e aggiungendo appositi nomi (per esempio T1, T2, ...) atti ad evidenziare i thread paralleli.

Per esempio in fig. 9.23 i due processi paralleli (atti, rispettivamente, a leggere il PIN digitato dall'utente e a ottenere i dati della carta dal server remoto), sono stati indicati, con i nomi `pcs0` e `pcs1`. In fig. 9.25, i processi paralleli sono stati indicati con i nomi A1 e B1.

Quale diagramma utilizzare?

L'interrogativo posto dal presente paragrafo dovrebbe risultare ormai superfluo, soprattutto in virtù della lettura dei paragrafi dedicati all'utilizzo dei diagrammi di sequenza e di collaborazione. Tuttavia si è ritenuto utile riportare qualche ulteriore chiarimento. Da tener presente che, giacché le due versioni di diagramma di interazione sono molto simili tra loro, in questo contesto, visualizzare un comportamento dinamico per mezzo della notazione meno conveniente tra le due non è poi cosa molto grave: i diagrammi di sequenza e di collaborazione illustrano le stesse informazioni evidenziando aspetti diversi. Inoltre, la maggioranza dei tool commerciali prevede apposite funzioni per passare da una rappresentazione all'altra. Ciò nonostante, è importante saper scegliere quale diagramma risulta più vantaggioso fondamentalmente per

- instaurare una comunicazione più idonea con il pubblico dei fruitori, al fine di ottenere il massimo riscontro;
- minimizzare la quantità di lavoro necessaria per la produzione e manutenzione degli stessi.

Nella fase di analisi dei requisiti non ci dovrebbero essere troppi problemi. La scelta del diagramma da utilizzare per mostrare opportune interazioni dovrebbe, quasi sempre, ricadere su quelli di sequenza: sono più semplici e immediati da comprendere specie da parte di un pubblico non esperto di modelli formali, quale per esempio quello degli utenti.

Meno frequentemente è comunque possibile ricorrere, anche in questa fase, ai diagrammi di collaborazione, specie qualora si vogliano mostrare interazioni tra diversi elementi. Molto importante è evitare sia un eccessivo livello di dettaglio, sia parte della notazione a elevato contenuto tecnico, particolarmente utile nelle fasi successive.

Come si vedrà meglio nel capitolo successivo, nella fase di analisi dei requisiti, un formalismo particolarmente apprezzato è quello dei diagrammi di attività.

Nell'ambito delle fasi di analisi e disegno, nella maggior parte dei casi, non vi sono enormi differenze. La regola empirica utilizzata spesso è legata al numero di elementi che prendono parte alla collaborazione e alla quantità di messaggi scambiati. Qualora il numero di elementi sia elevato — cosa non sempre gradita, ricordarsi la famosa regola dei sette elementi [BIB28] — è preferibile utilizzare il formalismo dei diagrammi di collaborazione. Permettono di posizionare gli elementi su tutta la superficie e quindi danno luogo a diagrammi più ordinati e leggibili. Qualora il numero di elementi non sia elevato e vi sia un consistente scambio di messaggi tra essi, allora è preferibile ricorrere al formalismo dei diagrammi di sequenza. Il ricorso alla notazione dei diagrammi di sequenza è inoltre privilegiato in quei casi in cui si voglia attribuire particolare enfasi all'organizzazione a strati dell'architettura in cui avviene l'interazione. Ancora, il ricorso ai diagrammi di collaborazione è preferibile per la realizzazione dei modelli di disegno poiché permettono di specificare più chiaramente dettagli a maggiore contenuto tecnico.

Un ultimo aspetto da valutare per la selezione del formalismo è la presenza di comportamenti concorrenti. La notazione dei diagrammi di sequenza non risulta particolarmente idonea per la modellazione di dinamiche di questo tipo. Pertanto la scelta ricade sui diagrammi di collaborazione. Anche questi non forniscono notazioni grafiche specifiche (come quelle dei diagrammi di attività), però permettono di sopperire a tale lacuna per mezzo della grammatica delle etichette dei messaggi e stimoli.

