Università degli Studi di Parma
Dipartimento di Matematica e Informatica
Corso di Laurea in Informatica

DISPENSE INTRODUTTIVE

INTRODUZIONE ALLA GESTIONE DEL PROGETTO SOFTWARE CON UML

Prof. Giulio Destri

URL originale: http://www.giuliodestri.it/doc/D03_IntroGestioneProgettoSW.pdf

Indice

Affrontare un problema nel mondo reale

Scomposizione di problemi e compiti

La maggior parte dei problemi del mondo reale sono complessi e non possono essere affrontati direttamente “in toto”, ma attraverso una metodologia di scomposizione, che li trasforma in una successione di problemi via via sempre più semplici, finché non diventano affrontabili.

I passaggi per arrivare alla scomposizione possono essere riconducibili a schemi più o meno standardizzati, che comunque sono trasformabili in un modello comune.

Definiamo anche alcuni concetti basilari.

Si definisce **azione elementare atomica**, o semplicemente **azione**, un singolo atto che non può essere ulteriormente scomponibile. Un esempio di azione elementare può essere considerata la pressione di un singolo tasto sulla tastiera di un PC.

Si definisce **attività** un insieme ordinato di azioni che ha uno scopo. Una attività è scomponibile sempre in azioni, anche se in molti casi la scomposizione non è necessaria in relazione al problema. Un esempio di attività è la vendita di un chilo di mele entro un negozio ortofrutticolo.

Un esempio di attività di cui non interessa normalmente la scomposizione è il click col mouse su un bottone. Se, infatti, l’attività venisse scomposta, si troverebbe che essa è la successione delle seguenti azioni

- Afferra il mouse con la mano;
- Sposta il mouse finché il suo cursore non sia sopra il bottone da premere
- Clicca col tasto sinistro del mouse, tenendolo premuto il tempo giusto

Normalmente un’attività così semplice può essere considerata come un’azione elementare senza perdere particolari interessanti.

Successione di azioni ed attività

Per svolgere un compito complesso i passaggi possono essere i seguenti.

1. Identificare chiaramente i limiti del compito o del problema da affrontare: occorre capire con precisione fino a dove ci si deve spingere;
2. Identificare le risorse che si hanno a disposizione e quali risorse ulteriori potrebbero essere necessarie;
3. Identificare con precisione i vincoli che devono essere rispettati: ad esempio la quantità di denaro a disposizione per svolgere il compito, il tempo massimo entro cui il compito deve essere completato ecc…;
4. Formalizzare, ossia definire con chiarezza ed un linguaggio appropriato le attività che devono essere svolte per completare il compito da svolgere identificato al punto 1 e in che ordine, utilizzando le risorse definite al punto 2 e rispettando i vincoli definiti al punto 3;
5. Se il compito è molto complesso ed anche le attività sue componenti, scomporle ulteriormente in attività più semplici e/o in azioni atomiche, ossia azioni elementari non ulteriormente scomponibili;
6. Porre in atto, nella sequenza giusta, le azioni identificate nei punti precedenti, verificando che il risultato di ognuna di esse sia quello attivo;
7. Completata la sequenza principale verificare il risultato così ottenuto;
8. Ripetere eventualmente alcuni dei passaggi precedenti

9. Completare il lavoro con i passaggi mancanti e verificare gli output finale del lavoro stesso

Nel momento in cui non si è da soli, ma si deve coordinare il lavoro di più persone, diviene necessario un passaggio ulteriore.

Si definisce **WBS** (Work Breakdown Structure) o **Scomposizione Strutturata del Lavoro** la scomposizione dell'attività da compiere in tante attività più piccole o direttamente in azioni (si veda [PMBoK 2008]).

Il lavoro deve essere quindi scomposto in azioni ed attività più piccole. Anche le persone a disposizione per il lavoro devono essere classificate in base alle loro capacità e velocità di esecuzione delle attività e lo stesso deve avvenire per le risorse a disposizione.

La pianificazione prevede quindi che alle persone vengano assegnate attività da eseguire entro un tempo prefissato usando opportune risorse. Tali attività diventano quindi **incarichi**.

Per esempio, dovendo dipingere un insieme di stanze e disponendo di 4 pittori, a ciascuno saranno assegnati un secchio di vernice, dei pennelli e dei solventi (risorse) e una parte delle stanze da dipingere (attività da svolgere) entro una giornata (tempo limite prefissato).

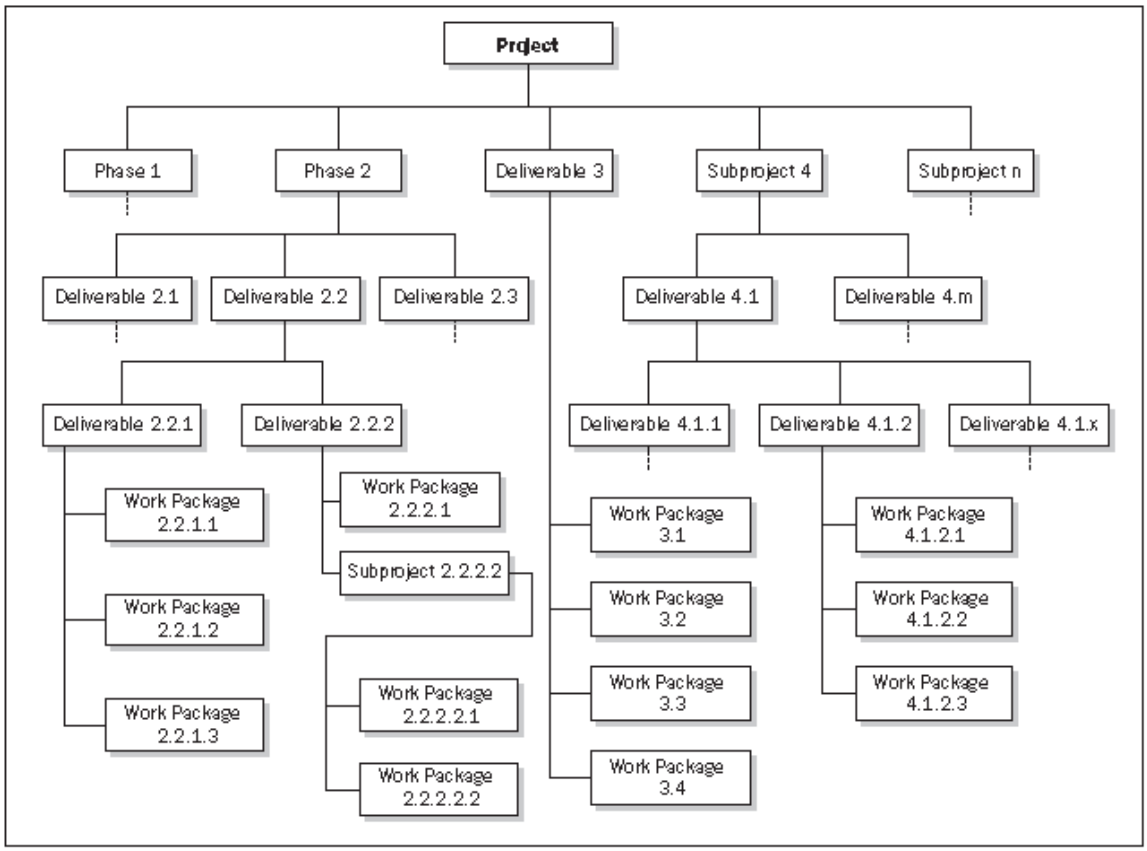


Fig. 1.1: la struttura di una WBS generica, tratta da [PMBoK 2008]. Il lavoro corrispondente a tutto il progetto viene suddiviso in fasi, ciascuna delle quali ha uno o più prodotti (deliverable) per costruire ciascuno dei quali serve un insieme di unità di lavoro (work package).

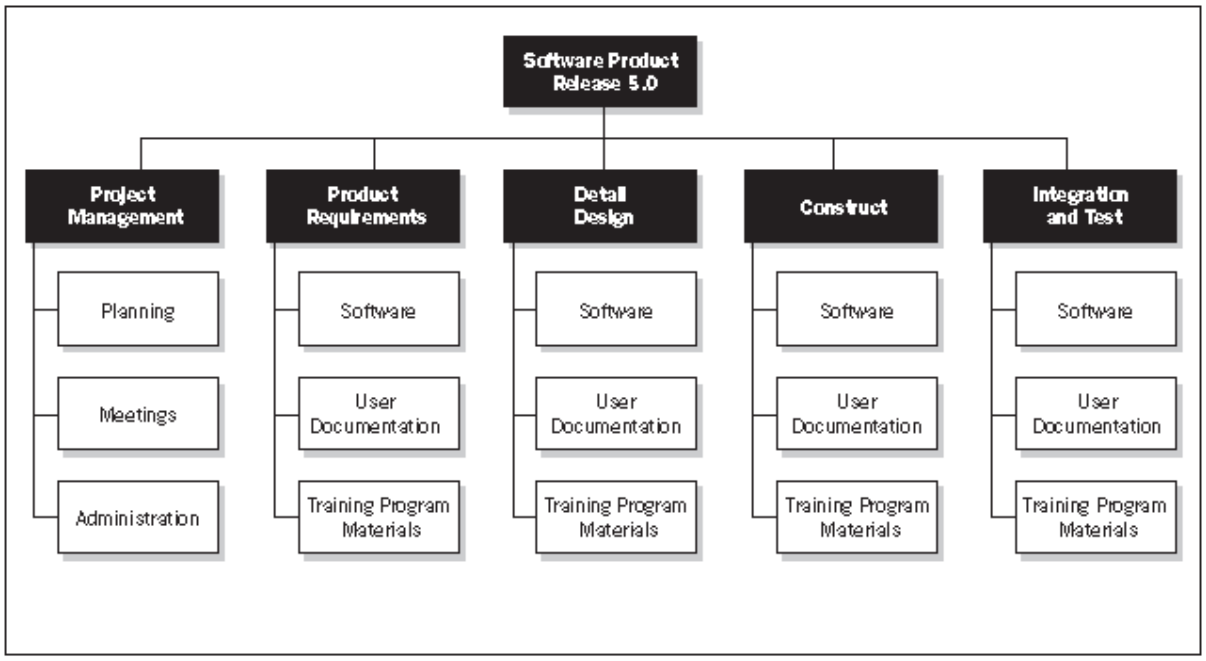


Fig. 1.2: la WBS applicata al progetto di realizzazione di un software entro un progetto IT, tratta da [PMBoK 2008]. Il lavoro di realizzazione della versione 5.0 del prodotto software è scomposto nelle sue fasi elementari, per ciascuna delle quali sono indicati i deliverable.

Il progetto standardizzato

Si definisce **progetto** la "gestione sistematica di un'impresa **complessa**, **unica** e di **durata determinata**, rivolta al raggiungimento di un **obiettivo chiaro e predefinito** mediante un processo continuo di pianificazione e controllo di risorse differenziate e con vincoli interdipendenti di costi-tempi-qualità." [PMBoK 2008]

Un progetto è uno sforzo **temporaneo e organizzato**, intrapreso allo scopo di creare **un prodotto, un servizio o un risultato unici**.

I deliverable sono i **risultati** del progetto. Possono essere:

- un prodotto o manufatto che viene prodotto, quantificabile, che costituisce un prodotto finale o un componente di un prodotto;
- la capacità di erogare un servizio, ad esempio una funzione aziendale a sostegno della produzione o della distribuzione;
- un risultato, come degli esiti o dei documenti.

Il project management o gestione dei progetti

Introduzione al project management

Si definisce ***project management*** l'applicazione di *conoscenze*, *skill*, *strumenti* e *tecniche* alle *attività di progetto* al fine di *soddisfarne i requisiti*. Il project management da alcuni anni segue standard internazionali come PMI [PMBoK 2008], PRINCE2 [PRINCE 2009] e IPMA [Web IPMA].

Il project management standardizzato definisce opportuni processi per le attività di inizio, pianificazione, esecuzione, monitoraggio, controllo e chiusura di un progetto. E' fondamentale identificare i requisiti, fissare obiettivi chiari e raggiungibili, individuare il giusto equilibrio tra le esigenze di qualità, ambito, tempo e costi, che sono in competenza tra di loro; adattare specifiche di prodotto, piani e approccio alle diverse aree di interesse e alle diverse aspettative dei vari **stakeholder**. Per stakeholder si intendono tutte le persone che hanno interesse in un'organizzazione, in un progetto, in un servizio ecc… Possono essere interessati alle attività, agli obiettivi, alle risorse o ai prodotti. Esempi di stakeholder sono:

- Clienti
- Partner
- Organi legislativi
- Impiegati
- Azionisti
- Proprietari
- Ecc…

Per capire meglio il Project Management si possono esplicitare le sue attività parziali, che sono strettamente legate fra loro:

- Gestione complessiva integrata
- Gestione dell'ambito
- Gestione dei tempi
- Gestione dei costi
- Gestione della qualità
- Gestione delle risorse umane
- Gestione della comunicazione
- Gestione dei rischi
- Gestione dell'approvvigionamento

Non tutte queste attività durano per tutto il progetto, alcune sono concentrate all'inizio o alla fine del progetto stesso.

Gli obiettivi del Project Management sono riassumibili come segue:

- Dare una visione realistica del progetto durante tutto il suo ciclo di vita
- Responsabilizzare tutti gli attori coinvolti su obiettivi specifici
- Evidenziare situazioni critiche e proporre valide alternative in modo tempestivo o comunque in tempo utile
- Tracciare un quadro previsionale dell'evoluzione futura del progetto
- Proporre ed imporre regole comuni a tutti gli attori coinvolti
- Assicurare la coerenza tra gli obiettivi parziali assegnati e quelli generali di progetto

Il **Project Manager** è la persona incaricata del raggiungimento degli obiettivi di progetto.

"Molto spesso il Project Management è la formalizzazione del buon senso" (G. Antonelli, direttore sezione PMI Emilia-Romagna).

Funzionamento del project management

Si definisce **process** o **processo** un insieme di *attività coordinate* rivolte ad un *compito/scopo specifico*, per produrre un *risultato* che direttamente od indirettamente *crea valore* per uno *stakeholder*.
All'interno di un progetto sono quindi individuabili i *processi esecutivi* per realizzare l'obiettivo del progetto (ad esempio, scrittura del codice in un progetto informatico) e i *processi di gestione* che devono realizzare il controllo del progetto, ossia tutte le attività sopra descritte che formano il project management.
I processi di gestione, definiti dal project management, sono suddivisibili in gruppi in base alla loro durata temporale entro il progetto:

- Gruppo di processi di concezione/avvio,
- Gruppo di processi di pianificazione
- Gruppo di processi di monitoraggio e controllo, che dovrebbero durare per tutta la fase esecutiva del progetto
- Gruppo di processi di chiusura
- Accanto ad essi si trova anche il gruppo dei processi di esecuzione, che sono esecutivi e non fanno parte dei processi di gestione

I dettagli interni di ogni gruppo di processi sono mostrati in fig. 2.1, mentre in fig. 2.2 viene vista la loro successione temporale. L'esecuzione è indicata in colore diverso dalle altre in quanto è l'insieme dei processi esecutivi e quindi non fa parte del project management.

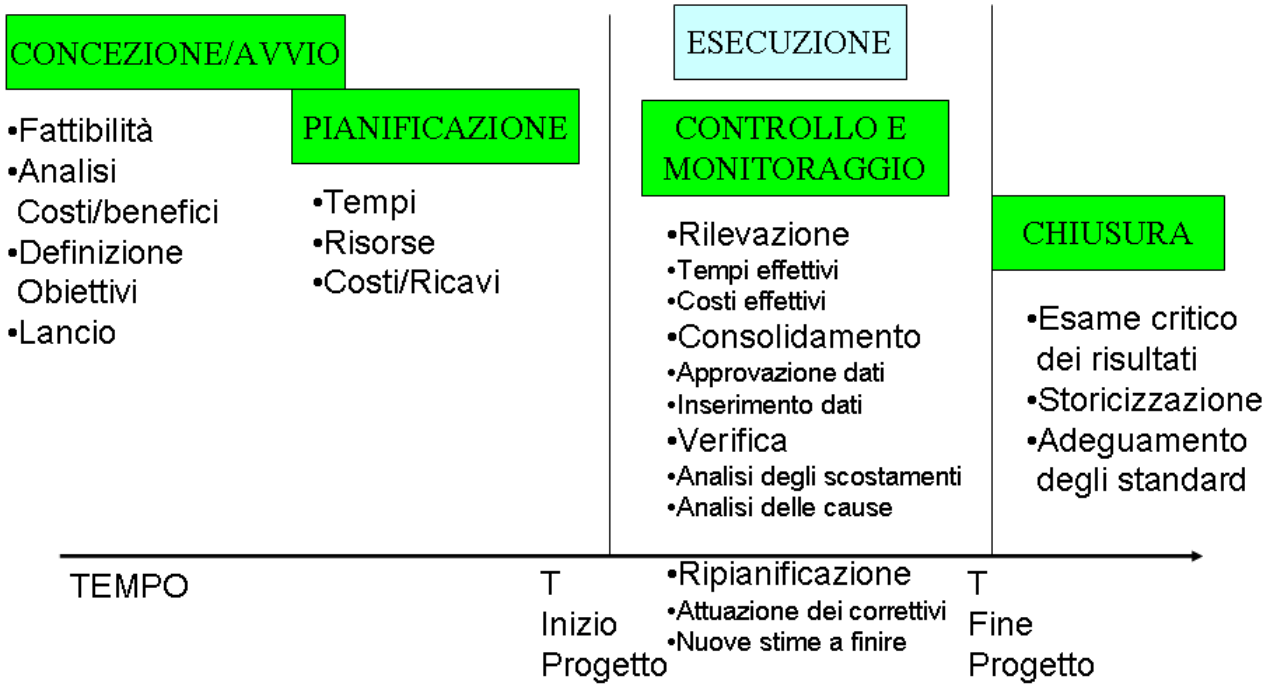


Fig. 2.1: dettagli dei gruppi di processi del project management. L'esecuzione è indicata in colore diverso in quanto non fa parte del project management.

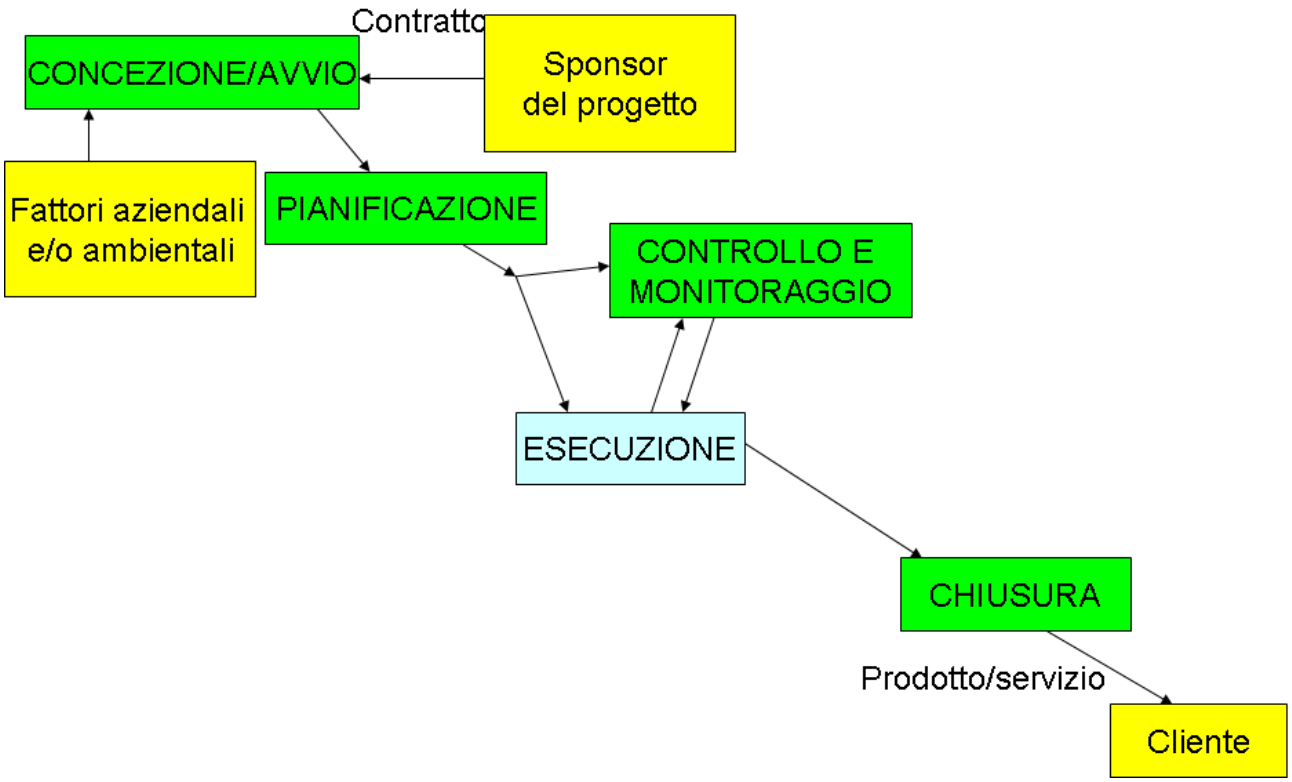


Fig. 2.2: successione temporale esplicita dei gruppi di processi del project management. Si noti l'interazione continua fra l'esecuzione ed il controllo durante tutta la fase esecutiva del progetto.