Ricapitolando...

Le parti che costituiscono un qualsivoglia sistema non sono assemblate per persistere semplicemente in una posizione stabile, bensì interagiscono tra loro scambiandosi messaggi al fine di raggiungere un preciso scopo. In ogni sistema ciascun componente svolge un determinato compito e interagisce con altri al fine di produrre un risultato globale, una funzionalità che va ben oltre la semplice somma dei risultati prodotti dai singoli oggetti. La progettazione di un sistema pertanto deve prevedere sia la proiezione statica, sia quella dinamica, le quali devono completarsi a vicenda; la descrizione di ogni comportamento deve essere modellata sia attraverso una prospettiva che mostri la struttura statica degli elementi cooperanti, sia per mezzo di un'altra che illustri il relativo comportamento dinamico.

Per quanto concerne la descrizione del comportamento dinamico, lo UML fornisce quattro notazioni: diagrammi di sequenza, collaborazione, attività e stato. Nel presente capitolo sono stati presentati i formalismi dei diagrammi di sequenza (*sequence diagrams*) e di quelli di collaborazione (*collaboration diagrams*). Si tratta delle due forme dei diagrammi di interazione (*interaction diagrams*). Quantunque

permettano di mostrare lo stesso contenuto informativo, si diversificano per l'aspetto dell'interazione cui è attribuita maggiore importanza. I diagrammi di sequenza enfatizzano lo scambio nel tempo dei messaggi di un insieme di oggetti cooperanti, mentre i diagrammi di collaborazione attribuiscono maggiore importanza alla struttura degli oggetti coinvolti nell'interazione modellata.

In OO, quando gli oggetti collaborano, si dice che si scambiano messaggi. Nella maggior parte dei casi, i messaggi consistono in semplici invocazioni di metodi. Altre volte possono consistere in invio di segnali e veri e propri scambi asincroni di messaggi (pubblicazione di messaggi su un apposito sistema di messaggistica).

Gli elementi base dei diagrammi di interazione, molto frequentemente, sono oggetti (nell'accezione propria OO) opportunamente collegati con evidenziate specifiche comunicazioni. Quantunque questo sia l'utilizzo più ricorrente, risulterebbe troppo limitativo relegare la notazione dei diagrammi di interazione alla modellazione di comportamenti dinamici relativi esclusivamente a istanze. Molto spesso è necessario realizzare diagrammi di interazione aventi come elementi primari diversi tipi di classificatori, come per esempio gli use case. Per questo motivo, il metamodello UML è stato arricchito introducendo nuovi elementi e prevedendo due versioni per ciascuno dei due diagrammi di interazione: una a livello di istanze (quella cui si è portati a pensare) e l'altra a un livello più astratto, di specifica. In quest'ultima versione l'elemento "ruolo classificatore" (`ClassifierRole`) possiede una funzione chiave. Definisce una proiezione ristretta del `Classificatore` cui è associato (precisamente `ClassifierRole` eredita da ed è associato all'elemento `Classificatore`), determinata dal comportamento richiesto dalla particolare collaborazione in cui è inserito. In questo modo è possibile specificare ruoli di `Classificatori` (si ricordi che `Classificatore` è l'elemento "genitore" di molti elementi dello UML quali `Classe`, `Interfaccia`, `Nodo`, `Componente`, `Attore`, `Caso d'Uso`, ecc.) nei diagrammi di interazione.

Logica conseguenza della presenza di due versioni di elementi base è la necessità di realizzare una doppia versione di tutti gli altri elementi (una a livello di istanze e l'altra a livello di specifica) al fine di poter definire le regole per la costruzioni dei relativi diagrammi.

Gli stimoli e i messaggi sono concetti base dei diagrammi di interazione e quindi condivisi sia dalla notazione dei diagrammi di sequenza, sia di quelli di collaborazione. In particolare uno stimolo è una comunicazione tra due istanze che trasporta informazioni con l'aspettativa che precise azioni abbiano luogo come conseguenza della sua ricezione. Un messaggio è la specificazione di uno stimolo: precisa i ruoli cui il mittente ed il destinatario si devono conformare e l'azione che, eseguita, genera lo stimolo che si conforma al messaggio.

In UML stimoli e messaggi sono rappresentati graficamente per mezzo di frecce con forme diverse in funzione della relativa semantica. In particolare:

- la freccia piena con punta piena indica un'invocazione di procedura o un flusso di controllo annidato. L'intero flusso annidato deve essere completamente concluso prima che la sequenza appartenente al livello esterno possa riprendere;
- la freccia piena con punta stilizzata rappresenta una comunicazione asincrona e quindi priva dell'annidamento del controllo;

- la freccia tratteggiata indica il ritorno dall'invocazione di una procedura.

Le comunicazioni sono corredate da opportune etichette. Nei diagrammi di sequenza, rappresentano il nome dell'operazione da invocare o il nome del segnale che le ha originate, mentre nei diagrammi di collaborazione indicano il messaggio inviato, gli eventuali argomenti e il valore di ritorno. Spesso l'etichetta è provvista del numero di sequenza per mostrare l'ordine del messaggio nell'interazione di appartenenza, considerando invocazioni annidate, iterazioni, divisioni del flusso, concorrenze e sincronizzazioni.

I diagrammi di sequenza illustrano interazioni tra specifici "oggetti" nei quali si conferisce particolare enfasi alla dimensione temporale. Il termine interazione indica un comportamento dinamico che include un insieme parzialmente ordinato di messaggi scambiati tra gli "oggetti" che prendono parte alla collaborazione al fine di raggiungere un risultato prestabilito. Pertanto un'interazione è definita nel contesto di una collaborazione: si specifica l'insieme degli elementi "collaboranti", quindi la sequenza dei messaggi scambiati. Sebbene questo costituisca l'utilizzo principale, i diagrammi di sequenza prevedono due versioni:

1. quella testé descritta, denominata *InteractionInstanceSet* (interazione di insieme di istanze che coinvolge *Stimoli* e *Istanze* di classificatori (nel metamodello, l'elemento *Instance* è associato ad almeno un elemento *Classificatore* che ne definisce le proprietà strutturali e comportamentali);
2. una meno intuitiva, definita in termini di *Interazione*, ossia scambi di *Messaggi* tra particolari entità modellate attraverso un apposito elemento denominato *ClassifierRole* (ruolo classificatore).

La rappresentazione grafica dei diagrammi di sequenza prevede due dimensioni: l'asse verticale che rappresenta il fattore tempo (per default aumenta verso il basso), l'asse orizzontale destinato a ospitare le varie *Istanze* e *ClassifierRole*. Tipicamente l'asse temporale non prevede alcuna gradazione. In specifici contesti tuttavia — per esempio la modellazione di sistemi real-time — può risultare molto utile disporre di un asse graduato al fine di evidenziare il tempo richiesto da ciascun servizio e il dettaglio di come tale valore sia ottenuto

Nei diagrammi di sequenza le entità partecipanti alla collaborazione sono mostrate attraverso il classico rettangolo (con notazione leggermente diversa a seconda del tipo dell'entità) con associato un segmento tratteggiato uscente dal rettangolo stesso e normale al suo lato inferiore, indicato con il nome di linea di vita (*lifeline*). Questo denota l'intervallo di tempo di esistenza della relativa istanza. Se una particolare istanza è creata e/o distrutta durante il periodo di tempo mostrato nel diagramma, allora la relativa linea di vita inizia e/o finisce negli appropriati punti, altrimenti è tracciata dall'inizio alla fine del diagramma. La creazione di un'istanza nel diagramma è mostrata attraverso una freccia la cui punta giunge direttamente al simbolo dell'oggetto, mentre la distruzione è mostrata attraverso un simbolo a forma di "X" posto alla fine della linea di vita. Un'istanza in vita prima dell'avvio del diagramma è mostrata riportando il relativo simbolo nella linea delle istanze in modo che preceda la prima freccia (avvio della transazione). Un'istanza

che resta in vita alla conclusione del diagramma è mostrata attraverso una linea di vita che continua dopo l'ultima freccia.