I processi di controllo e monitoraggio, nella loro interazione con i processi esecutivi, seguono un modello teorico chiamato ciclo PDCA o ciclo di Deming, composto dalla successione ciclica di 4 fasi:

- Pianificare (Plan)
- Eseguire (Do)
- Verifcare la buona riuscita dell'esecuzione (Check)
- Agire, se necessario, per il cambiamento (Act)

All'interno di un progetto dunque le azioni dei processi di project management possono essere rappresentate logicamente come in fig. 2.3:

- dopo la concezione del progetto, si valuta se il progetto è fattibile, rispetto ai vincoli di tempo, budget, risorse a disposizione ecc…;
- se non è fattibile, il progetto termina con l'annullamento;
- se è fattibile, si passa alla fase di pianificazione, che sarà descritta sotto; eventuali variazioni degli obiettivi del progetto richiedono ancora di passare dalla pianificazione;
- a fianco delle fasi operative, non indicate in figura, sono le fasi di rilevazione e consolidamento e di verifica, corrispondenti al controllo e monitoraggio della fig. 2.2;
- qualora dalla verifica emergano scostamenti sopra appropriate soglie rispetto a tempi e/o costi o necessità di cambiamenti in corso d'opera (ad esempio, dovuti a cambiamenti nella legislazione corrente) si passa dalla fase di ripianificazione
- quando la rilevazione conferma che gli obiettivi del progetto sono stati raggiunti, si può passare alle fasi di chiusura.

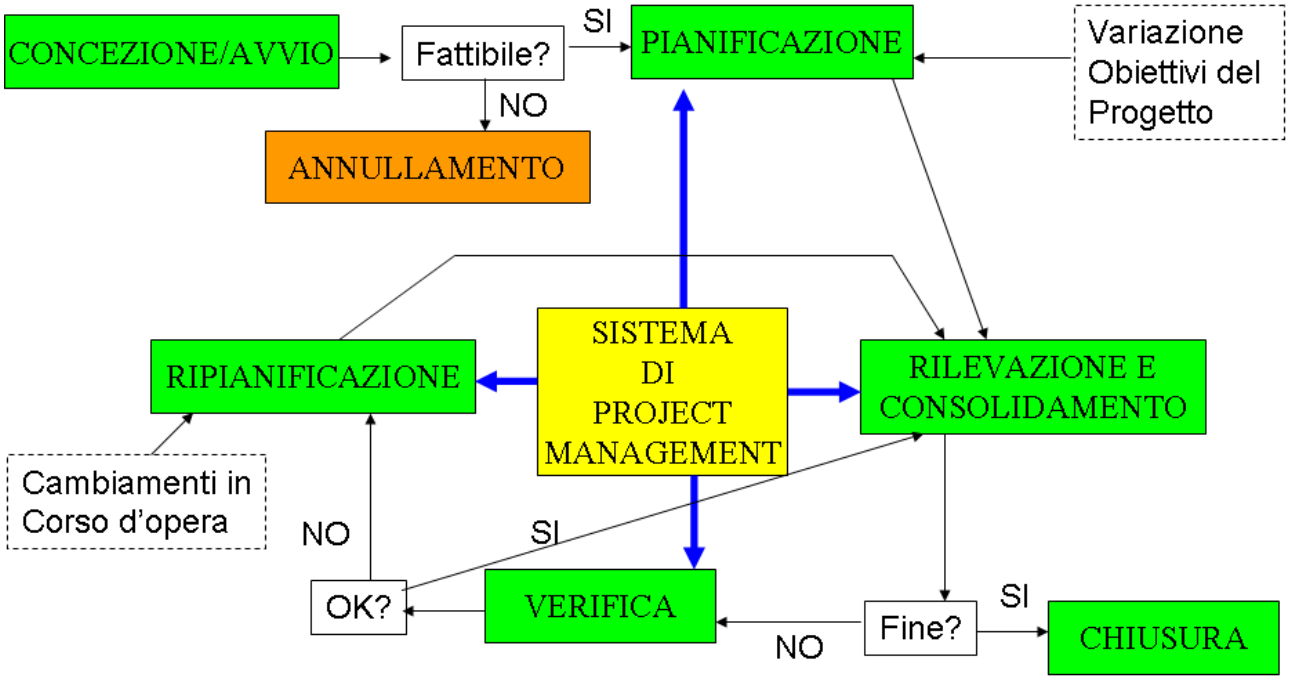


Fig. 2.3: le azioni di controllo nel corso della esecuzione di un progetto.

Strumenti per il project management

Scomposizioni strutturate di attività e attori

Un'altra funzionalità molto utile del Project Management è la scomposizione progressiva sia delle attività da svolgere (ossia la WBS definita nel capitolo precedente), sia delle entità organizzative (ad esempio, singole persone, singoli uffici, ecc…) che le svolgono.

Per *Struttura Analitica dell'Organizzazione* o *Scomposizione Strutturata dell'Organizzazione*, di solito indicata con l'acronimo inglese ***OBS*** (Organization Breakdown Structure) si intende una metodologia standard per scomporre l'organizzazione per assegnare in modo puntuale le responsabilità delle singole attività alle componenti elementari dell'organizzazione. Le unità sono solitamente le singole persone ma possono anche rappresentare uffici o altre suddivisioni gerarchiche o funzionali dell'azienda.

Entro un progetto, di solito la OBS è creata dopo la WBS in maniera ad essa complementare, per individuare le responsabilità nelle singole attività trasformando quindi le attività in incarichi assegnati a specifiche persone o divisioni. Quindi la OBS non è necessariamente la stessa suddivisione organizzativa espressa dall'organigramma di un'azienda o ente, ma può essere anche costruita ad hoc.

Il processo di integrazione fra WBS e OBS è rappresentato in fig. 2.4.

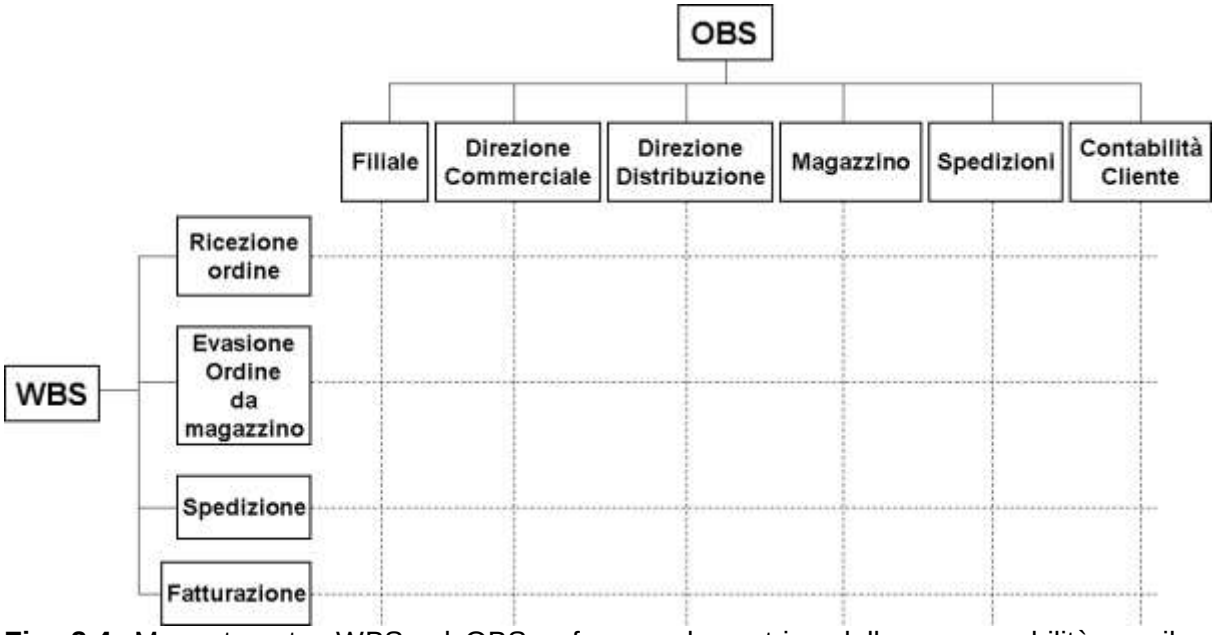


Fig. 2.4: Mappatura tra WBS ed OBS a formare la matrice delle responsabilità per il caso di soddisfacimento di un ordine di un cliente. La risultante matrice è mostrata in fig. 2.5

Le matrici di responsabilità

Il metodo ***LRC*** (Linear Responsibility Charting), evoluzione del metodo omonimo introdotto negli Stati Uniti negli anni'50, prevede una specificazione dei ruoli e delle strutture nei processi, attraverso una visione tabellare della responsabilità organizzativa, che integra l'organigramma ed incrocia le attività o fasi del processo con le strutture organizzative o con loro parti (anche singole risorse umane). Tale metodo, integrato negli standard PMI, ha condotto alla *matrice delle responsabilità*, detta anche *matrice RAM (Responsibility Assignment Matrix)*. La forma più diffusa di matrice RAM è la ***matrice RACI***, definita dallo standard PMI. L'acronimo RACI, dalle iniziali delle parole inglesi Responsible, Accountable, Consult, Inform, esprime il ruolo che ogni persona o divisione organizzativa coinvolta nel processo assume relativamente alla fase indicata, come sotto descritto.

- ***R*** (in italiano indicabile come **Realizzatore**)- significa che la persona o il ruolo o la divisione è responsabile della deliverable (nel caso di progetto) o della attitvità (nel caso di processo). Quindi la persona o la divisione è attivamente coinvolta con la propria opera nella attività.
- ***A*** (in italiano indicabile come **Approvatore**)- significa che la persona (o ruolo) approva la deliverable e quindi si assume la responsabilità di tale approvazione "mettendoci la faccia". Normalmente esiste solo una persona responsabile (A) di approvare, mentre più persone possono contribuire (R). I ruoli A ed R non sono mutuamente esclusivi: una persona può assumere contemporaneamente i ruoli A ed R rispetto ad una certa fase.
- ***C*** (in italiano indicabile come **Consultato** o in altri contesti **Contributore**)- significa che la persona (o ruolo, o divisione) viene consultata sul rilascio della deliverable. Ciò implica che c'è un confronto con una comunicazione bidirezionale.
- ***I*** (in italiano indicato con **Informato**) - significa che la persona o il ruolo viene informata della deliverable. Questa è una comunicazione ad una via, monodirezionale.

Dal ruolo svolto in relazione ad una certa attività o fase si può ricavare anche se la persona o ruolo è coinvolto al 100% del suo tempo lavorativo oppure no. Ad esempio, chi è Responsible sarà probabilmente impegnato al 100% su quella attività, mentre chi è Consult sarà probabilmente coinvolto al 10%.

Dallo standard sono derivate altre matrici simili con più ruoli, largamente usate.
In fig. 2.5 viene rappresentato attraverso una matrice RACI un esempio di mappatura delle varie attività di gestione di un ordine su un organigramma. In fig. 2.6 è invece una matrice derivata, con più ruoli di dettaglio rispetto alla matrice RACI.

	Fliale	Dir. Com.	Dir. Distr.	Magazzino	Sped.	Contab. Cli.
Ricezione ordine	R	A	I	I		C
Evasione ordine da magazzino	I	I	A	R	I	
Spedizione	I		A	C	R	I
Fatturazione	I	I	I			AR

Fig. 2.5: Esempio di LRC rappresentata come matrice RACI, adattato da [BFM 2001].

	RU1	RU2	RU3	RU4	RU5	RU6
Analisi	R	A	I	C	C	
Definizione	I	A	C			R
Gestione	R	A	I			S
Monitoraggio	R	A	I	C	C	S

- R = responsabile
- S = supporta
- I = informato
- A = approva
- C = consultato
- V = verifica

Fig. 2.6 Esempio di matrice RAM derivata dalla RACI ma con più ruoli, come indicato nella legenda. In questo caso si vuole esprimere la differenza fra chi è semplicemente **C**onsultato (occupazione massima 10% del tempo) e chi invece **S**upporta l'attività (occupazione media del 30% del tempo).

Strumenti per l'assegnazione degli incarichi

Accanto alle matrici RAM e ai documenti testuali o tabelle simili alla LRC, spesso vengono usati due strumenti grafici: il ***diagramma delle dipendenze*** e/o un suo parente stretto, il ***grafo di progetto***.

Il diagramma delle dipendenze definisce le attività e le risorse impiegate in esse ed esprime le dipendenze fra le varie entità ed attività coinvolte nel processo. Il diagramma può fotografare la situazione in un dato momento (es. iniziale e finale) e può esprimere lo stato effettivo (IS, AS-IS) e quello desiderato (SHOULD, TO-BE).
Il grafo di progetto, chiamato anche diagramma a rete del progetto, è un grafo i cui nodi rappresentano attività da svolgere ed i cui archi esprimono condizionamenti tra attività. Oltre alle informazioni delle dipendenze fra attività ed azioni da un lato e risorse o regole dall'altro, esprime anche le informazioni di flusso logico. Il grafo di progetto viene usato spesso entro la metodologia PERT/CPM. Spesso col termine PERT si indica anche direttamente la rappresentazione grafica del progetto.
Le tecniche basate sul PERT e il connesso metodo sono tecniche ideate ed utilizzate per affrontare l'analisi di grandi progetti di ricerca, costruzione, programmazione ed organizzazione aziendale. Lo scopo principale è quello di pianificare e coordinare diverse attività che concorrono al raggiungimento dell'obiettivo in oggetto, assegnando nel tempo le risorse disponibili a ciascuna attività in modo "razionale". Un esempio di PERT è in fig. 2.7.
In ogni caso la metodologia risultante nella pratica è:

1. Determinazione delle attività del progetto, attraverso una WBS;
2. Individuazione delle dipendenze logico-temporali fra le attività, per poterne definire una sequenza precisa;
3. Calcolo dei tempi delle singole attività e della somma estesa a tutto il progetto;
4. Mappatura dei tempi su un calendario reale, tenendo presenti gli intervalli di disponibilità delle risorse umane cui le attività vengono assegnate come incarichi.

Una volta definite tutte le relazioni di dipendenza e le durate delle singole attività, si può passare al punto 4: la pianificazione temporale vera e propria, ossia alla mappatura dei tempi di esecuzione delle attività su un asse temporale corrispondente al calendario, e quindi comprensivo delle informazioni relative alle festività, orari di lavoro ecc...
In questa fase vengono usati i ***diagrammi di Gantt***, detti anche diagrammi a barre. Due sono i tipi fondamentali di diagrammi di Gantt:

1. *diagramma di Gantt delle attività*, in cui l'asse orizzontale rappresenta il tempo e le attività del progetto vengono indicate sull'asse verticale; le singole attività sono rappresentate come barre parallele all'asse temporale orizzontale; esso evidenzia tempi, dipendenze e criticità e permette di monitorare giorno per giorno l'andamento dei progetti, come mostrato in fig. 2.8;
2. *diagramma di allocazione delle risorse o diagramma degli incarichi* in cui le attività sono già suddivise fra le varie risorse, ovvero sono state ad esse assegnate; le risorse sono rappresentate come punti sull'asse verticale, per cui la barra corrispondente rappresenta i periodi di occupazione delle singole risorse, come mostrato in fig. 2.9.

Nelle figure seguenti è rappresentato un esempio generico di un semplice processo suddiviso in cinque attività, di cui le B e C sono alternative alla D. Viene presentato il grafo di progetto, applicato il PERT ed il conseguente Gantt. Si veda [PMBoK 2008] per approfondimenti.

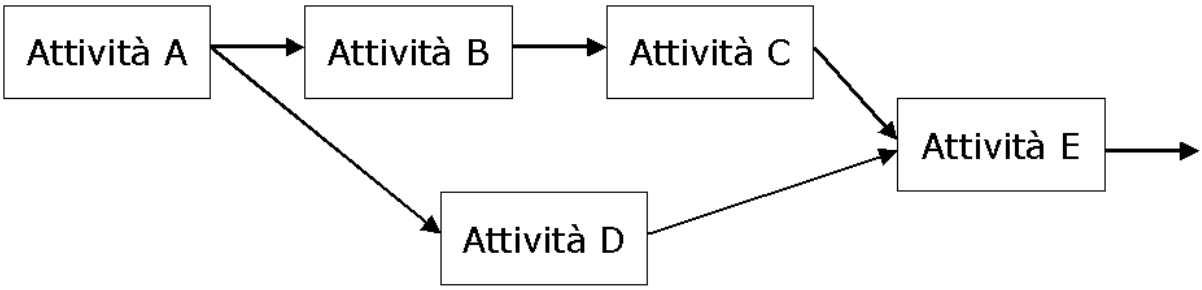


Fig. 2.7: Il grafo di un progetto suddiviso in 5 attività, di cui le B e C sono alternative alla D. Tutte le attività sono bloccanti, ossia non esistono parallelismi e un'attività conseguente può iniziare solo dopo che la precedente si è conclusa.

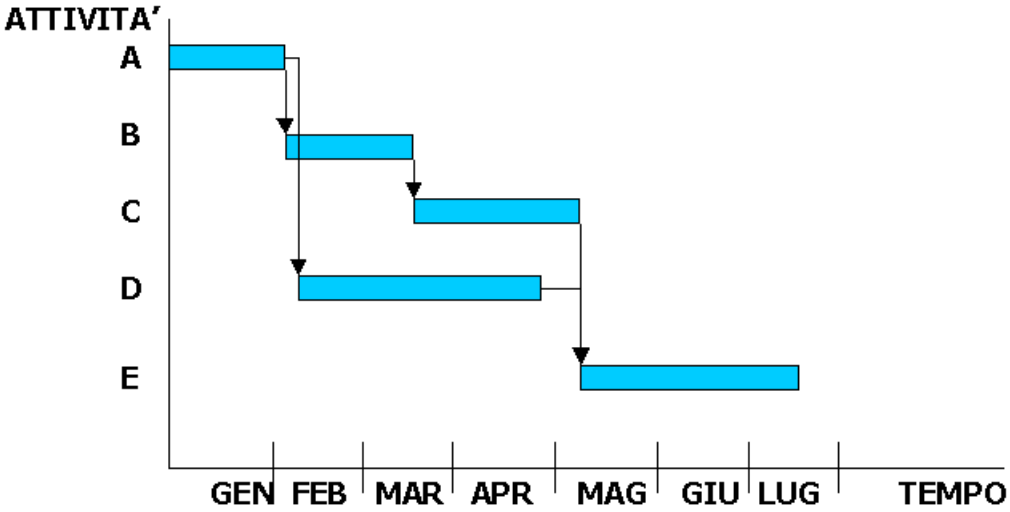


Fig. 2.8: Il diagramma Gantt delle attività risultante dal grafo di fig. 2.7; si ricordi che le attività B e C sono alternative alla D.

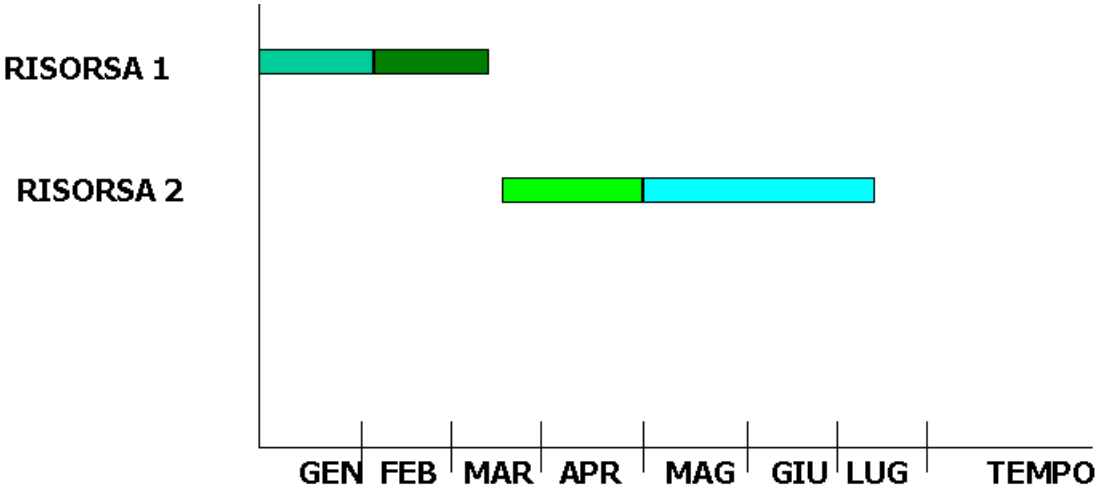


Fig. 2.9: Supponendo di avere due risorse umane in grado di svolgere le attività del diagramma di fig. 2.8, ecco una ipotesi di diagramma di allocazione di risorse risultante. Le quattro attività sono evidenziate dai diversi colori e si suppone che venga scelta la successione A, B, C, E, con la risorsa 1 allocata per A e B, mentre la risorsa 2 viene allocata per C ed E.