Nei diagrammi di sequenza è possibile illustrare comportamenti condizionali spezzando la linea di vita di un'istanza in due o più linee di vita concorrenti in cui ciascuna rappresenta uno specifico comportamento condizionale. Queste linee di vita possono eventualmente ricongiungersi in un punto conveniente. I luoghi in cui avviene la diramazione sono mostrati attraverso diverse frecce uscenti dallo stesso punto. Queste potrebbero consistere in esecuzioni parallele o semplicemente alternative. In questo ultimo caso, è conveniente mostrare nelle etichette associate alle frecce le condizioni che danno luogo alla relativa diramazione.

Il periodo di tempo durante il quale un'istanza esegue delle azioni è evidenziato per mezzo di un'attivazione (*focus of control*, localizzazione del controllo). Le attivazioni sono mostrate graficamente per mezzo di rettangoli sovrapposti alle linee di vita. Il lato in alto è allineato con l'istante di tempo di avvio dell'attivazione mentre quello in basso con l'istante di tempo di completamento.

Spesso accade che una precisa interazione sia ripetuta più volte, ossia che "l'emissione" di un preciso insieme di stimoli possa avvenire diverse volte. In tali circostanze è possibile raggruppare l'insieme delle frecce che danno luogo alla "sottoiterazione" ed evidenziarle (attraverso un opportuno rettangolo) come iterazione. La condizione dell'iterazione è riportata alla fine della stessa.

Nell'ambito dei diagrammi di sequenza gli stimoli e i messaggi sono rappresentati per mezzo di frecce, tipicamente orizzontali, aventi coda nel rettangolo che mostra l'attivazione dell'istanza mittente e testa nell'attivazione di quella ricevente. Qualora si voglia evidenziare un messaggio spedito da un'istanza a sé stessa, la freccia parte e arriva all'attivazione della stessa istanza.

Le frecce sono in genere rappresentate orizzontalmente. Ciò indica sia che l'intervallo di tempo necessario per la trasmissione dello stimolo è del tutto trascurabile comparato con la durata dell'attivazione, sia che l'evento è atomico, ossia che, nell'ambito dello scenario, durante l'intervallo temporale in cui avviene la trasmissione, null'altro può accadere. Nei contesti in cui queste ipotesi non sono valide (per esempio ambienti real-time), messaggi, o stimoli, sono illustrati attraverso una freccia inclinata verso il basso.

Il formalismo dei diagrammi di sequenza, nel contesto dei processi di sviluppo del software, si presta ad essere impiegato nelle seguenti fasi.

Analisi dei requisiti. In questo stadio possono essere adottati per illustrare graficamente il comportamento dinamico dei casi d'uso; più precisamente dei corrispondenti *scenario*. Data la relativa semplicità, si prestano a essere facilmente compresi anche da un pubblico con limitate conoscenze informatiche. In questa fase i principali benefici derivanti dall'utilizzo dei diagrammi di sequenza sono essenzialmente legati all'adozione di un formalismo grafico: risultano più accattivanti di uno "spento" testo, aumentano il livello di chiarezza e immediatezza, accrescono la capacità di comprensione e memorizzazione di quanto illustrato, agevolano il dialogo con l'utente, ecc. Gli svantaggi sono legati essenzialmente all'elevata quantità di tempo necessaria per produrre e mantenere aggiornati i vari diagrammi.

Analisi e disegno. In queste fasi è possibile avvalersi del formalismo dei diagrammi di sequenza per molteplici motivi:

- navigare formalmente i modelli di struttura statica, definiti attraverso appositi diagrammi come quello delle classi, per verificare che siano effettivamente in grado di erogare i servizi richiesti dal sistema;

- effettuare una prima verifica della qualità del modello. Comportamenti dinamici troppo complessi raramente scaturiscono da architetture ben disegnate;
- fornire ai programmatori la descrizione del comportamento dinamico da implementare;
- realizzare un'ottima documentazione del sistema;
- agevolare il processo di individuazione dei “colli di bottiglia”;
- mettere a punto una tecnica per la realizzazione di modelli di analisi e disegno di struttura statica.

Per quanto attiene allo stile, i suggerimenti da seguire sono:

- nei limiti del possibile organizzare i diagrammi di sequenza affinché emerga una sola direzione del flusso dei messaggi, da sinistra verso destra e dall'alto verso il basso;
- organizzare i diagrammi di sequenza in modo tale da enfatizzare la struttura multistrato dell'architettura;
- utilizzare in maniera consistente gli elementi visualizzati, prelevandoli dagli appositi diagrammi di struttura;
- mostrare il “giusto” livello di dettaglio;
- qualora i diagrammi di sequenza risultino particolarmente complessi, è possibile renderli maggiormente fruibili inserendo opportuni commenti;
- demandare, per quanto possibile, la descrizione di organizzazioni polimorfiche ai diagrammi di struttura statica;
- utilizzare i colori in modo appropriato.

I diagrammi di collaborazione mostrano grafici formati da entità interconnesse (tipicamente oggetti) eventualmente corredati con la rappresentazione di comportamento dinamico. Tale comportamento include un insieme parzialmente ordinato di messaggi scambiati tra le entità cooperanti al fine di raggiungere un determinato risultato come, per esempio, la fornitura di un servizio all'utente, l'elaborazione di un algoritmo, e così via. Il formalismo dei diagrammi di collaborazione prevede due aspetti: la struttura statica delle entità partecipanti (detta *collaborazione*) e, eventualmente, la descrizione della relativa rete di comunicazioni (*interazione*). In particolare, la realizzazione di diagrammi di collaborazione prevede, più o meno implicitamente, la seguente sequenza di attività:

1. definizione della collaborazione
2. modellazione dell'iterazione
3. specificazione delle etichette dei messaggi/stimoli.

In UML, la definizione formale di diagramma di collaborazione prevede due versioni:

- la prima sancisce che un diagramma di collaborazione rappresenta una collaborazione (*Collaboration*) con eventualmente specificata una interazione;
- la seconda afferma che un diagramma di collaborazione rappresenta un `CollaborationInstanceSet` (insieme di istanze collaboranti) con eventualmente specificato un `InteractionInstanceSet` (insieme di istanze interagenti).

La seconda definizione (detta al *livello di istanze*) è quella cui si è più comunemente portati a pensare: un diagramma di collaborazione rappresenta un grafo di oggetti con specificati eventualmente gli stimoli scambiati tra questi, necessari per realizzare il servizio.

Pertanto, i diagrammi di collaborazione sono costituiti da un grafo di nodi connessi da opportuni archi, in cui i nodi rappresentano istanze o ruoli classificatori (rappresentati graficamente, rispettivamente, attraverso il rettangolo degli oggetti e delle classi), mentre gli archi rappresentano le connessioni tra questi elementi (`Link` o `AssociationRole`).