Uso degli strumenti per la pianificazione e gestione

Mettendo insieme gli strumenti sinora visti, è possibile scomporre la fase di pianificazione nei seguenti passi, schematizzati insieme agli strumenti in uso in fig. 2.10:

1. Stabilire cosa si deve fare, attraverso la WBS, scomponendo il progetto in una successione di attività parziali;
2. Stabilire, attraverso la OBS, chi fa che cosa, ossia chi sono i responsabili delle varie attività, con i vari ruoli coinvolti;
3. Assegnare le attività alle persone sotto forma di incarichi, attraverso una matrice RAM;
4. Stabilire la successione temporale esatta delle attività, ossia la logica di progetto, attraverso un PERT;
5. Mappare la successione delle attività su un calendario reale, attraverso i diagrammi di Gantt;
6. Verificare il carico e l'occupazione effettiva delle singole risorse, stabilendo un piano delle risorse;
7. Calcolare i costi complessivi, stabilendo il piano dei costi.

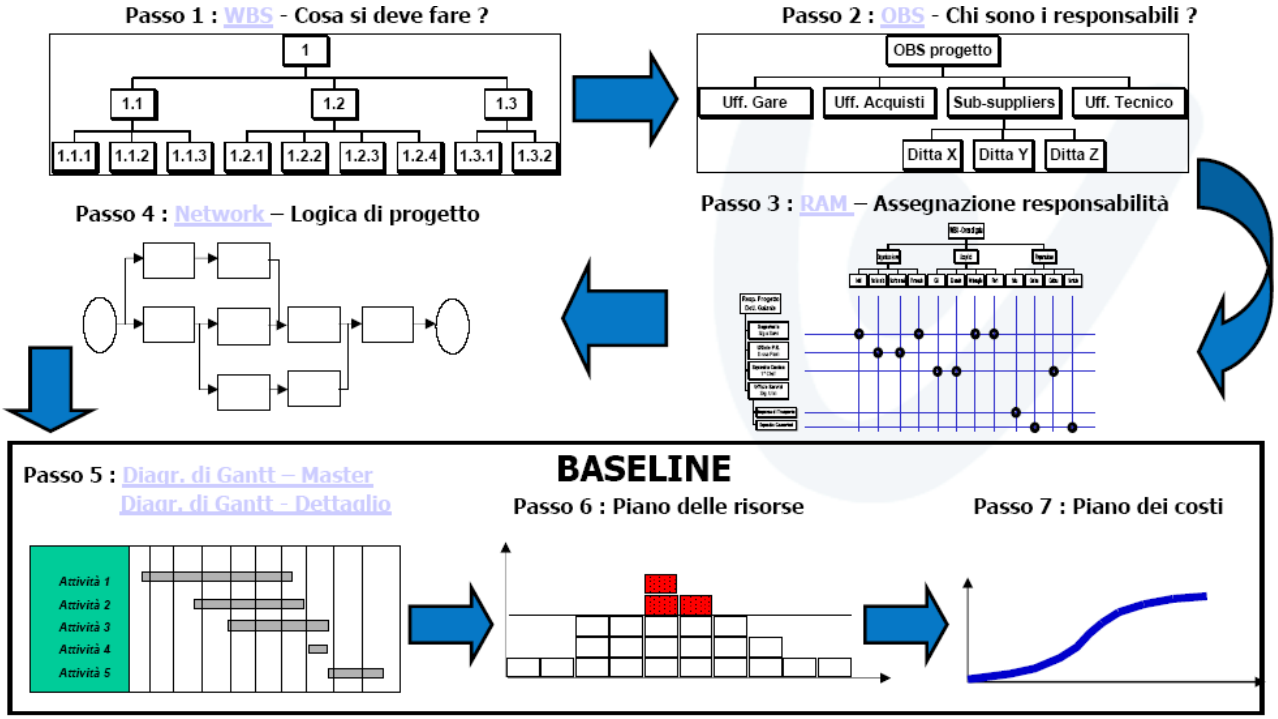


Fig. 2.10: successione delle fasi di pianificazione del progetto con i relativi strumenti usati.

Le pianificazioni devono essere realistiche. Occorre infatti:

- Avere pianificazioni complete ma concise
- Avere pianificazioni adattabili (almeno entro certi limiti)
- Evitare che pianificazioni arbitrarie sconvolgano tutto
- Non demoralizzare il gruppo
- Progetti lunghi vanno suddivisi
- Progetti parziali ben definiti e con risultati chiari

I costi vanno pianificati rispetto al budget disponibile, come già visto anche nel capitolo 1.
Il progetto deve essere controllato realizzando le funzioni schematizzate in figura 2.3. Questo significa:

- Rilevare gli avanzamenti del progetto entro le attività operative
- Verificare gli avanzamenti

- Analizzare le tendenze in atto e l'andamento rispetto alle scadenze fondamentali (indicate con **milestone**)
- Approvare gli avanzamenti
- Individuare se ci sono scostamenti rispetto alle pianificazioni e ricercarne le cause
- Definire ed apportare eventuali attività correttive
- Aggiornare di conseguenza la pianificazione temporale ed economica.

La pianificazione economica è fondamentale per i progetti grandi: difficilmente il gruppo di progetto ha a disposizione fin dall'inizio tutto il budget necessario per il progetto stesso. Pertanto devono essere accuratamente pianificati anche i flussi di cassa di ingresso (ad esempio, garantiti da vendite di altri prodotti software) che arrivino a coprire le spese del progetto nel tempo.

Il progetto informatico

La produzione del software

Cosa significa realizzazione del software? Significa tradurre un'idea in algoritmi, strutture dati e, in definitiva codice sorgente, che poi deve essere compilato per ottenere eseguibili e librerie, pronti per essere copiati e/o installati nella locazione ove dovranno svolgere il loro compito.
Pertanto possiamo affermare che l'obiettivo fondamentale per ottenere applicativi è scrivere il codice sorgente del software. Ciò comporta essenzialmente le seguenti attività lavorative:

- Definizione (attraverso analisi e progettazione) di *cosa* il programma deve fare
- Realizzazione (progettazione di dettaglio e scrittura) del **codice sorgente** di un programma applicativo (ossia dell'insieme di istruzioni scritte in formato testo ASCII secondo un particolare linguaggio di programmazione) in base a quanto stabilito nel punto precedente
- Compilazione del medesimo codice e correzione degli errori (debugging)
- Produzione dell'eseguibile risultante dalla compilazione
- Eventuale pacchettizzazione dell'eseguibile in un programma di installazione o sua installazione diretta nel computer ove esso svolgerà il servizio per cui è stato concepito

Le altre fasi (analisi, progettazione, test ecc…) sono essenzialmente fasi ausiliarie, che devono aiutare alla scrittura di buon codice sorgente, per quanto ormai fondamentali per ottenere un buon progetto.
Si definisce ***Ingegneria del software*** la disciplina che aiuta ad ingegnerizzare il processo di sviluppo del software

- stabilendo delle regole (quindi mettendovi ordine)
- rendendolo più efficiente
- applicando metodologie di project management

aiutando nella gestione dei grandi progetti (con anche decine di persone coinvolte)

Il progetto informatico per la produzione del software

Si parla spesso di "progetto" software, intendendo con questo termine tutto il lavoro dall'idea alla realizzazione del software stesso, che viene anche inteso come unità, risultante da un contratto tra un cliente ed un fornitore e quantificabile in termini di prezzo.
Ma si può definire il "progetto" software anche come l'organizzazione del processo di sviluppo del software, ossia l'applicazione del project management a tale processo di sviluppo. L'organizzazione è necessaria per team di sviluppatori, pena il rischio di perdita di controllo del progetto stesso. Prima dell'inizio della fase di codifica vera e propria, ossia di stesura del codice sorgente, devono essere stabiliti i requisiti da raggiungere e, quantomeno, le suddivisioni del lavoro fra i membri del gruppo di lavoro.
La struttura di base del progetto software comprende le seguenti fasi

- Raccolta dei requisiti
- Analisi
- Progettazione
- Implementazione (sviluppo)
- Test
- Installazione (messa in produzione)
- Manutenzione (ordinaria ed evolutiva)

Il software, nel suo complesso, ha poi un ciclo di vita più esteso della semplice sua produzione, che può essere schematizzato come in figura 1, dove la definizione corrisponde a raccolta dei requisiti ed analisi insieme, l'accettazione al termine dei test e l'avvio alla installazione.
Nei paragrafi seguenti verranno presentate le varie fasi del progetto informatico con le loro caratteristiche specifiche.

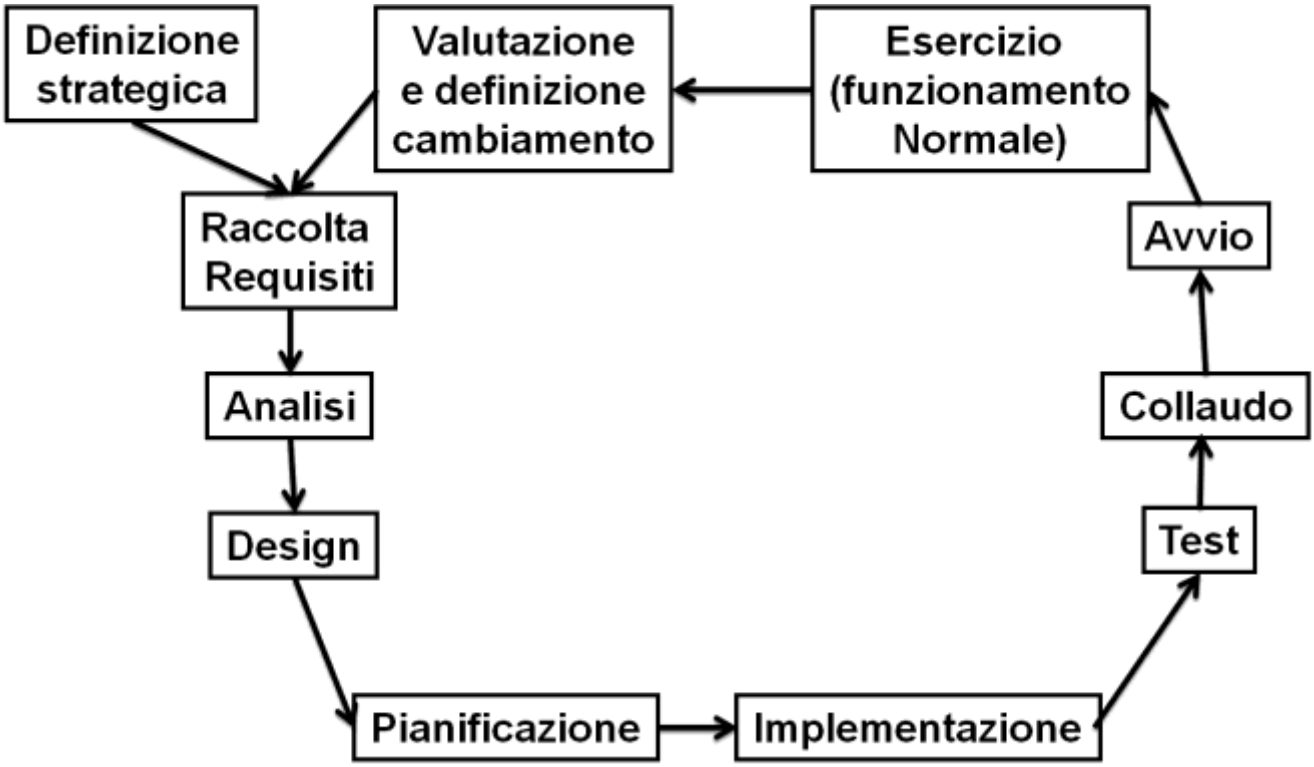


Fig. 3.1: Il ciclo di vita del software; si noti che, mentre la prima versione di un applicativo parte dall'idea, le versioni successive fondono i risultati delle valutazioni d'uso dell'applicativo con nuove idee (manutenzione evolutiva del software).

Raccolta dei requisiti ed Analisi

Ricordiamo che in questo contesto si definisce ***requisito*** una proprietà o una qualità, di natura funzionale o no, che un prodotto software deve avere o soddisfare.
Il progetto inizia con la fase di raccolta dei requisiti, realizzata attraverso la cosiddetta *elicitazione dei requisiti* (*requirement elicitation*). Con questo termine, mutuato dalla psicologia, si intende un insieme di diverse tecniche, combinate fra loro, per riuscire ad "estrarre dalla mente del cliente o committente le idee per capire cosa il software oggetto del contratto dovrà fare".
Le tecniche di elicitazione più usate sono:

- le *interviste*, durante le quali si pongono alle persone del cliente domande esplicite;
- l'*osservazione*, durante la quale si osservano "sul campo" le funzioni svolte in cui il software dovrà intervenire;
- il *gruppo di lavoro* (si veda [BABoK 2009] per approfondimenti).

La fase di analisi segue la raccolta dei requisiti e serve per descrivere in modo formale e preciso i requisiti che dovrà avere il sistema oggetto del progetto. Durante la fase di analisi devono essere individuati e formalizzati tutti i requisiti, funzionali e non, cui il software deve adempiere, i vincoli che esso deve rispettare e il contesto dove esso deve operare. Le informazioni formalizzate nella fase di analisi rappresentano il punto di partenza per la

progettazione di un prodotto software e per l'intero processo della sua realizzazione, validazione e manutenzione.
La fase di analisi è fondamentale per la buona riuscita di un progetto software. Come mostrato in fig. 3.2, un errore eventualmente compiuto durante la fase di raccolta requisiti o di analisi produce effetti sempre maggiori e sempre più costosi da correggere man mano che si procede con le fasi successive di progetto.

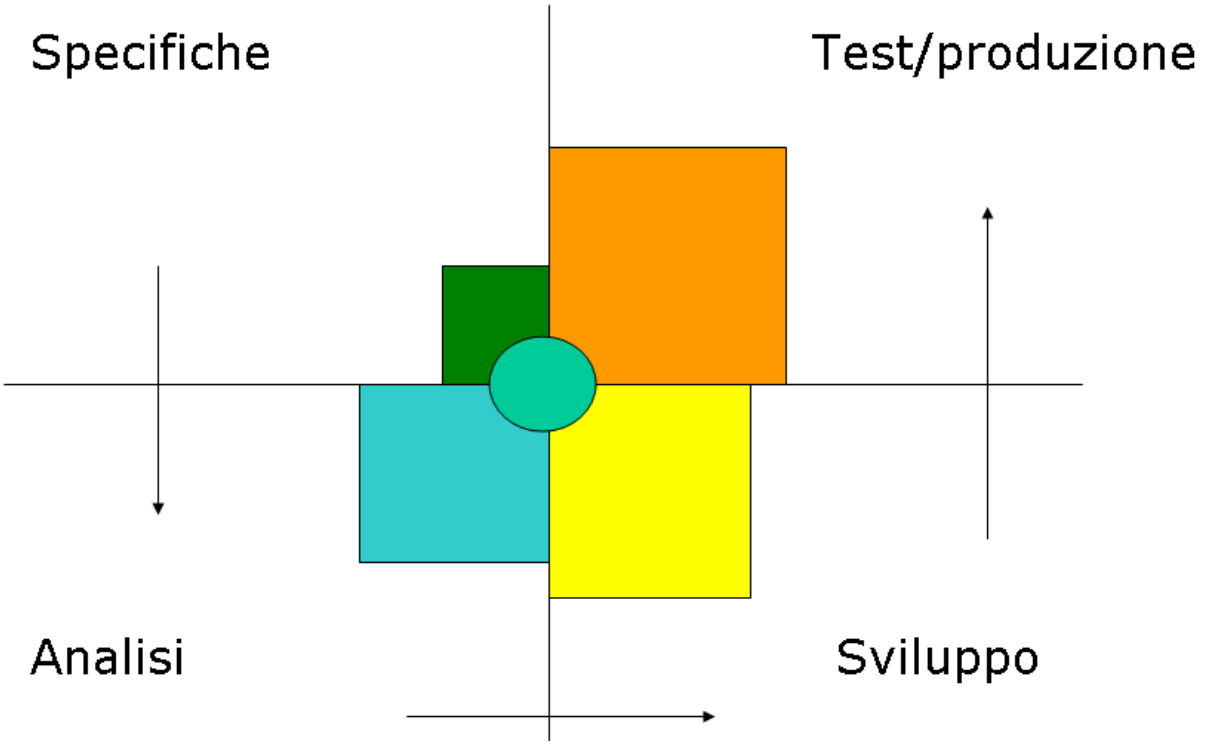


Fig. 3.2: Se durante la raccolta delle specifiche si commettono errori, il loro impatto aumenta sempre più al procedere delle fasi successive e lo stesso vale per il costo della correzione degli errori.

In progetti piccoli l'analisi è quasi sempre una fase conclusa una volta per tutte all'inizio del processo di realizzazione del prodotto software, in progetti grandi invece spesso l'analisi viene svolta iterativamente nel processo stesso. Il particolare processo di sviluppo usato, con il relativo modello del ciclo di vita del software, definisce il ripetersi o meno dell'analisi, così come di altre fasi successive.
L'obiettivo della fase di analisi è produrre una **descrizione completa formalizzata**, con il giusto livello di dettaglio, di **tutto ciò che il sistema deve fare** (requisiti funzionali), dell'ambiente in cui dovrà operare (requisiti non funzionali) e dei **vincoli che dovrà rispettare**. Tale descrizione dovrà spiegare quindi cosa il sistema dovrà fare senza affrontare il come dovrà farlo, in modo tale da potere servire come base di discussione ed eventualmente di contratto tra il team di sviluppo ed il committente, per arrivare ad una specifica univocamente interpretabile da ambo le parti per il progetto. La descrizione del sistema software da realizzare seguirà quindi il modello scatola nera, concentrandosi solo sulle interazioni fra il sistema ed il mondo esterno, in primo luogo sulle interfacce utente.
I **documenti di analisi**, talvolta chiamati anche **documenti di specifica** o semplicemente **specifiche**, sono il prodotto risultante dalla fase di analisi e devono contenere in modo chiaro le informazioni sopra definite. La loro forma, i linguaggi e le simbologie usate sono ovviamente dipendenti dal tipo di processo di sviluppo seguito.

La fase di progettazione

L'output della fase di analisi deve descrivere con precisione e senza ambiguità i requisiti che dovrà avere il sistema oggetto del progetto.
Partendo dai risultati dell'analisi, la fase di progettazione deve produrre le istruzioni operative per la realizzazione effettiva del progetto informatico (implementazione). Le istruzioni devono avere il giusto livello di dettaglio ed essere espresse in forma di documenti strutturati. Pertanto, la progettazione di un'applicazione è composta dalle attività per individuare la soluzione implementativa migliore rispetto agli obiettivi funzionali, a quelli non funzionali ed ai vincoli. Queste attività possono essere di varia natura, essere svolte in tempi e modi diversi in base all'approccio seguito, ma in generale aiutano progettisti e team di sviluppo a prendere decisioni importanti, spesso di natura strutturale.
Il risultato della progettazione è la definizione dell'architettura del sistema, intendendo con questo termine l'organizzazione strutturale del sistema stesso, che comprende i suoi componenti software, le proprietà visibili esternamente di ciascuno di essi (l'interfaccia dei componenti) e le relazioni fra le parti.
In analisi requisiti e struttura del sistema sono rappresentati in forma astratta e (teoricamente) indipendente dalla tecnologia. La progettazione tiene conto anche di tutti i fattori relativi all'utilizzo di una tecnologia concreta e quindi, a differenza dell'analisi, la progettazione non può essere svolta indipendentemente dalla tecnologia utilizzata. Durante la progettazione devono anche essere definiti tutti gli aspetti necessari per una implementazione non ambigua.
Uno strumento molto usato nella progettazione è il diagramma di flusso o flow-chart (si veda [Camuso3 2012]), oppure la sua evoluzione UML activity diagram o diagramma delle attività (si veda anche [Fowler 2010]). Ambedue gli strumenti servono a realizzare la scomposizione delle attività da compiere in elementi sempre più piccoli che possano essere facilmente implementati con opportuni insiemi di istruzioni del linguaggio di programmazione scelto, ossia la WBS del lavoro di programmazione.

La fase di implementazione o scrittura del codice

La scrittura del codice sorgente può essere svolta con molti strumenti, il più semplice dei quali è l'editor di file ASCII di base. Per la complessità che i moderni linguaggi ad oggetti richiedono però tale approccio è troppo poco produttivo. Dovendo infatti garantire il rispetto di tempi stretti, è necessario disporre di tutte le funzioni di facilitazione integrate entro un unico strumento per la scrittura del codice sorgente.
Un *integrated development environment* (IDE), in italiano ambiente di sviluppo integrato, (conosciuto anche come integrated design environment o integrated debugging environment, rispettivamente ambiente integrato di progettazione e ambiente integrato di debugging) è un software che aiuta i programmatori nello sviluppo del codice.
Normalmente consiste in un editor di codice sorgente, un compilatore e/o un interprete, un tool di building automatico, e (solitamente) un debugger.
A volte è integrato con un sistema di controllo di versione e con uno o più tool per semplificare la costruzione di una GUI.
Alcuni IDE, rivolti allo sviluppo di software orientato agli oggetti, comprendono anche un navigatore di classi, un analizzatore di oggetti e un diagramma della gerarchia delle classi.

Sebbene siano in uso alcuni IDE multi-linguaggio, come Eclipse, NetBeans e Visual Studio, generalmente gli IDE sono rivolti ad uno specifico linguaggio di programmazione, come Visual Basic o Delphi.
La scrittura del codice sorgente, per quanto spesso poco considerata, rimane comunque la fase fondamentale di ogni progetto informatico. L'analisi e la progettazione possono essere state svolte al meglio, ma, se il codice viene scritto male, l'applicazione risultante avrà problemi e funzionerà male. Facendo un'analogia nell'ambito dell'ingegneria civile, è chiaro che anche il progetto più bello, se i pilastri non sono fabbricati bene, se i mattoni non sono posati con accuratezza o se i materiali impiegati sono di scarso pregio, darà origine ad un edificio di pessima qualità.
Per questo anche metodologie di programmazione moderna tendono a fare diventare la fase di stesura di codice quella principale di tutto un progetto informatico (si veda ad esempio [Web Agile]).
Durante la stesura del codice ha luogo anche il primo debugging, ossia la rimozione degli errori di sintassi e degli errori più evidenti, come la mancata inizializzazione di variabili, che possono compromettere la compilazione o il funzionamento del programma.

La fase di test

Una volta che il programma è stato completato o comunque può iniziare a funzionare, occorre verificare che il suo funzionamento sia conforme a tutte le specifiche che erano state stabilite nella fase di analisi. Questo è lo scopo fondamentale della fase di test.
Gli errori che portano al non rispetto delle specifiche sono di solito molto più insidiosi da scoprire rispetto a quelli che vengono trovati durante il debugging. Non sono errori di sintassi, ma errori logici e concettuali, come, per esempio, lo scrivere il segno '+' invece del segno '-' entro un algoritmo che richieda la sottrazione e non la somma.
La fase di test prevede quindi di verificare il comportamento effettivo del programma rispetto a quello previsto e di segnalare le differenze di comportamento ai programmatori che dovranno procedere alla ricerca e alla eliminazione delle cause di tali differenze.

La fase di avvio o entrata in produzione

Dopo il test, raggiunto un livello sufficiente di qualità, il programma può entrare in produzione. Questo termine ha un significato diverso secondo il tipo di programmi in realizzazione.
Per i programmi destinati alla vendita presso il pubblico, o alla distribuzione se gratuiti, questa fase rappresenta il rilascio sul mercato.
Per i programmi realizzati specificatamente per un cliente (i cosiddetti "programmi custom"), questa fase rappresenta l'installazione ed il collaudo presso la sede del cliente che li ha richiesti.
Al termine di questa fase i programmi iniziano la propria vita operativa, durante la quale svolgono il compito previsto nel contesto per cui sono stati progettati, che può proseguire anche per molti anni.

La fase di manutenzione

Durante la vita operativa possono verificarsi necessità di interventi correttivi o di aggiornamento sui programmi, che prevedono nuove fasi di progettazione, implementazione e test. Tali interventi correttivi sono raggruppabili in due distinte famiglie:

1. **Manutenzione ordinaria**, l'insieme di interventi correttivi necessari per via di errori sfuggiti ai test o dovuti al funzionamento del programma in condizioni non previste durante la sua progettazione, come ad esempio il funzionamento su Windows7 di un programma nato per WindowsXP;
2. **Manutenzione evolutiva**, l'insieme di interventi di variazione od arricchimento delle funzioni del programma per via di nuove necessità operative del programma stesso; un esempio è l'aggiornamento continuo dei programmi gestionali per stare aggiornati rispetto alle normative fiscali

In generale, comunque, ogni programma durante la sua vita è soggetto a interventi evolutivi e correttivi. Solo una piccola percentuale di programmi non viene più toccata dopo il rilascio.

Il linguaggio UML

Introduzione a UML

Unified Modeling Language (UML) è un linguaggio semi-grafico unificato per la modellazione di concetti, entità, funzionalità, processi e relazioni che fra essi intercorrono (si veda [Fowler 2010] per approfondimenti). UML nasce nel 1997, unificando precedenti sintassi di modellazione (Booch, OMT, OOSE) ed oggi è uno standard internazionale gestito dal consorzio OMG (si veda [Web OMG]). Dopo il 1997 le estensioni standard di UML si susseguono e anche l'uso viene esteso alle varie fasi di realizzazione dei sistemi IT, comprendendo anche la fase di analisi dei processi business (indicata anche come "analisi business", si vedano a tal proposito [EP 2000] e [BABoK 2009]) . Infatti nel tempo è emersa la necessità di un linguaggio chiaro comune a tutti i componenti del sistema informativo, dal business alla tecnologia.
Il linguaggio UML serve a descrivere, in modo grafico e "compatto"

- I requisiti utente (use case), ossia, in pratica, le funzionalità che un sistema IT deve rendere disponibile ad un utente
- Le componenti "concrete" dei sistemi
- I dati in esse contenuti
- Le azioni da esse svolte
- Le relazioni che fra loro intercorrono (il processo in cui esse operano).

E' stato rilasciato nel 2005 lo standard 2.0, anche se molti strumenti informatici di ausilio all'analisi UML si rifanno ancora allo standard 1.5 o anche addirittura alla versione 1.3. Sono poi state realizzate estensioni non standard di UML, come quelle dei class diagram di Microsoft.
I diagrammi che fanno parte di UML 2.0 sono divisbili in tre gruppi.

- **Diagrammi "comportamentali" (Behaviour Diagram)**:
    - **Diagramma dei casi d'uso** o **Use Case Diagram**, che definisce le fasi di interazione fra un operatore umano, che identifica un ruolo, ed uno strumento, oppure fra due strumenti o apparati automatici; lo strumento o il sistema oggetto dell'analisi viene visto come black box, ossia vengono definite con precisione le azioni che esso compie, senza specificare come esso le compie;
    - **Diagramma di Attività** o **Activity Diagram**, che definisce la successione di attività compiute dal sistema oggetto di analisi; questi due diagrammi possono essere combinati efficacemente tra loro, come sarà spiegato nel capitolo 3;
    - **Diagramma di Stato** o **Statechart Diagram**, che descrive la successione di stati attraverso cui un elemento passa; l'elemento può essere un singolo oggetto o un intero sotto-sistema.

- **Diagrammi "strutturali" (Structure Diagram)**, tra cui gli ultimi due sono anche indicati come diagrammi di implementazione:
    - **Diagramma delle Classi** o **Class Diagram**, che identifica le entità coinvolte nel funzionamento del sistema ed i loro legami logici; un derivato da questo è il **Diagramma degli Oggetti** od **Object Diagram**, che definisce legami logici non fra entità astratte come persone o impiegati generici, ma tra entità concrete, chiaramente identificate;
    - **Diagramma delle strutture composite** o **Composite Structure Diagram** (introdotto nella versione 2.0);

- **Diagramma dei Package** o **Package Diagram** (introdotto nella versione 1.5, ma standardizzato solo nella versione 2.0);
- **Diagramma dei Componenti** o **Component Diagram**, che descrive il sistema suddiviso in singoli componenti;
- **Diagramma di Distribuzione** o **Deployment Diagram**, che mostra l'architettura "fisica" del sistema, descrivendo, ad esempio, i computer e i vari dispositivi presenti, le varie connessioni che intercorrono tra di essi e, ancora, i componenti software installati su ogni macchina.

- **Diagrammi "comportamentali di interazione" (Interaction Diagram)**:
  - **Diagramma di Sequenza** o **Sequence Diagram**, che esprime l'interazione dinamica fra le entità individuate dal Class Diagram privilegiando la sequenza temporale dello scambio di messaggi;
  - **Diagramma di Comunicazione** o **Communication Diagram** (chiamato in UML 1.5 Diagramma di Collaborazione o Collaboration Diagram), che esprime legami di interazione dinamica fra le entità individuate dal Class Diagram privilegiando le comunicazioni;
  - **Diagramma dei Tempi** o **Timing Diagram**, che esprime le azioni ed i cambiamenti di stato interni al sistema proiettate su un asse dei tempi
  - **Diagramma di Sintesi dell'Interazione** o **Interaction Overview Diagram**

L'insieme di tutti i diagrammi viene rappresentato graficamente con la sintassi dei class diagram in fig. 4.22.
L'obiettivo dei vari diagrammi è quello di costruire molteplici viste, tutte correlate tra di loro, del sistema oggetto della progettazione, ciascuna delle quali focalizza l'attenzione su un punto diverso.
E' importante osservare che lo standard UML definisce la sintassi e la semantica per la rappresentazione, ma non prescrive una sequenza di realizzazione dei diversi diagrammi e offre un'ampia gamma di possibilità di uso, tra cui i progettisti software possono scegliere. I diagrammi possono quindi essere usati in vari modi entro i diversi standard di progettazione, ad esempio si possono individuare i seguenti usi:

- **Use Case Diagram**: per capire nei dettagli "cosa" il sistema deve fare
- **Activity/Statechart Diagram**: per definire i processi fondamentali, ovvero mettere in ordine cronologico i casi d'uso definiti dagli Use Case Diagram
- **Class/Object Diagram**: per definire le entità fondamentali coinvolte nel funzionamento del sistema
- **Communication Diagram**: per definire le interazioni fra le entità fondamentali
- **Sequence Diagram**: per definire la sequenza delle interazioni fra entità
- **Component Diagram**: per definire nei dettagli quali parti compongono il sistema/prodotto software finito e le loro relazioni
- **Deployment Diagram**: per definire dove le varie parti di un programma devono essere poste in un'architettura distribuita (es. client-server)

Nei paragrafi successivi saranno presentate le sintassi ed i particolari relativi a ciascuno dei diagrammi.

Use Case Diagram

Negli Use Case Diagram (diagrammi dei casi d'uso), vengono individuati i casi d'uso principali del sistema, che vengono poi descritti nei dettagli come successioni di interazioni fra un **attore** (che può identificare un operatore umano con un preciso ruolo o un sistema esterno) ed il **sistema** oggetto dell'analisi che dovrà essere realizzato. Il dettaglio dei singoli casi d'uso diviene quindi una successione di interazioni elementari (**richieste**) compiute dall'operatore, soggetto attivo della situazione, e dalle conseguenti **risposte** del sistema.

Vediamo ora di definire in modo più preciso cosa rappresenta un caso d'uso ed il relativo diagramma. In modo formale possiamo definire un caso d'uso o use case come: una **sequenza di transazioni**, eseguita da un **attore** in interazione col **sistema**, la quale fornisce un **valore misurabile** per l'attore. Risulta chiaro quindi il legame fra il caso d'uso e l'attività.

Lo use case è in pratica un contratto, che descrive l'interazione fra due entità che interagiscono fra loro (ossia l'attore e il sistema), consentendo di stabilire con precisione:

- Servizi richiesti dall'attore
- Servizi forniti dal sistema
- Utenti abilitati a effettuare determinate richieste e quindi ad ottenere determinati servizi
- Vincoli nell'erogazione dei servizi stessi.

Per individuare un caso d'uso il primo passo è quello di trovare un confine preciso (boundary) per il sistema/sottosistema/componente che si sta analizzando (e questo dovrebbe risultare dall'analisi dell'attività che si sta "aprendo" per arrivare al caso d'uso). Una volta definito il confine si può stabilire cosa fa il sistema rispetto all'esterno e identificare attori e use case. In pratica quindi un caso d'uso è

- Una sequenza di transazioni compiute dall'attore in dialogo col sistema
- Comporta sempre uno o più attori
- Rappresenta nei dettaglio **cosa** il sistema offre all'attore, non come ciò viene realizzato
- Dovrebbe essere mappato alle attività di business.

Uno use case rappresenta una situazione tipica di utilizzo del sistema e comprende in sé vari flussi possibili di esecuzione. Tale situazione rappresenta un'importante parte di funzionalità e dovrebbe essere completa dall'inizio alla fine.

Un attore rappresenta un'entità esterna al sistema (una persona, un altro sistema software, un componente hardware) che interagisce col sistema. Un attore individua un ruolo piuttosto che un'entità fisica, per individuare il quale si possono usare le domande seguenti:

- chi ha bisogno del sistema?
- chi userà le funzionalità principali?
- chi dovrà manutenere e amministrare il sistema?
- di quali dispositivi (fisici, hardware, software...) il sistema ha bisogno?
- con quali altri sistemi il sistema dovrà comunicare?

Poi, per ognuno degli attori precedentemente identificati, si può rispondere alle seguenti domande:

- quali funzioni l'attore richiede al sistema?
- l'attore ha bisogno di leggere o scrivere o immagazzinare informazioni nel sistema?
- l'attore deve ricevere notifiche di eventi dal sistema?

Attori e use case sono sempre collegati fra loro, un attore isolato non può interagire col sistema, mentre uno use case isolato non fornisce alcuna funzionalità all'esterno (intendendo funzionalità che abbiano un senso all'esterno).
Un attore può essere:

- **attivo**, ovvero inizia uno use case
- **passivo**, ovvero partecipa a uno use case, ma non lo inizia.

Un esempio di attore attivo è l'impiegato che immette i dati in un gestionale, mentre un esempio di attore passivo è l'operatore di cassa di un supermercato che partecipa al caso d'uso acquisto degli articoli, che però è iniziato dal cliente.
Vediamo ora la sintassi grafica degli Use Case Diagram. Nella fig. 4.1 viene rappresentato un caso d'uso generico, ove l'attore è l'omino stilizzato, il sistema è esplicitato con un rettangolo e il caso d'uso con un'ellisse. Il caso d'uso nelle fasi successive di analisi viene espanso nella sequenza delle transazioni, ovvero di singole azioni od operazioni, non ulteriormente scomponibili, che lo compongono.
L'uso dei diagrammi serve, una volta identificati i singoli casi d'uso, a esplicitare sia le relazioni con gli attori, sia soprattutto eventuali relazioni di dipendenza o successione temporale che esistano tra i vari casi d'uso. Poiché tali relazioni non vengono esplicitate chiaramente dagli Use Case Diagram, esiste una metodologia di IBM che prevede di passare dallo Use Case Diagram ad un corrispondente Activity Diagram, dove le singole attività corrispondono ai singoli casi d'uso e la successione dei passaggi da un'attività all'altra individua la navigazione cronologica tra i casi d'uso, ovvero tra le maschere di interfaccia utente che da essi hanno origine durante l'implementazione. In tal modo si può costruire quindi il diagramma di navigazione fra le finestre del programma, come sarà mostrato negli esempi del prossimo capitolo.

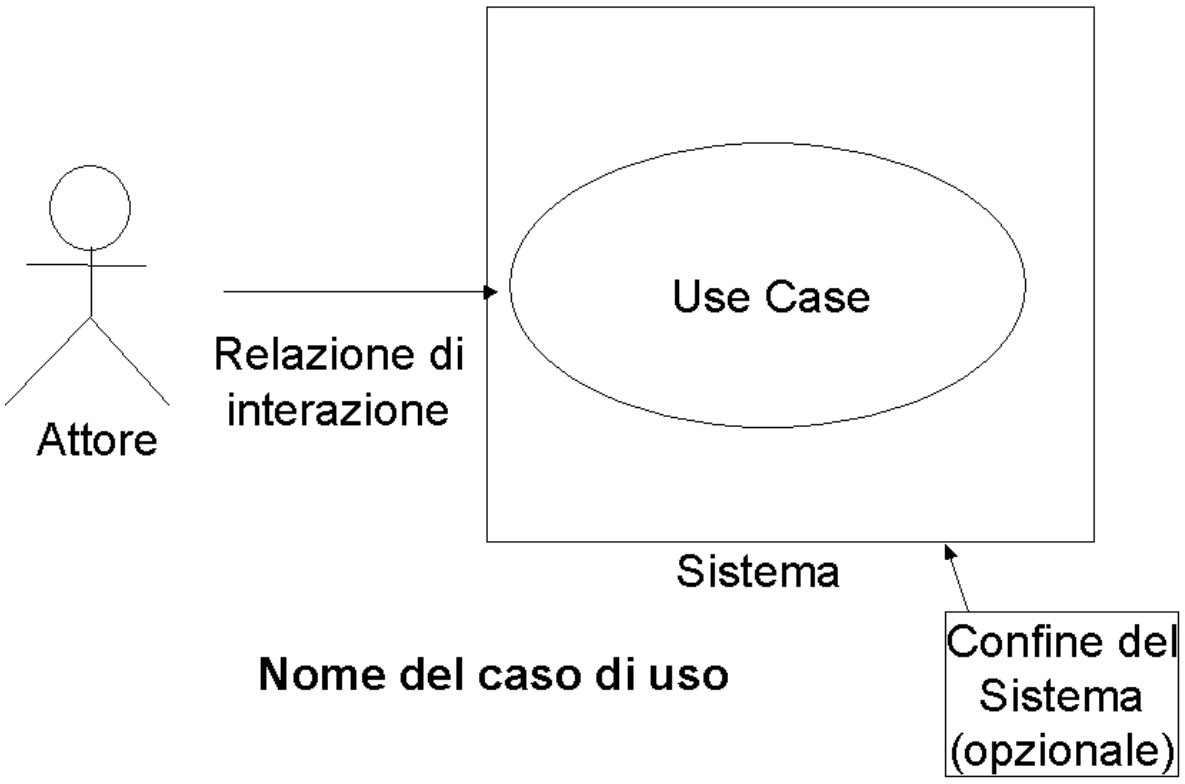


Fig. 4.1: Gli elementi basilari di un Use Case Diagram. Si noti la interazione fra l'attore ed il singolo caso d'uso, che può essere indicata con la freccia o con la semplice linea.

La relazione di ***generalizzazione***, mostrata in fig. 4.2, collega un attore o caso d'uso ad un altro più generale. Il derivato specializza il genitore aggiungendovi nuove caratteristiche.
La relazione di ***include***, mostrata in fig. 4.3, esprime una dipendenza tra casi d'uso; il caso incluso fa parte del comportamento di quello che lo include. L'inclusione non è opzionale ed avviene in ogni istanza del caso d'uso. La corretta esecuzione del caso d'uso che include dipende da quella del caso d'uso incluso. Non si possono formare cicli di include. In pratica rappresenta graficamente l'azione della WBS sull'insieme di interazioni che il caso d'uso racchiude. Di solito la relazione di include viene usata per riutilizzare parti comuni a più casi d'uso.
La relazione di ***extend***, mostrata in fig. 4.4, esprime una diversa dipendenza tra casi d'uso (notare il verso della freccia, che va dal caso che estende a quello esteso). Il caso d'uso che estende (client) specifica un incremento di comportamento a quello esteso (supplier). Si tratta di comportamento supplementare ed opzionale che si verifica in casi particolari o non standard.
L'extend è diverso da una generalizzazione tra casi d'uso: in una generalizzazione, entrambi i casi d'uso sono ugualmente significativi, mentre, in un extend, il client non ha necessariamente senso se preso da solo.
Un caso d'uso raggiunto da almeno un extend può opzionalmente visualizzare i propri extension points, ossia i punti e/o condizioni dell'esecuzione in cui il comportamento viene esteso. Nella fig. 4.5 è mostrato un esempio, in cui la registrazione del cliente deve avere luogo nella condizione "cliente non (ancora) registrato", descritta dall'extension point.
Riassumendo:

- Include specifica comportamento *obbligatorio*, mentre extend specifica comportamento *supplementare* (varianti).
- Nell'include la freccia va dal caso d'uso che include verso quello incluso.
- Nell'extend la freccia va dal caso d'uso che estende verso quello esteso.
- Sono entrambi costrutti utili, ma non se ne deve abusare o i diagrammi diventano illeggibili.

In fig. 4.6 è mostrato un esempio completo con uso delle relazioni viste.