Un ruolo classificatore è mostrato utilizzando lo stesso simbolo del rettangolo utilizzato per le classi, in cui è specificata una stringa secondo il formato:

```
[ '/' ClassifierRoleName ] [ ':' ClassifierName [ ',' ClassifierName ] * ]
```

Un'istanza che recita il ruolo di un ruolo classificatore è mostrata graficamente attraverso il classico rettangolo degli oggetti. In questo caso, il nome rispetta la seguente convenzione:

```
[ObjectName] [ '/' ClassifierRoleName ] [ ':' ClassifierName [ ',' ClassifierName ] * ]
```

La notazione dei diagrammi di collaborazione non prevede un apposito asse dedicato al fattore tempo. Per sopperire a tale lacuna, l'ordinamento temporale dei messaggi è mostrato per mezzo di un'esplicita numerazione riportata nell'etichetta associata ai messaggi stessi. Tale numerazione permette inoltre di evidenziare sottosequenze di messaggi associate a invocazioni annidate, per cicli e comportamenti alternativi.

Per via dell'impossibilità di ricorrere a convenzioni grafiche specifiche per mostrare il ciclo di vita degli elementi mostrati in un diagramma di collaborazione, il relativo formalismo prevede i seguenti vincoli standard:

- {new} utilizzato per mostrare istanze e relazioni create nel corso dell'interazione disegnata;
- {destroyed} atto a mostrare istanze e relazioni distrutte nel corso dell'interazione disegnata;
- {transient} utilizzato per mostrare istanze e relazioni create e distrutte nel corso della stessa interazione;

In una collaborazione è possibile mostrare oggetti multipli. Ciascuno di essi (*multiobject*) rappresenta un insieme di istanze presenti alla terminazione di una relazione *molti* di un'associazione. La rappresentazione grafica si ottiene creando un effetto "pila".

Una comunicazione con il multiobject rappresenta uno stimolo inviato all'insieme di istanze. L'esecuzione di una determinata operazione su tutte le istanze di un oggetto multiplo viene mostrata attraverso un pattern standard basato su due stimoli: il primo diretto al multiobject stesso al fine di ottenere il riferimento alle singole istanze, il secondo inviato a ciascuna di esse attraverso un collegamento temporaneo. Questa strategia prevede un'alternativa: inglobare i due collegamenti in un uno solo che mostra un'interazione e un'applicazione a ciascuna istanza. In questo caso è importante riportare l'indicatore di molti (*) nel nome del ruolo del destinatario.

Un oggetto attivo (*active object*) è un'istanza che detiene un flusso di controllo (*thread of control*) e quindi può dar luogo a un'attività di controllo. Viceversa, un oggetto passivo tipicamente mantiene dei dati, ma non ha facoltà di avviare un controllo. Può comunque inviare stimoli e processare quelli ricevuti. In maniera corrispondente agli oggetti attivi è possibile definire i ruoli classificatori attivi, ossia classi attive le cui istanze sono appunto oggetti attivi.

La notazione grafica utilizzabile per evidenziare oggetti attivi contempla diverse alternative. La prima prevede di disegnare il classico rettangolo utilizzando un bordo più spesso con eventualmente riportata l'apposita parola chiave {active} (valore etichettato). La seconda consiste nel rappresentare gli oggetti attivi come oggetti composti con all'interno le varie parti.

L'etichetta specificabile nei collegamenti tra le entità di un diagramma di collaborazione permette di dichiarare il messaggio/stimolo spedito, gli eventuali argomenti, compreso il valore di ritorno e la sequenza dei messaggi all'interno di una interazione più ampia, includendo chiamate annidate, iterazioni, suddivisioni del flusso, esecuzioni concorrenti e sincronizzazioni.

La sintassi prevede la seguente struttura:

```
[predecessor] sequence-expression signature
```

Il campo `predecessor` è una lista di numeri, separati da virgola, conclusa dal carattere slash (/), omessa, qualora vuota. Ogni elemento di questa lista rappresenta un'espressione di sequenza (*sequence expression*) senza ripetizioni e deve necessariamente corrispondere al numero di sequenza di un altro messaggio. Questo meccanismo serve a indicare che un determinato messaggio o stimolo non può aver luogo fintantoché non siano stati emessi tutti i messaggi/stimoli presenti nella lista `predecessor`.