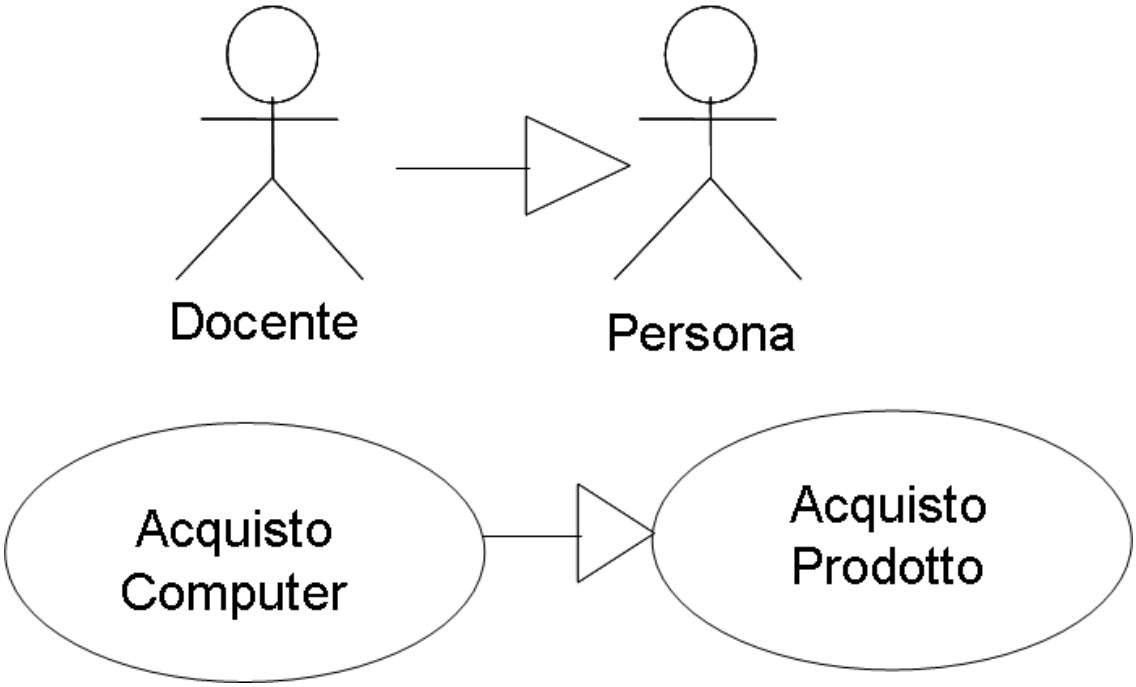


Fig. 4.2: relazione di generalizzazione fra due attori e due casi d'uso.

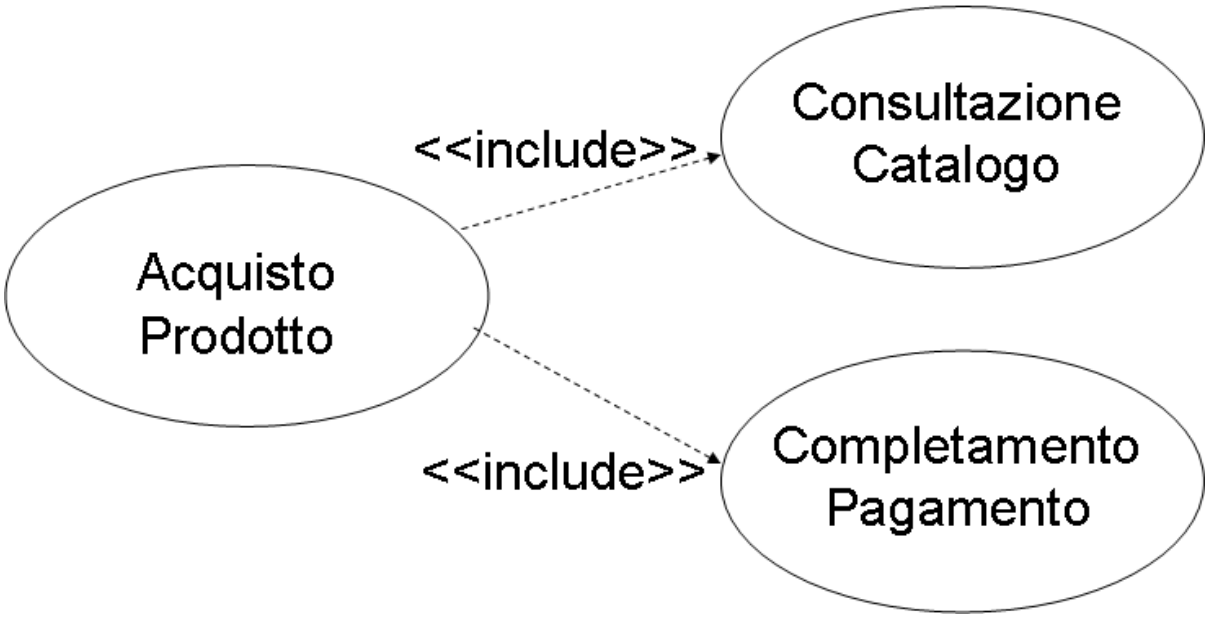


Fig. 4.3: relazione di include fra due casi d'uso.

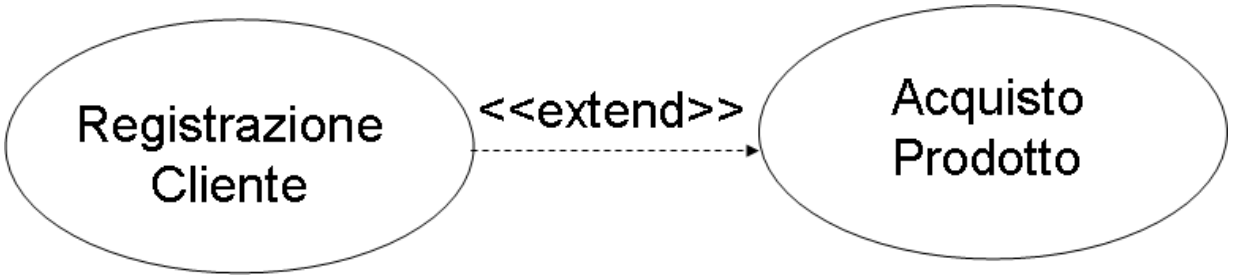


Fig. 4.4: relazione di extend fra due casi d'uso.

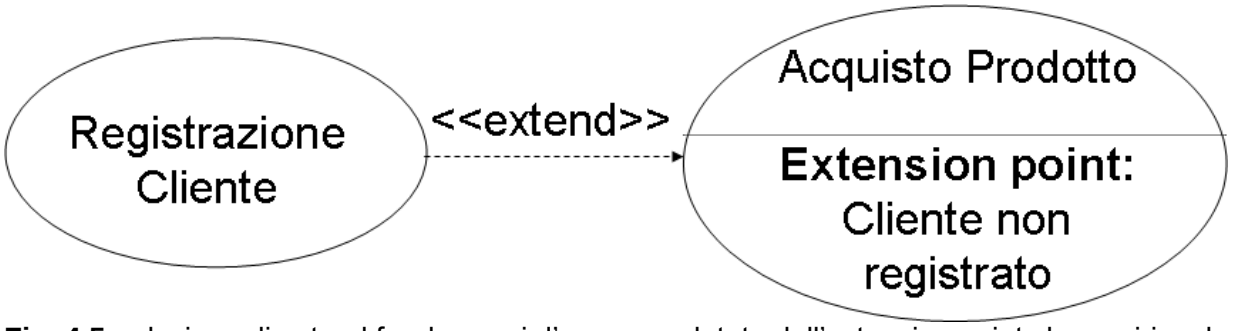


Fig. 4.5: relazione di extend fra due casi d'uso, completata dall'extension point che espirime la condizione che causa l'estensione.

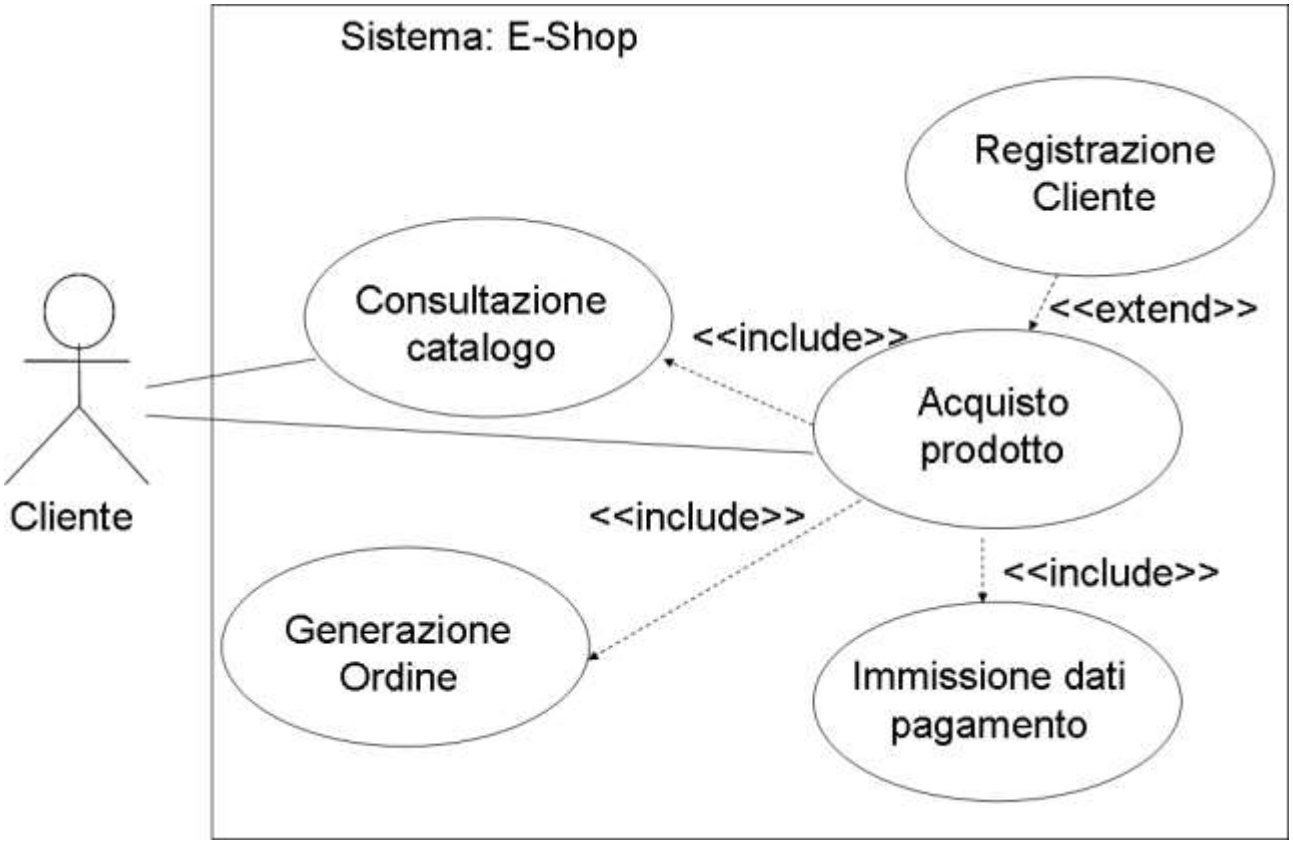


Fig. 4.6: esempio di casi d'uso di un sistema E-Shop collegati fra loro da relazioni di include ed extend.

I nomi dei casi d'uso vanno scelti con cura per poter essere significativi. Le buone pratiche prevedono di usare un sostantivo descrivente una macro-azione "riassumente" il caso d'uso, come negli esempi riportati, oppure direttamente un verbo all'infinito, relativo alla macro azione compiuta (ad esempio, "acquisto prodotto" o "acquistare prodotto").
I casi d'uso sono molto importanti ed aiutano nella WBS dei progetto di un sistema, ma non sono sufficienti alla sua descrizione completa. Per questo vengono completati dalle descrizioni che saranno trattate con gli esempi nel prossimo capitolo.

Activity Diagram

Nei diagrammi di attività l'enfasi viene posta sulle attività, ma senza specificare di solito chi le compie, o comunque come le compie, ossia anche se sono presenti come oggetti le rappresentazioni di attori, non viene esplicitata l'azione compiuta dagli attori, ovvero l'interazione fra più entità presenti allo scopo di realizzare l'attività.
Un processo è formato da sottoprocessi o fasi, formate da attività, a loro volta formate da azioni od operazioni (si veda [Destri 2013]). L'Activity Diagram evidenzia le componenti del processo o dell'attività sotto forma di successione logico-temporale di azioni, eventualmente riportando entità usate o modificate nelle azioni stesse, ma sempre da un punto di vista di successione di azioni. Le azioni sono viste in primo piano come componenti del processo, senza evidenziare chi le compie o come vengono compiute.
In modo più preciso, possiamo dire che gli Activity Diagram:

- Sono una evoluzione dei diagrammi di flusso o flow-chart

- Rappresentano una procedura o un workflow mostrando l'evoluzione di un flusso di attività
- Ogni attività è definita come un'evoluzione continua, non necessariamente atomica, di uno stato

Gli elementi di base degli Activity Diagram sono i seguenti.

- **Activity (Attività)**: Esecuzione non atomica entro un sistema dotato di stati. Può essere scomposta in azioni.
- **Action (Azione)**: Operazione atomica eseguibile che produce come risultato un cambiamento nello stato di un sistema o il ritorno di un valore.
- **Action State (Stato di azione)**: Uno stato che rappresenta l'esecuzione di un'azione (atomica), tipicamente l'invocazione di una operazione.
- **Activity State (Stato di attività)**: Stato composito, in cui il flusso di controllo è formato di altri stati di attività e stati di azione. Non è atomico, il che significa anche che può essere interrotto, rimanendo "congelato" in un certo punto della sua evoluzione. Può anche essere ulteriormente scomposto in altri diagrammi di attività.
- **Transition (Transizione)**: Rappresenta il flusso di controllo fra due attività, che mostra il percorso da un action o activity state al successivo action o activity state.
- **Object Flow (Flusso di oggetti)**: Rappresenta uno o più oggetti (un'entità) coinvolti nel flusso di controllo associato con un activity diagram.
- **Object State (Stato di oggetto/i)**: Una condizione o situazione operativa nella vita di un oggetto (un'entità) durante la quale l'oggetto soddisfa certe condizioni, compie certe attività o attende certi eventi.
- **Swimlane** o **Corsia**: Una suddivisione per l'organizzazione di responsabilità per le attività. Non ha un significato fisso, ma spesso corrisponde alla unità organizzativa entro un business model (es. ufficio acquisti, vendite...). Viene rappresentata graficamente con una linea tratteggiata che divide in sezioni il diagramma.

I simboli grafici elementari che compongono un diagramma di attività sono riportati in figura .

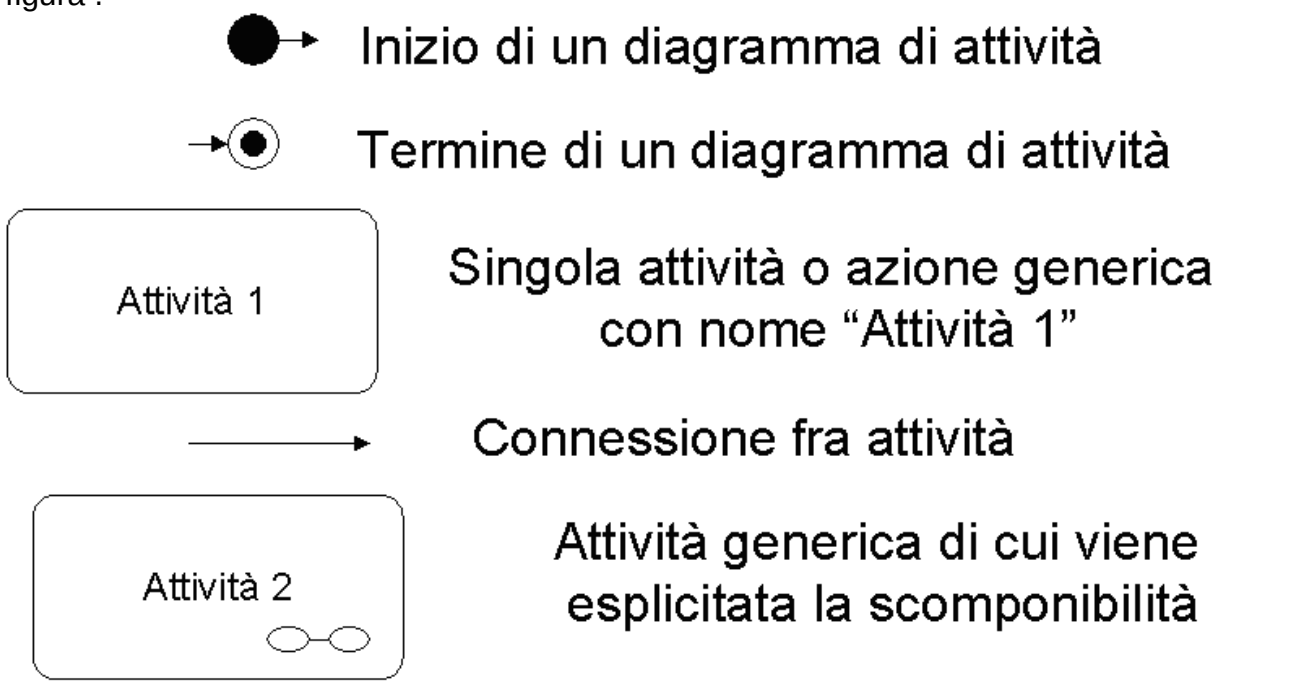


Fig. 4.7: Gli elementi costitutivi fondamentali di un diagramma di attività. Eventuali oggetti (entità) sono rappresentati come rettangoli con gli spigoli normali (non arrotondati). La connessione indica normalmente legame di successione temporale. Azioni ed attività possono essere indicate con lo stesso simbolo.

A livello macroscopico gli Activity Diagram possono individuare i processi e la scomposizione delle singole attività può dare origine a insiemi di casi d'uso. A livello più dettagliato invece si passa da uno Use Case Diagram al corrispondente Activity per mettere in evidenza la successione cronologica delle attività corrispondenti ai singoli casi d'uso.
Come nei flow-chart o diagrammi di flusso da cui i diagrammi di attività derivano, i legami di sequenza temporale che legano fra di loro le attività in un diagramma possono essere di vari tipi. Il più semplice è senza dubbio la sequenza semplice, rappresentata in fig. 4.8, in cui due attività sono semplicemente consecutive.
Altri tipi di legame logico-temporale sono rappresentati nelle fig. da 4.9 a 4.13, dove troviamo percorsi di biforcazione (in base ad una condizione il processo prosegue con attività diverse), punti di attesa e sincronizzazione (finché non sono state completate tutte le attività collegate non può avere luogo l'attività successiva a tutte) ed iterazioni, ossia ripetizioni multiple della stessa attività.

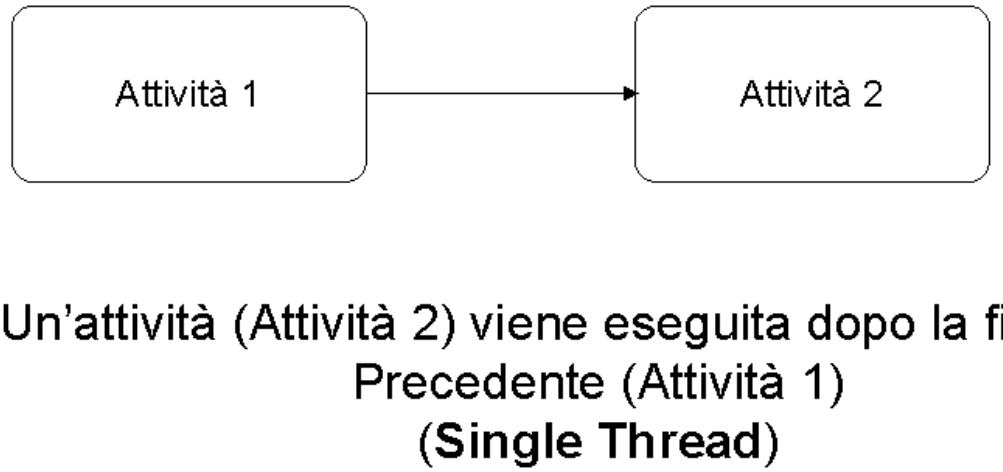
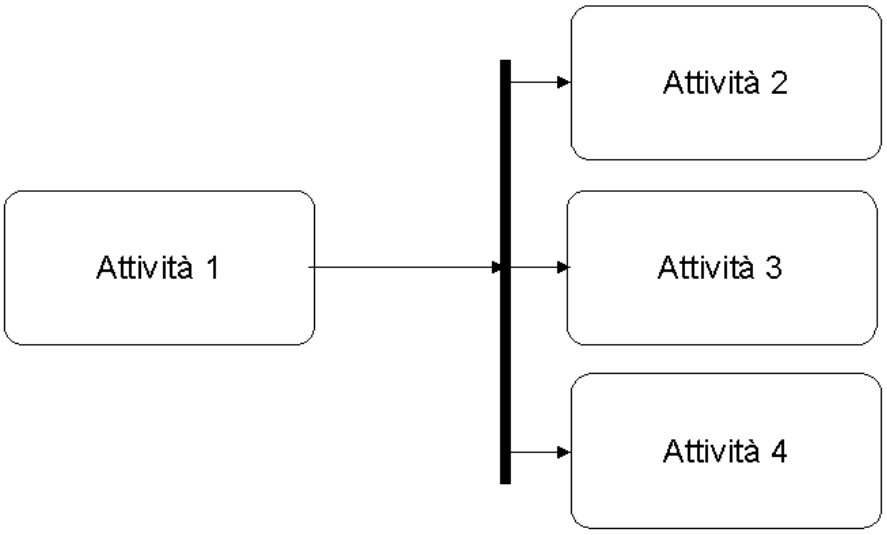
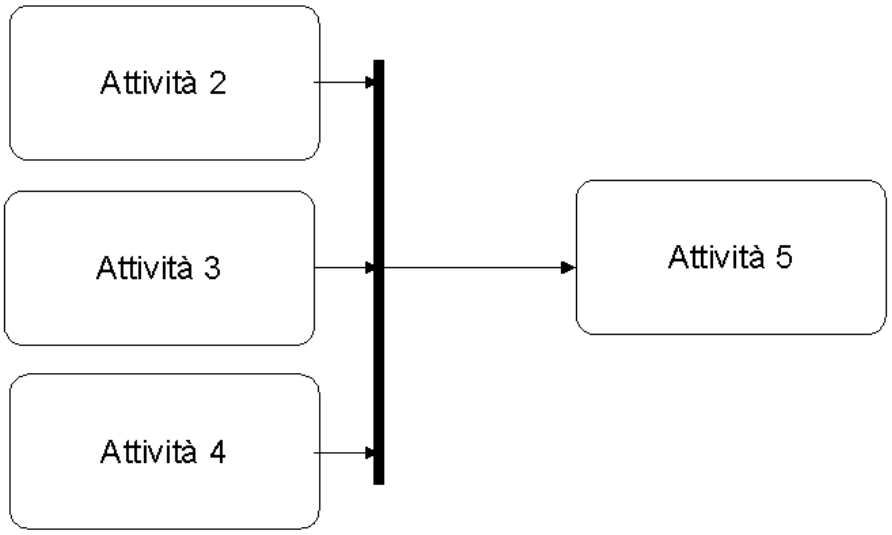


Fig. 4.8: L'attività 2 viene eseguita dopo la fine della precedente attività 1.



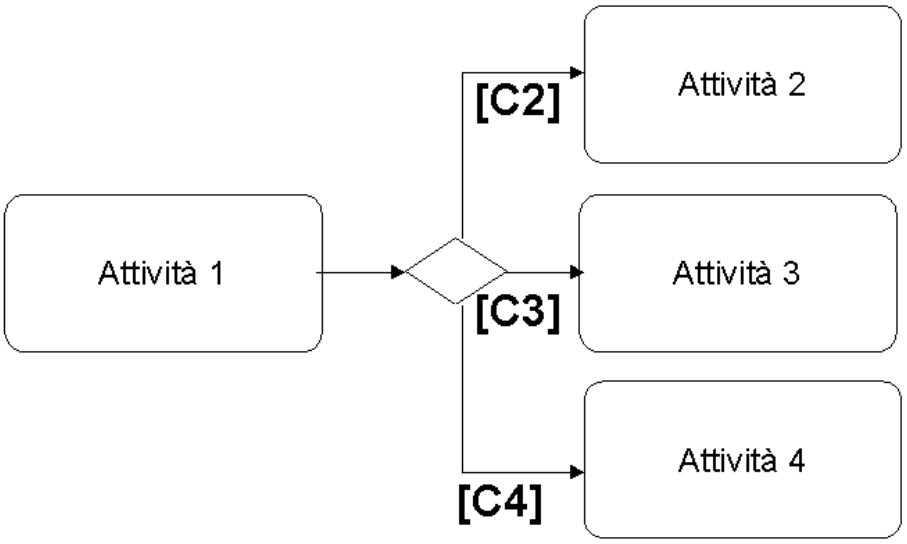
Un singolo flusso di attività si divide in più flussi, consentendo l'esecuzione simultanea di più attività
(**Multiple Thread**)

Fig. 4.9: And-split.



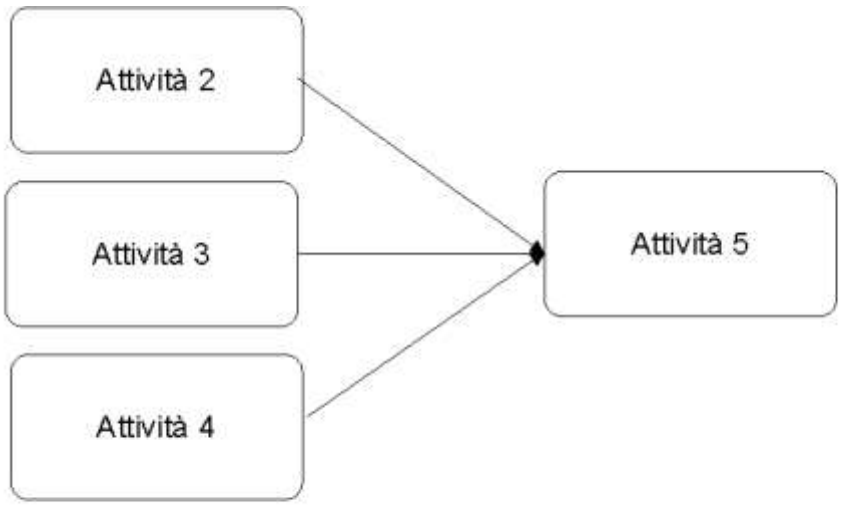
Due o più flussi di attività convergono in uno solo
E' un punto di sincronizzazione per il workflow: non si va avanti finché non sono terminate tutte le attività precedenti

Fig. 4.10: And-join.



Un singolo flusso di attività prosegue per uno dei cammini in base al verificarsi delle condizioni di transizione, indicate fra parentesi quadre

Fig. 4.11: Or-split.



Un punto dove due o più flussi di attività ri-convergono in uno solo ovvero hanno tutti Attività 5 come elemento successivo

Fig. 4.12: Or-join.

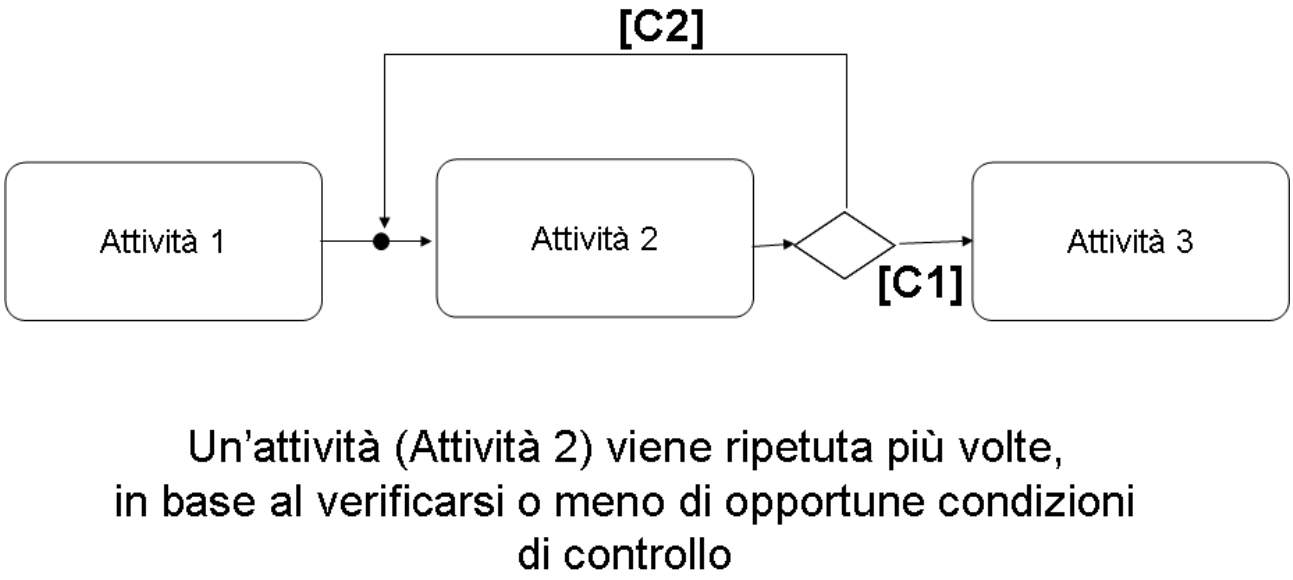


Fig. 4.13: Iterazione.

Class ed Object Diagram

I **Class Diagram** identificano le entità e le relazioni o associazioni che fra esse intercorrono. Le entità rappresentano essenzialmente il modello formale associato al concetto da esse espresso (ad esempio, le persone, i prodotti, gli ordini, le fatture), ottenuto attraverso un processo di astrazione dalla realtà. Le associazioni identificano i rapporti che legano le entità, comprensivi dei legami numerici, definiti dalle molteplicità (ad esempio un auto ha 4 ruote, un'azienda ha molti dipendenti, un'auto ha in un determinato momento un solo proprietario).
Talvolta si rende necessario indicare non entità astratte, ma istanze concrete di tali entità (ad esempio non un docente generico, ma il professor Giulio Destri, non un dirigente generico, ma il direttore del personale Dr. Guido Ventura, non un prodotto generico, ma il sapone Mantovani).
I diagrammi relativi a questo caso particolare si definiscono Object Diagram, sono simili ai Class Diagram, ma per ogni entità concreta al loro interno, definiscono il nome univoco dell'oggetto (ovvero dell'oggetto del mondo reale modellato) e la sua classe di appartenenza (ossia la categoria astratta cui esso appartiene, definita con il nome della classe di appartenenza).
Per definire i Class Diagram è necessario anche definire la terminologia del sistema reale che si sta analizzando, identificando con precisione le entità (persone, ruoli, luoghi, oggetti materiali, eventi, strutturazioni ecc...) coinvolte nel sistema del mondo reale (ovvero del dominio di business) che hanno importanza per il sistema informativo che lo dovrà gestire e per le componenti informatiche al suo interno. La terminologia viene definita attraverso un documento specifico: il **glossario**, che verrà presentato con esempi nel prossimo capitolo.
I Class Diagram sono una evoluzione dei diagrammi Entità-Relazione e vengono usati anche per progettare le Basi di Dati atte a contenere e rendere persistenti (ossia permanenti sino a che non vengono cancellati esplicitamente) i dati aziendali che compongono l'informazione entro l'azienda.
Nei diagrammi delle classi i componenti elementari sono le classi stesse, rappresentate come rettangoli se identificano elementi del mondo reale, anche astratti. Le classi possono essere rappresentate a vari livello di dettaglio, come mostrato nella fig. 4.14, dove

vediamo le classi come semplici elementi senza una struttura esplicitata, le classi come appartenenti a categorie, rappresentate dagli stereotipi, la classi con esplicitati i propri attributi, ovvero elementi caratterizzanti, e con esplicitati anche i metodi, ovvero i "servizi", le azioni che le entità che le classi rappresentano sono in grado di compiere dietro indicazioni provenienti dall'esterno.
I diagrammi delle classi esprimono anche relazioni di tipo logico, di dipendenza, di derivazione e di inclusione, attraverso opportune simbologie. Il legame generale che due classi possono avere tra loro prende il nome di **associazione**. Un esempio di associazione è mostrato in fig. 4.15.
Accanto all'associazione semplice troviamo l'associazione con **molteplicità**, che indica un valore numerico, ossia una cardinalità associata ad un legame. Per esempio un'auto ha quattro ruote (più il ruotino di scorta), una bicicletta ha due ruote e così via; tale informazione viene indicata dai numeri vicini al legame, come indicato in fig. 4.16.
Qualora il legame espresso dall'associazione abbia delle caratteristiche esprimibili come attributi, o comunque la molteplicità dell'associazione sia molti-a-molti, può essere necessario introdurre il concetto di **classe di associazione**, come mostrato in fig. 4.17. La classe di associazione esprime e "da dignità di entità" ad un legame logico, come la proprietà di un auto, caratterizzata dalle date di inizio e date di fine. Nella stessa fig. 4.17 sono anche introdotti gli elementi UML **nota** (identificata dal simbolo grafico del post-it) e **vincolo** (post-it con l'aggiunta delle parentesi grafiche a racchiudere il testo. Questi elementi possono essere usati in qualunque diagramma UML.
Qualora si voglia esplicitare un legame di "inclusione" di una entità (o di un gruppo di entità) in un'altra, si possono usare i simboli di **aggregazione** e **composizione**, indicati in fig. 4.18, rispettivamente, con il rombo bianco (vuoto) e quello nero (peino). L'aggregazione esprime un legame di inclusione in cui però l'elemento incluso (indicato in fig. 4.18 come "Detail") esiste anche senza l'inclusione. Viceversa, nella composizione, l'elemento incluso non può esistere senza l'includente, per cui, in caso di cancellazione di quest'ultimo devono essere eliminati anche tutti gli elementi inclusi attraverso la composizione. Un esempio di uso combinato dei due è rappresentato in fig. 4.19: l'ordine include le righe ordine attraverso una composizione (quindi possiamo leggere: l'ordine è composto di righe ordine) e ciascuna riga ordine include un prodotto, con molteplicità uno o più (quindi aggrega una certa quantità di prodotti, ad esempio 10 telefonini). La cancellazione dell'ordine prevede anche la cancellazione delle righe ordine che lo compongono, ma non dei prodotti che esse aggregano.
Un altro tipo di legame logico è la **derivazione** o **ereditarietà**, rappresentati dal legame con il triangolo vuoto, che esprime la specializzazione di una classe in un'altra, ad esempio la persona (generica) si specializza nel dipendente che a sua volta si specializza nell'impiegato, nell'operaio e nel dirigente, come mostrato in fig. 4.20. Se percorsa in senso inverso, questo legame logico è invece una **generalizzazione**. La derivazione è comune ad altri diagrammi UML, come già visto per lo Use Case Diagram.

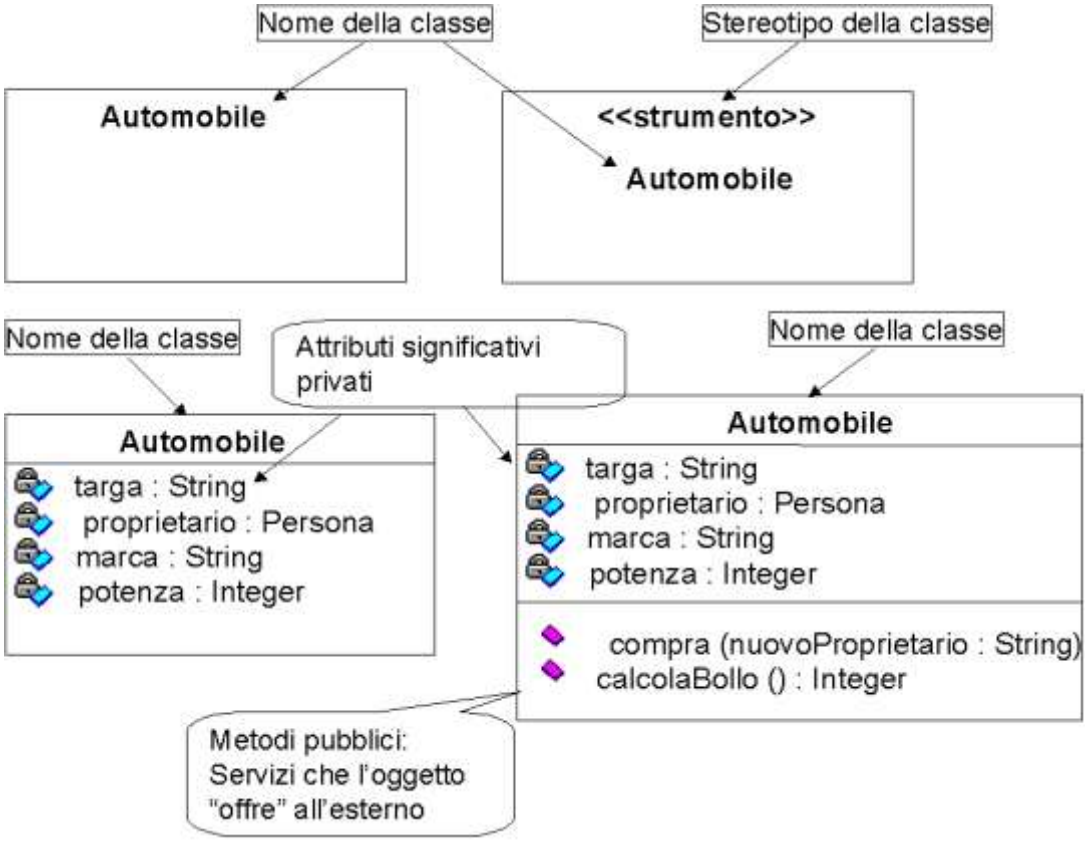


Fig. 4.14: Vari modi di rappresentare una classe entro un Class Diagram, a vari livelli di dettaglio.

Fig. 4.15: Esempio di associazione, l'impiegato è associato all'azienda per cui lavora.

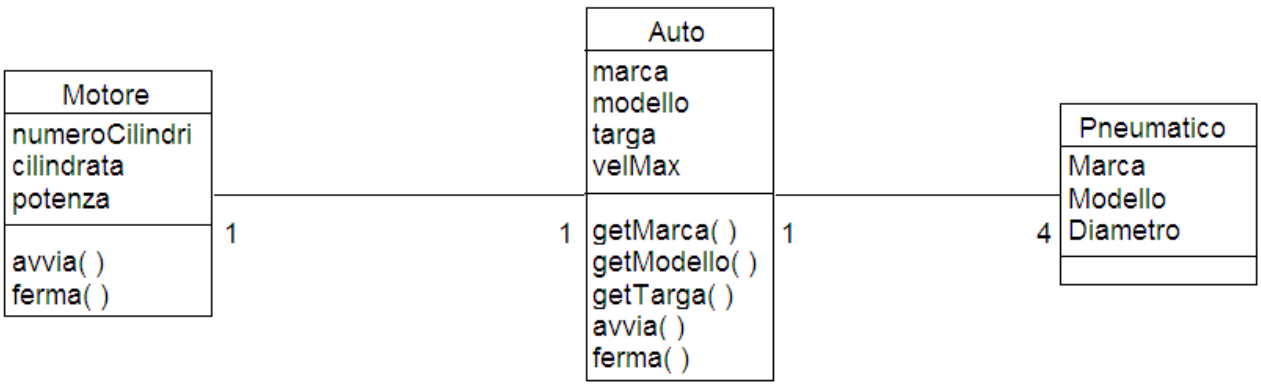


Fig. 4.16: Esempio di associazione con molteplicità. Un auto ha un motore e 4 pneumatici.

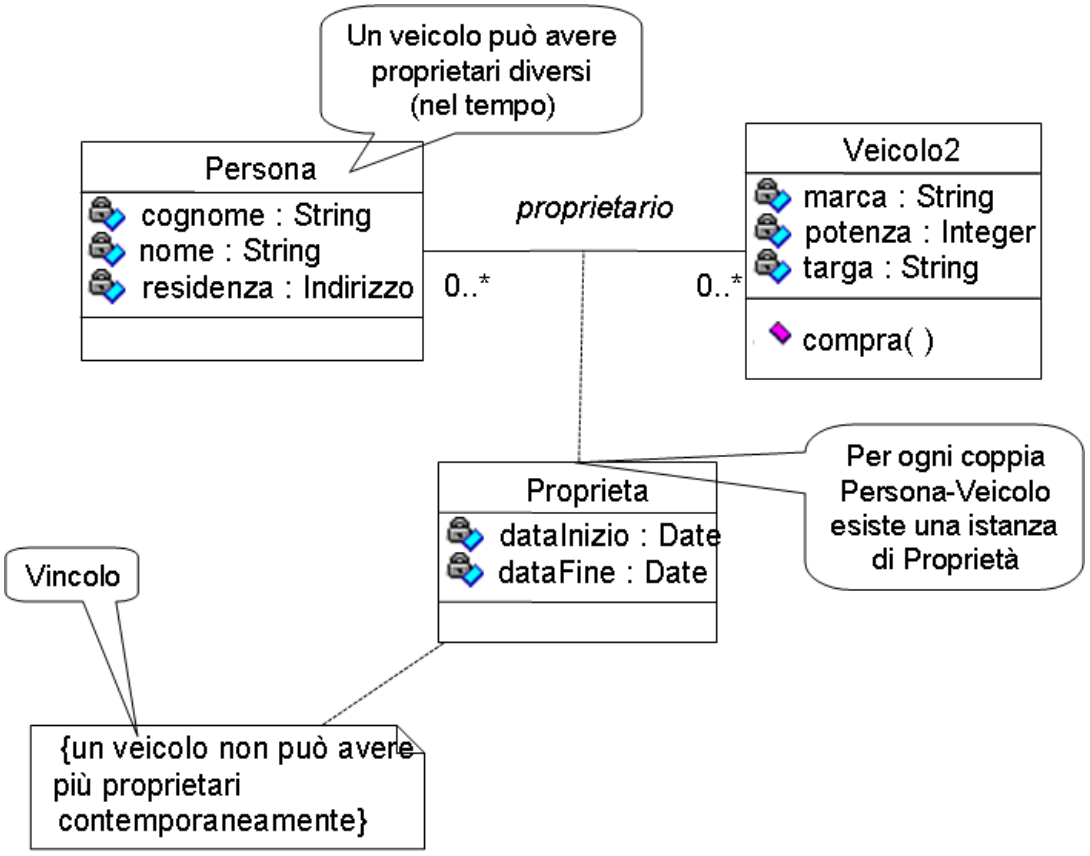


Fig. 4.17: Esempio di classe di associazione. Un veicolo, in un dato momento può avere un solo proprietario, ma nel tempo può averne più di uno. Ogni rapporto di proprietà, espresso dall'associazione, ha una durata finita e quindi una data di inizio ed una di fine. I simboli simili ai post-it rappresentano le note ed i vincoli.