L'espressione di sequenza (*sequence expression*) è una lista di termini di sequenza (*sequence term*), separati dal carattere "punto", seguita dal carattere "due punti":

```
sequence-term '.' ... ':'
```

Ogni *sequence term* rappresenta un livello di annidamento procedurale all'interno dell'intera interazione. La sintassi prevede:

```
[integer | name] [recurrence]
```

L'intero rappresenta l'ordine sequenziale dei messaggi (stimoli) all'interno del livello superiore di invocazione procedurale. Il nome rappresenta un controllo di flusso concorrente. Messaggi (stimoli) che differiscono per il nome sono concorrenti al relativo livello di annidamento.

La *recurrence* rappresenta un'esecuzione o condizionale o iterativa. Le alternative sono:

```
'*' [||] [' interaction-clause ']
```

che indica un'iterazione e

```
[' condition-clause ']
```

che indica una ramificazione del flusso di controllo.

Un'interazione rappresenta una sequenza di messaggi o stimoli appartenenti allo stesso livello di annidamento. La clausola di interazione può essere specificata in pseudocodice o utilizzando un opportuno linguaggio di programmazione. Per default viene assunto che le iterazioni abbiano luogo sequenzialmente. Qualora le iterazioni avvengano in parallelo, la notazione prevede l'aggiunta delle doppie barrette.

Una condizione rappresenta un messaggio (o stimolo) la cui esecuzione è subordinata al verificarsi della condizione espressa nella clausola. Tale condizione si presta a essere espressa o attraverso un pseudocodice appropriato oppure utilizzando un opportuno linguaggio di programmazione.

La firma è una stringa che specifica il nome, gli eventuali valori restituiti e la possibile lista di parametri dell'operazione invocata o della ricezione, secondo la sintassi:

```
[return-value := ] message-name [argument-list]
```

Il valore di ritorno rappresenta una lista di nomi indicanti i valori restituiti alla fine dell'esecuzione dell'elaborazione generata dalla comunicazione. Il nome del messaggio corrisponde o a quello dell'operazione che l'entità ricevente deve eseguire all'atto del ricevimento della comunicazione, oppure al nome del segnale inviato al ricevitore. La lista degli argomenti è un elenco separato da virgola racchiuso tra parentesi (omesse qualora l'elenco sia vuoto). Si tratta dei parametri attuali da sostituire a quelli formali previsti dall'operazione invocata.

Invece di mostrare stringhe di testo per specificare gli eventuali valori di ritorno e lista di argomenti, è possibile ricorrere a una notazione grafica alternativa. Questa prevede di rappresentare tali elementi

per mezzo di piccole circonferenze collegate ad apposite frecce con indicato il nome del parametro di ritorno o del parametro attuale.

Come nel caso dei diagrammi di sequenza, il ritorno da un flusso può essere indicato esplicitamente attraverso una freccia tratteggiata.

I diagrammi di collaborazione prevedono un utilizzo molto simile a quello dei diagrammi di sequenza. Sebbene tendano a essere preferiti nelle fasi a maggiore contenuto tecnico (analisi e disegno), durante la fase di analisi dei requisiti, il ricorso ai diagrammi di collaborazione è abbastanza raro: il relativo carattere tecnico e la sfumata importanza conferita al fattore tempo li rende meno fruibili a un pubblico non molto esperto come quello degli utenti/clienti. Inoltre, la tipica caratteristica dei diagrammi del comportamento dinamico prodotti nelle prime fasi del processo di sviluppo del SW (pochi “macro” elementi corredati da un gran numero di messaggi), ne rende l'utilizzo proibitivo. Viceversa, nella fase di disegno, le caratteristiche che li rendono poco appetibili nelle fasi precedenti si trasformano in importanti vantaggi: è possibile evidenziare dettagli squisitamente tecnici, mostrare collaborazioni tra un elevato numero di oggetti, comportamenti concorrenti, e così via.

Gli utilizzi che la notazione dei diagrammi di collaborazione condividono con quella dei diagrammi di sequenza sono: navigare formalmente i modelli di struttura statica, fornire ai programmatori la descrizione del comportamento dinamico da implementare, realizzare un'ottima documentazione del sistema, verificare la complessità del sistema e la relativa qualità in generale e realizzare i modelli di analisi e disegno.

Gli utilizzi per i quali i diagrammi di collaborazione si dimostrano privilegiati sono: modellare l'implementazione logica di operazioni complesse, specie se coinvolgenti un numero significativo di oggetti e comportamenti concorrenti, analizzare con maggior dettaglio la suddivisione del comportamento tra le varie classi per mezzo dell'esplorazione degli aspetti comportamentali del sistema, realizzare un modello che permetta di fornire una migliore visione della collezione degli oggetti collaboranti.

Oltre agli utilizzi classici citati nel precedente paragrafo, il formalismo dei diagrammi di collaborazione si presta a specificare l'implementazione di costrutti di disegno, come per esempio la struttura di design pattern. In questo caso si rappresenta una collaborazione parametrizzata da specializzare a ogni applicazione specifica. La notazione grafica utilizzata prevede che l'intera collaborazione sia rappresentata attraverso un'ellisse disegnata con bordo tratteggiato e con il nome della collaborazione. riportato all'interno Dal perimetro dell'ellisse si fanno poi uscire tanti segmenti tratteggiati quanti sono i classificatori che prendono parte alla collaborazione, etichettati con il ruolo del partecipante.

I criteri da seguire al fine di realizzare diagrammi di collaborazione di maggiore qualità, sono:

- evidenziare la struttura multistrato;
- utilizzare i nomi modo consistente;
- selezionare l'adeguato livello di dettaglio;
- mostrare la freccia per ogni messaggio;

- evidenziare chiaramente i processi concorrenti.

Poiché i diagrammi di sequenza e collaborazione sono molto simili tra loro (illustrano le stesse informazioni evidenziando aspetti diversi), la scelta su quali utilizzare non dovrebbe creare troppi problemi. Ciò nonostante, è importante saper scegliere quale diagramma risulta più vantaggioso per instaurare una comunicazione più efficace con il pubblico dei fruitori, per minimizzare la quantità di lavoro necessaria per la produzione e manutenzione degli stessi, e così via.

Nella fase di analisi dei requisiti non ci dovrebbero essere troppi problemi. La scelta del diagramma da utilizzare per mostrare opportune interazioni dovrebbe, quasi sempre, ricadere su quelli di sequenza.

Nell'ambito delle fasi di analisi e disegno, nella maggior parte dei casi non vi sono enormi differenze. La regola empirica spesso utilizzata è legata al numero di elementi che prendono parte alla collaborazione e alla quantità di messaggi scambiati. Qualora il numero di elementi sia elevato, è preferibile utilizzare il formalismo dei diagrammi di collaborazione. Mentre nel caso in cui il numero di elementi non sia elevato e vi sia un consistente scambio di messaggi tra essi, allora è preferibile ricorrere al formalismo dei diagrammi di sequenza. Il ricorso alla notazione dei diagrammi di sequenza è in oltre privilegiato in quei casi in cui si voglia attribuire particolare enfasi all'organizzazione a strati dell'architettura in cui avviene l'interazione. Il ricorso ai diagrammi di collaborazione, tipicamente, è preferibile per la realizzazione dei modelli di disegno poiché tali diagrammi permettono di specificare più chiaramente dettagli a maggiore contenuto tecnico. Infine, un ultimo aspetto da valutare per la selezione del formalismo è la presenza di comportamenti concorrenti. In questi casi è preferibile ricorrere ai diagrammi di collaborazione. Sebbene non forniscano notazioni grafiche specifiche, permettono di sopperire a tale lacuna per mezzo della grammatica delle etichette di messaggi e stimoli.