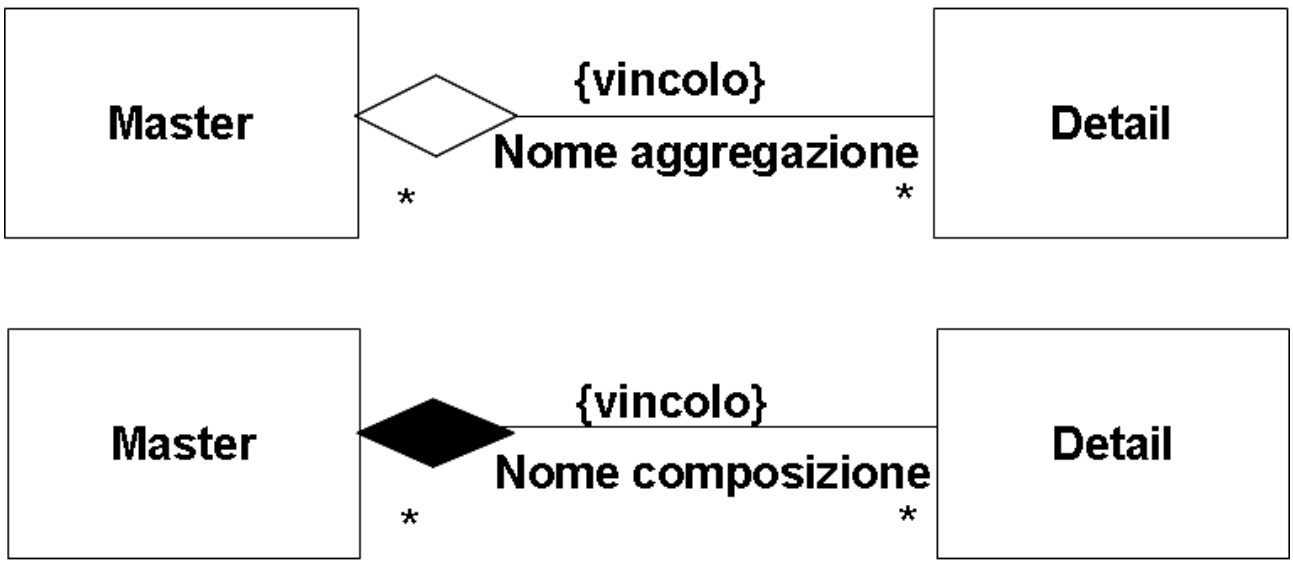


Fig. 4.18: Simbologia per aggregazione (rombo bianco in alto) e composizione (rombo nero in basso). L'eliminazione di una composizione elimina anche tutti i suoi elementi componenti, mentre l'eliminazione di una aggregazione invece no (i componenti hanno anche una natura indipendente).

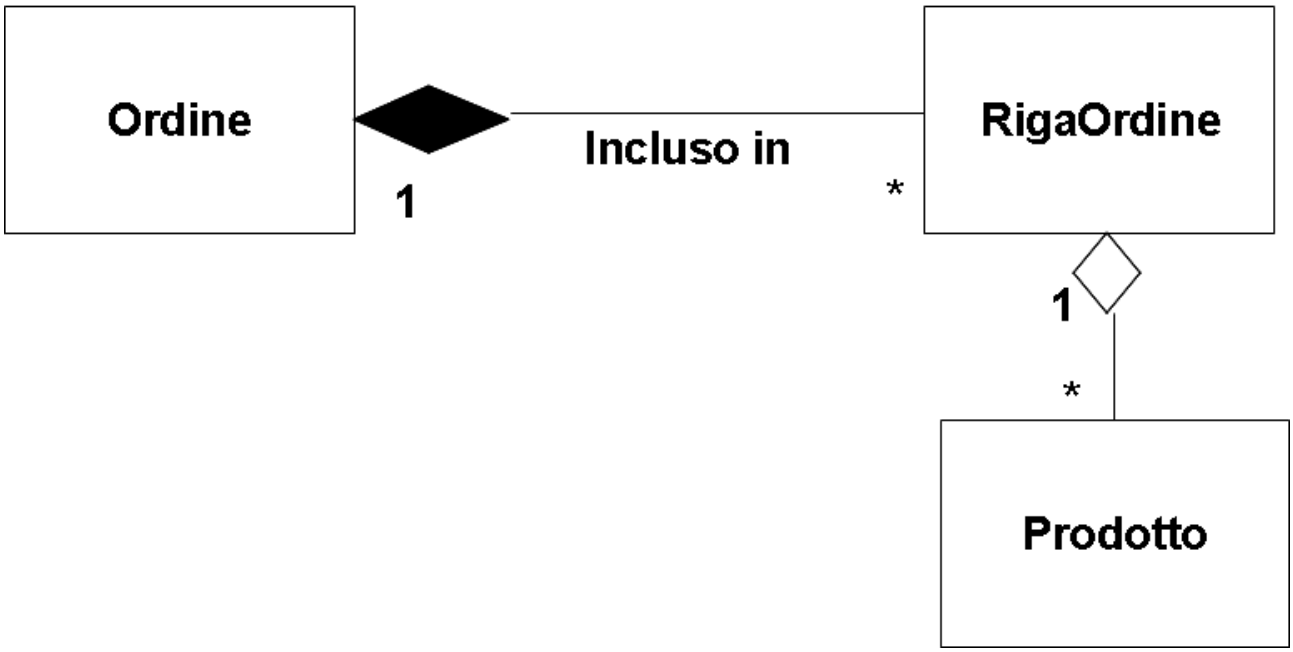


Fig. 4.19: Esempio di aggregazione e composizione combinate nello stesso diagramma. L'eliminazione di un ordine elimina anche le righe ordine ma non elimina i prodotti che esse aggregano.

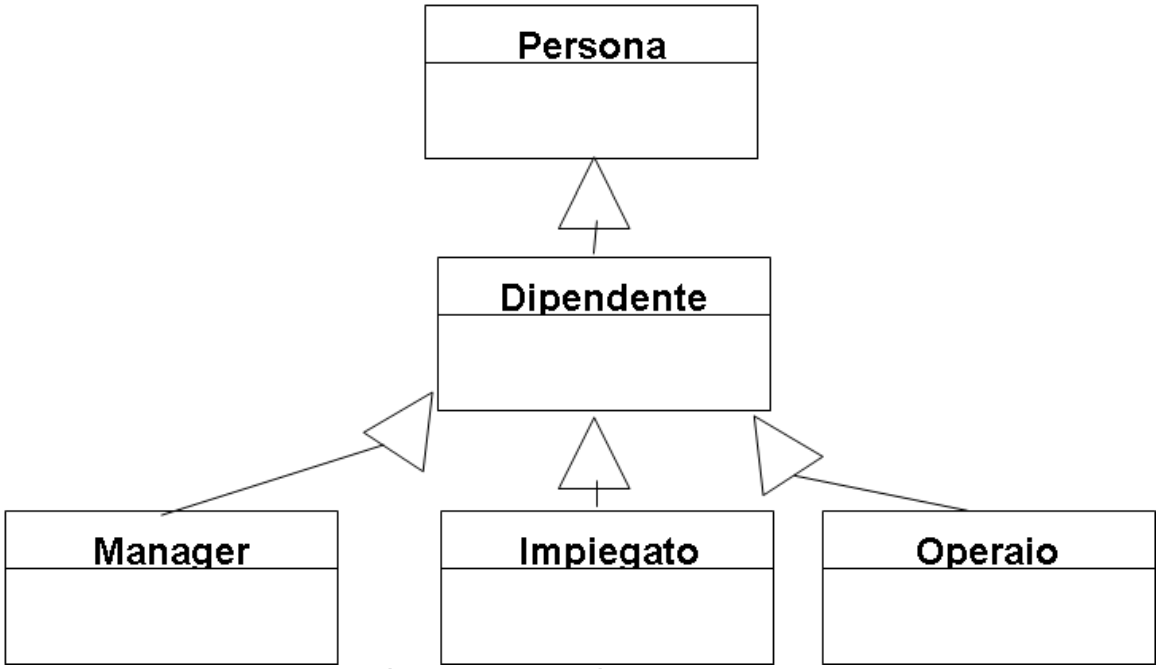


Fig. 4.20: Esempio di ereditarietà. La persona è una classe molto generica, da cui deriva il dipendente, che comprende come casi particolari le tre classi derivate manager, impiegato, operaio.

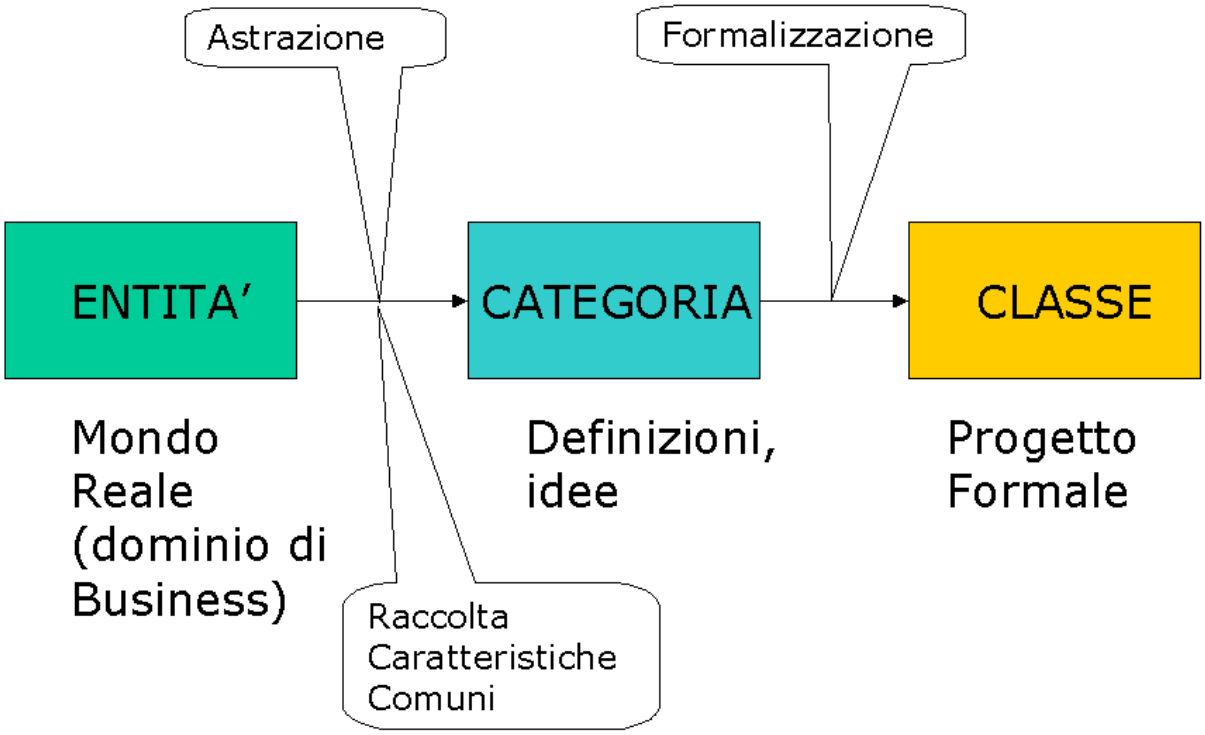


Fig. 4.21: Il percorso logico di analisi che conduce dalle entità alle classi.

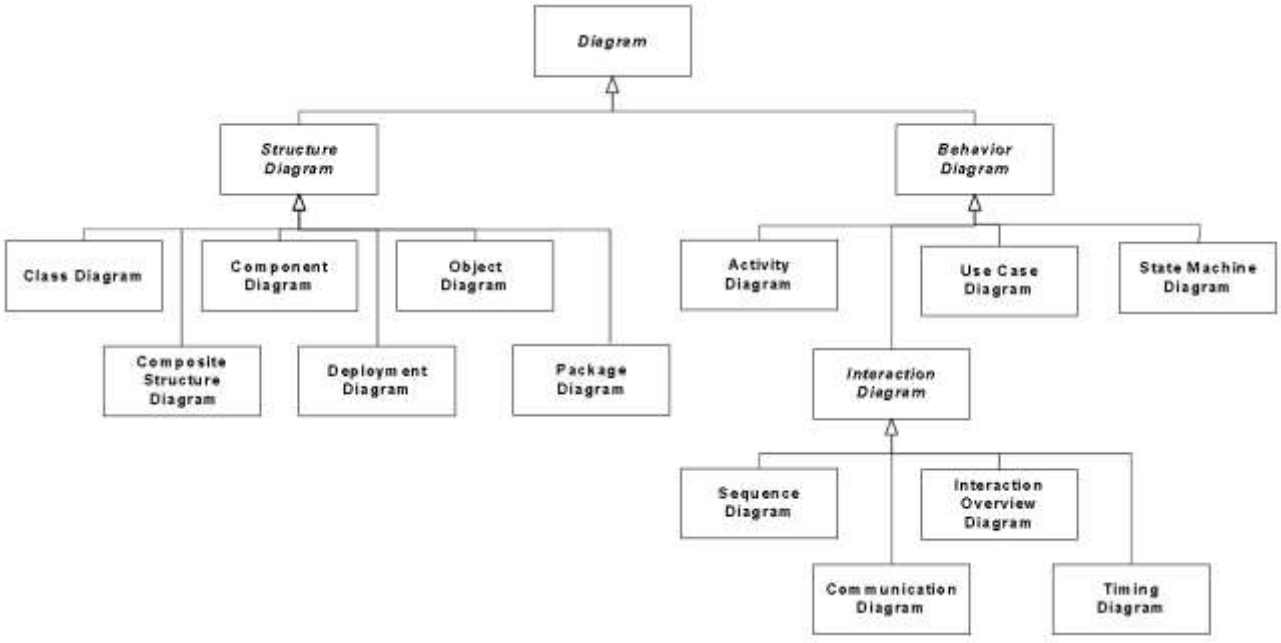


Fig.4.22: L'insieme dei diagrammi di UML 2.x visto come un class diagram, esprimente i legami di derivazione delle tre famiglie dal diagramma generico, e dei diagrammi specifici dalle famiglie.

Sequence e Communication Diagram

Un limite dei Class e Object Diagram è quello di esprimere solo legami “statici” fra entità, senza porre enfasi sulle interazioni dinamiche che fra esse avvengono. D’altronde, gli Use Case Diagram descrivono interazioni solo ad un livello molto elevato di astrazione, trascurando volutamente molte caratteristiche interne dei sistemi. Inoltre non rappresentano le entità su un piano paritetico, ponendo enfasi sul ruolo dell’attore. Per questo, per valutare con chiarezza un comportamento dinamico vengono aggiunti i diagrammi di interazione, di cui molto importanti sono i Sequence Diagram o diagrammi di sequenza e Communication Diagram o diagrammi di comunicazione, che definiscono le interazioni fra entità.

Le interazioni sono praticamente dei flussi di informazione che scorrono tra le classi e, normalmente, rappresentano comandi, ovvero invocazioni di servizio fatte da una classe all’altra, oppure trasferimento del controllo del processo dall’una all’altra classe o oggetto.

I due tipi di diagrammi sono semanticamente equivalenti, ma il diagramma di sequenza ordina temporalmente la sequenza dei messaggi e questo lo rende uno strumento migliore per la comprensione dei vincoli temporali, mentre il diagramma di comunicazione mette in evidenza i legami di dipendenza tra le classi, e quindi tra le entità che esse rappresentano, per l’esecuzione delle attività.

Gli elementi sintattici di base dei due diagrammi sono riportati nelle fig. 4.23 e 4.24. I Sequence Diagram sono formati da un insieme di oggetti sotto i quali stanno gli assi temporali diretti verso il basso. I messaggi che gli oggetti si scambiano sono espressi tramite le frecce e in corrispondenza alla durata dell’azione svolta dagli oggetti in rispondenza ai messaggi sta il rettangolo bianco sull’asse. Gli oggetti possono talvolta anche essere direttamente sostituiti dalle classi, indicando che in quel momento viene rappresentato il ruolo della categoria e non del singolo componente della categoria.

In fig. 4.25 è mostrato un diagramma di sequenza con due tipi di interazioni, la prima è asincrona e quindi il chiamante non attende risposta dal chiamato, mentre la seconda è sincrona e, al termine dell’esecuzione di quanto richiesto, il chiamato dovrà tornare un risultato al chiamante (valore di ritorno). In fig. 4.26 è mostrata l’interazione sincrona espressa sia con un diagramma di sequenza, sia con l’equivalente diagramma di comunicazione. In quest’ultimo è necessario esplicitare la sequenza dei numeri di messaggio, mancando l’elemento ordinante dell’asse temporale rivolto verso il basso.

Nel prossimo capitolo saranno mostrati esempi d’uso di questi diagrammi.

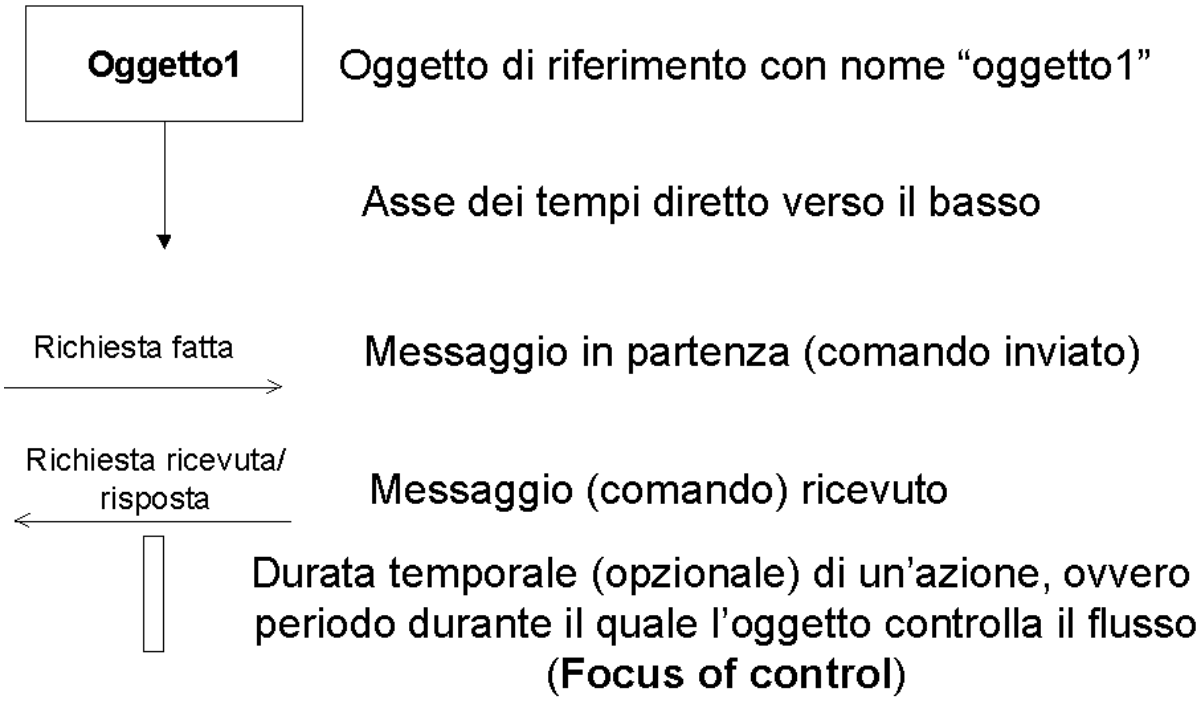


Fig. 4.23: Gli elementi base che formano i diagrammi di sequenza.

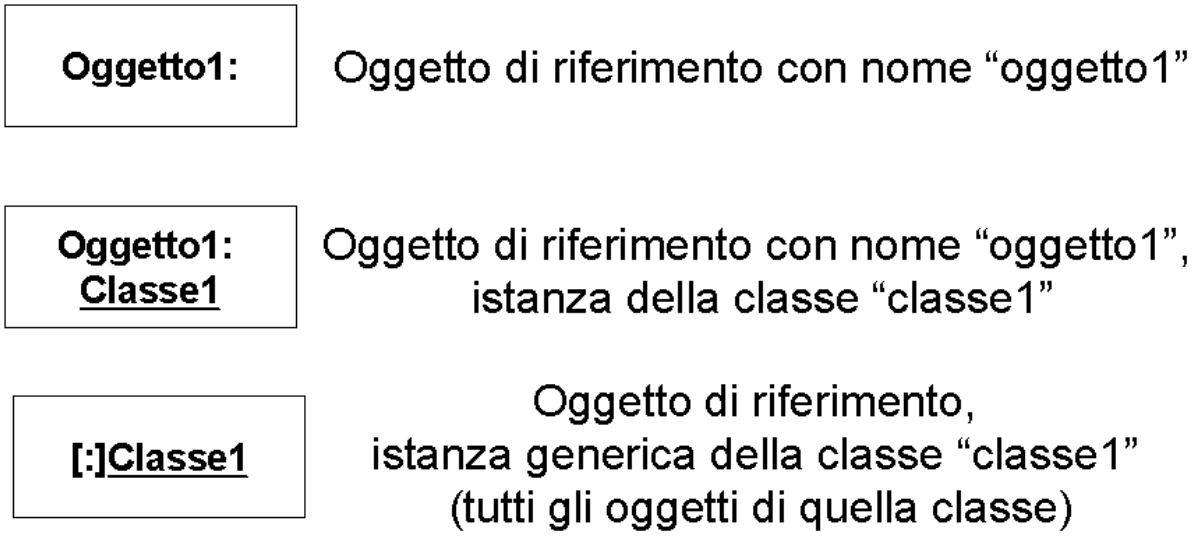


Fig. 4.24: Diverse tipologie di oggetto entro i diagrammi di sequenza o di comunicazione.

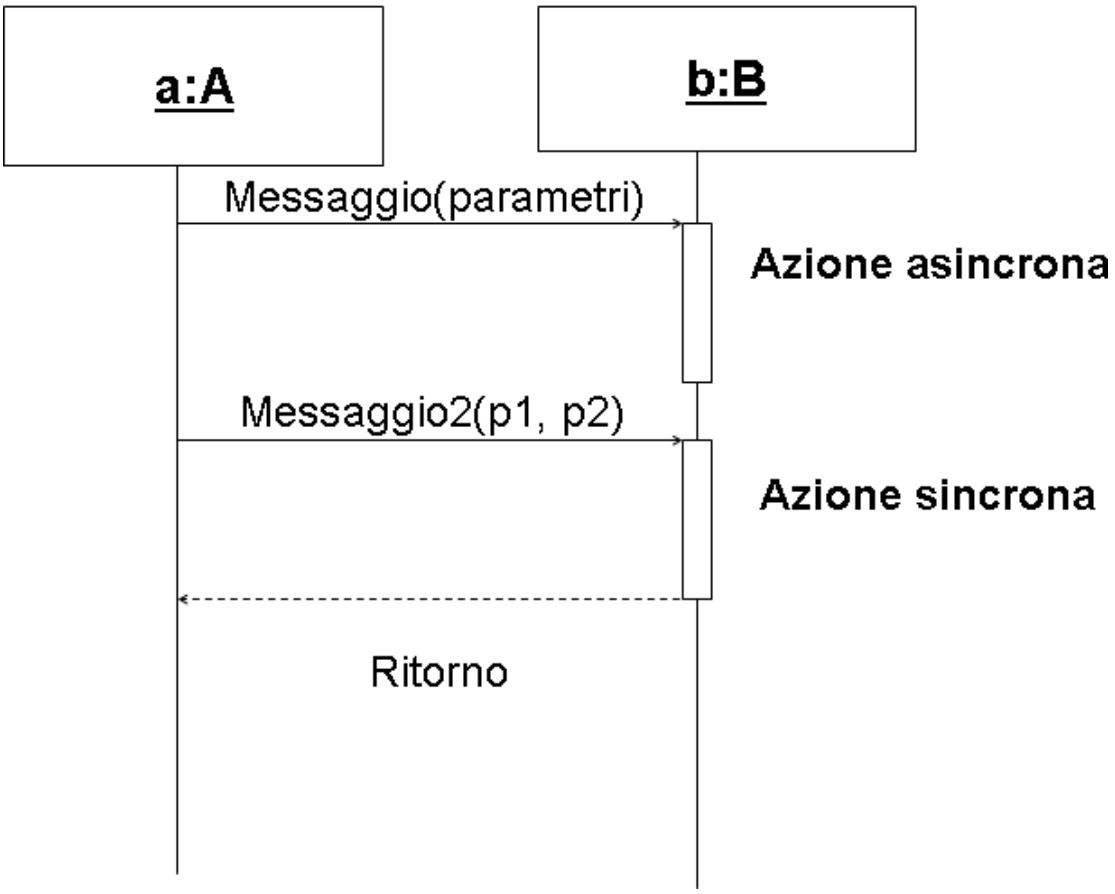


Fig. 4.25: Due diverse interazioni fra oggetti in un diagramma di sequenza, la prima asincrona (e quindi priva di ritorno) la seconda sincrona (al termine dell'esecuzione viene comunicato un risultato di ritorno, che il chiamante attende).

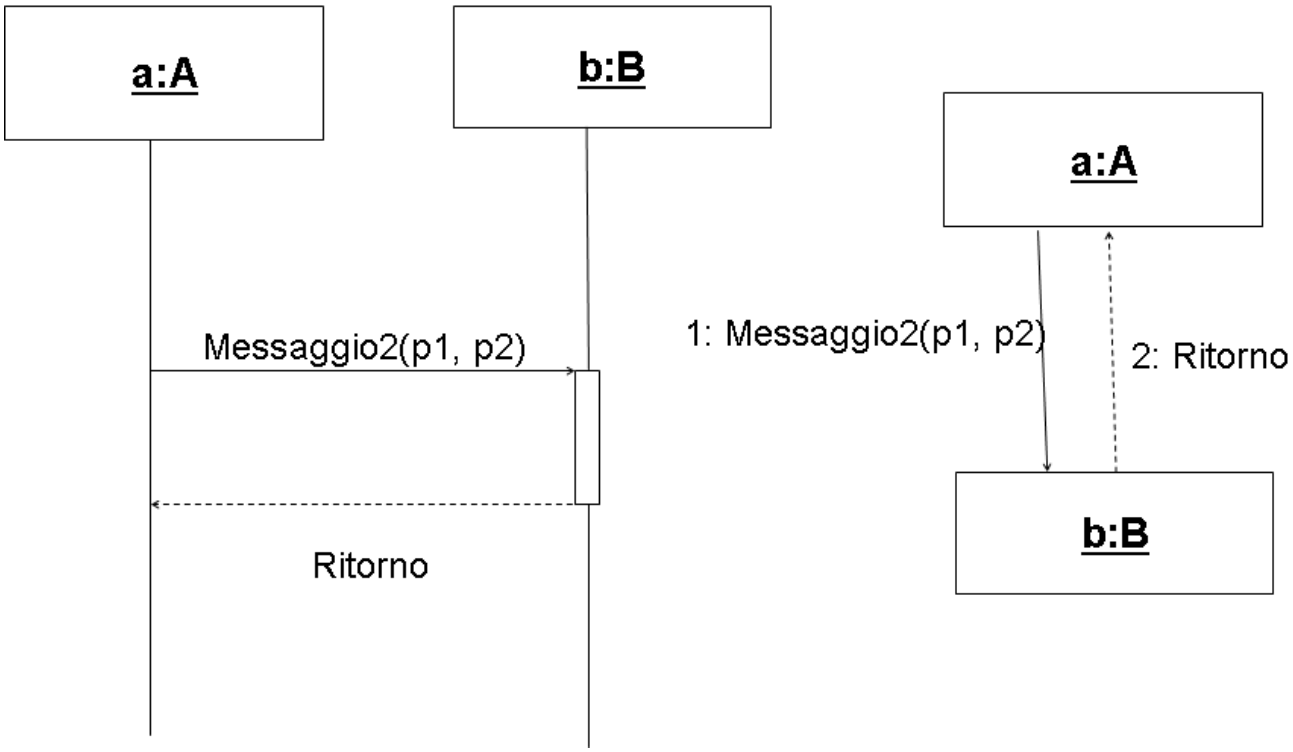


Fig. 4.26: Un diagramma di sequenza e a destra il suo equivalente diagramma di comunicazione. In quest'ultimo è bene esplicitare sempre il numero di sequenza dei messaggi.

Statechart Diagram

Gli Statechart Diagram (diagrammi di stato) esprimono una informazione duale a quella degli Activity Diagram, focalizzando l'attenzione non sulle azioni che avvengono ma sul cambiamento di stato di una particolare entità coinvolta nelle azioni stesse (ad esempio l'offerta di vendita passa dallo stato di "sottoposta" a quello di "accettata" e poi a quello di "spedizione in atto").
I diagrammi di stato possono essere usati per descrivere il comportamento nel tempo di un particolare elemento come un oggetto (ovvero una singola entità) o un intero sottosistema, ovvero l'evoluzione di una interazione. In pratica essi descrivono sequenze di stati ed azioni attraverso cui l'elemento considerato passa durante la propria vita reagendo a eventi discreti (segnali, chiamate a funzionalità...). L'enfasi è posta sugli stati e non sulle azioni.
I simboli di base dei diagrammi di stato sono molto simili a quelli dei diagrammi di attività, e sono rappresentati in figura 4.27 e 4.28 dove vengono mostrati sia stati semplici, sia stati che racchiudono al loro interno delle attività, ossia all'ingresso o all'uscita dei quali devono avvenire eventi che sono in realtà attività vere e proprie.

Fig. 4.27: Gli elementi di base di un diagramma di stato.

Nei diagrammi di stato sono presenti anche biforcazioni o riunioni delle transizioni da uno stato ad un altro, come rappresentato nelle figure 4.29 e 4.30. Un esempio d'uso dei diagrammi di stato è rappresentato nella figura 4.31, dove vediamo gli stati attraverso cui si passa durante la composizione di un numero telefonico e il conseguente inizio della chiamata. Ogni stato comprende un'azione, non indicata esplicitamente e le condizioni esterne che si verificano e determinano il percorso seguito vengono indicate come eventi.

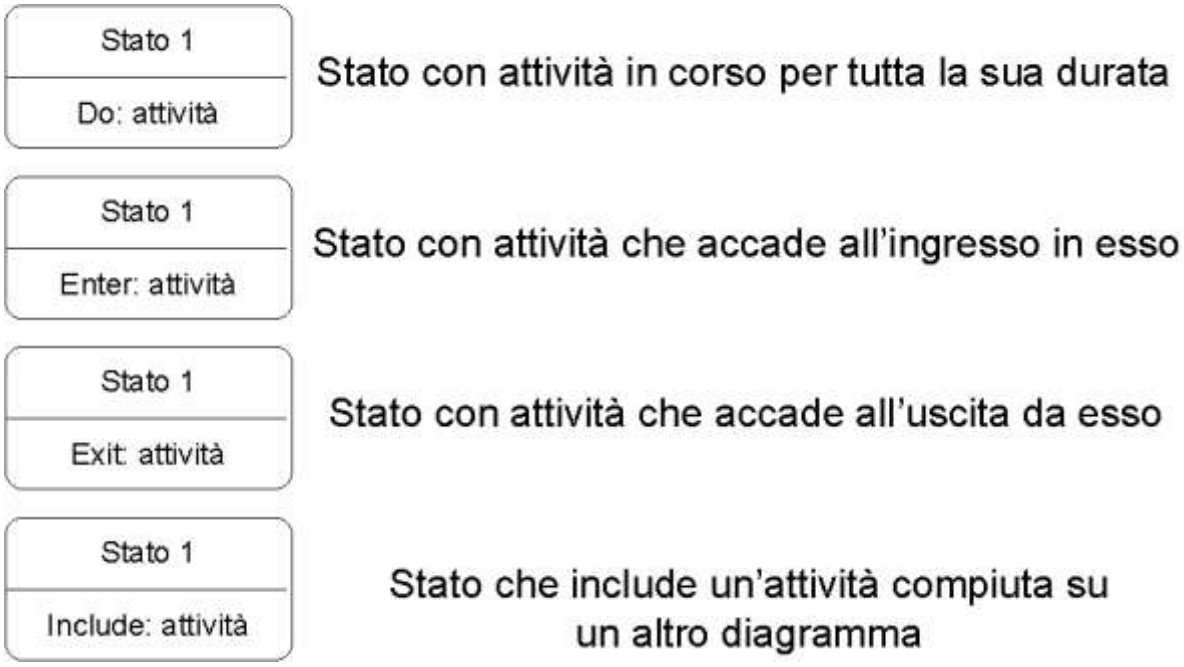


Fig. 4.28: Elementi di base i diagrammi di stato composti, che evidenziano la presenza di attività associate allo stato.

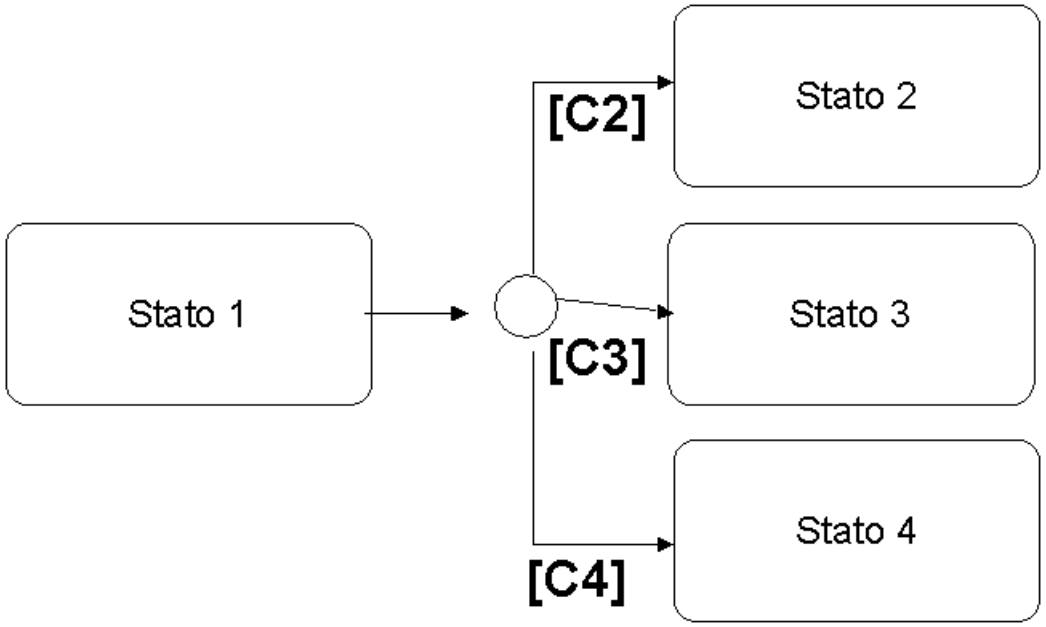


Fig. 4.29: Punto di scelta dinamica per un cambiamento di stato, dove in base alle condizioni, rappresentate tra le partenstesi quadre, si passa ad uno degli stati. Forma equivalente: le frecce partono direttamente da Stato 1.

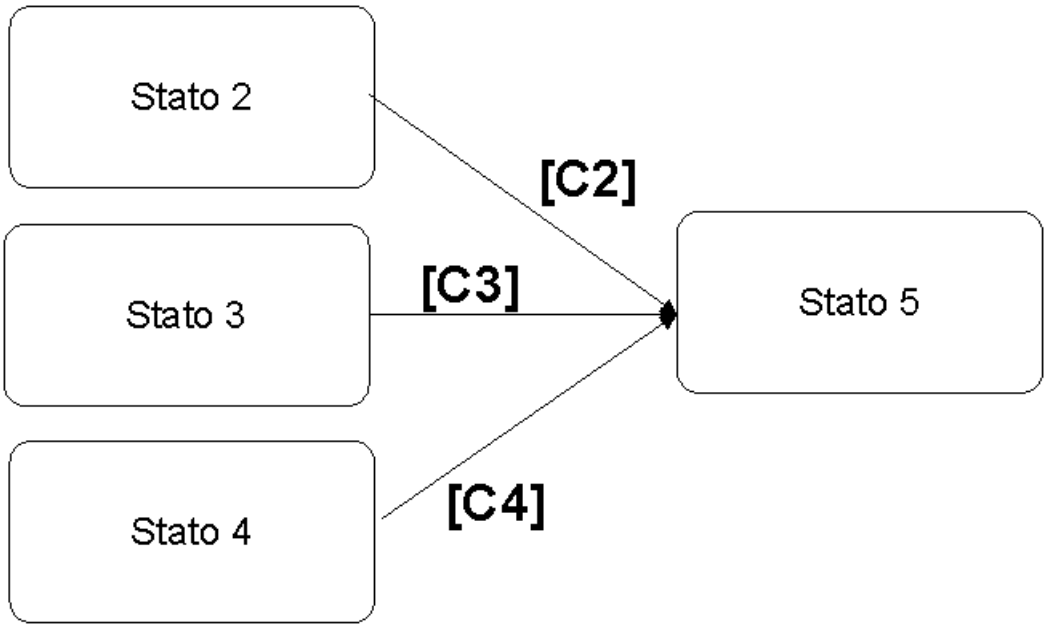


Fig. 4.30: Punto di giunzione per una transizione di stato, dove il successivo di tutti e 3 gli stati, seguendo le condizioni, è Stato 5

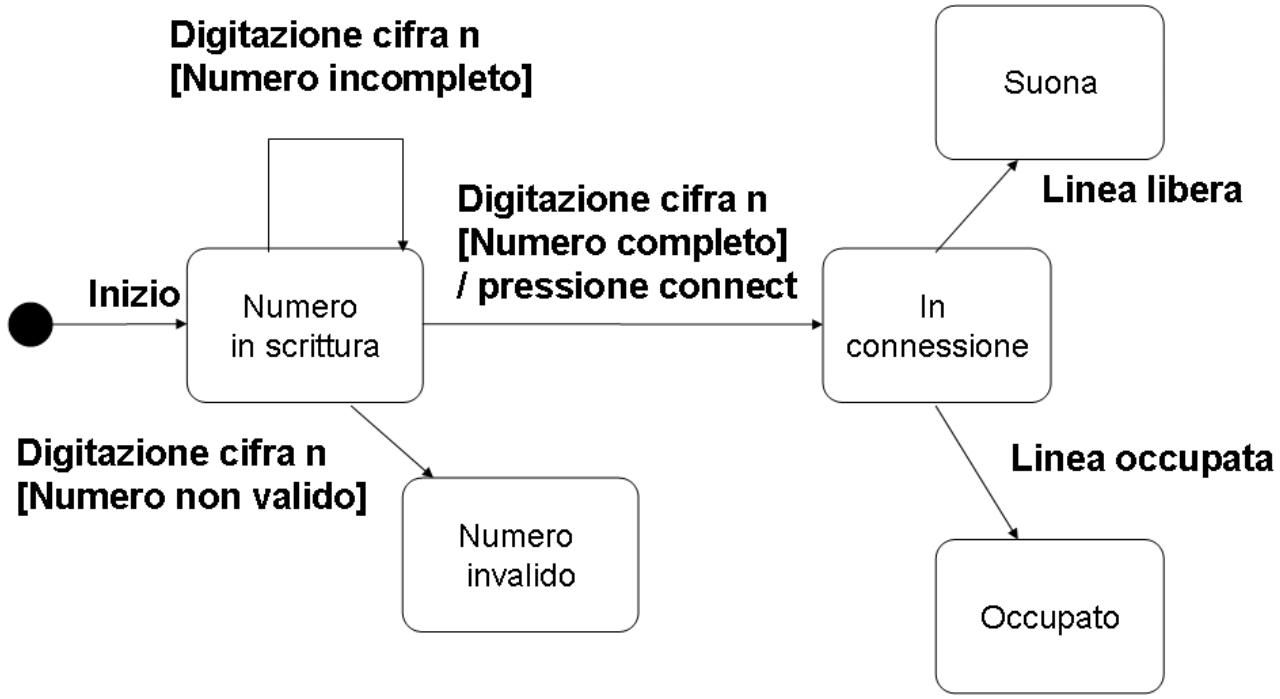


Fig. 4.31: Esempio di diagramma di stato che descrive i passaggi elementari compiuti per fare una chiamata telefonica.

Il legame fra i principali diagrammi UML

Il legame fra le viste rappresentate dai diagrammi e le informazioni ad esse associate viene espresso nella figura 4.32, dove si pone in evidenza che cosa significano i vari diagrammi e come sono legati l'uno all'altro. In base all'analisi che si sta compiendo ed al tipo di progetto può essere necessario porre l'accento sull'una o sull'altra delle caratteristiche, ovvero esaminare l'una o l'altra delle possibili viste del sistema che si vuole realizzare.

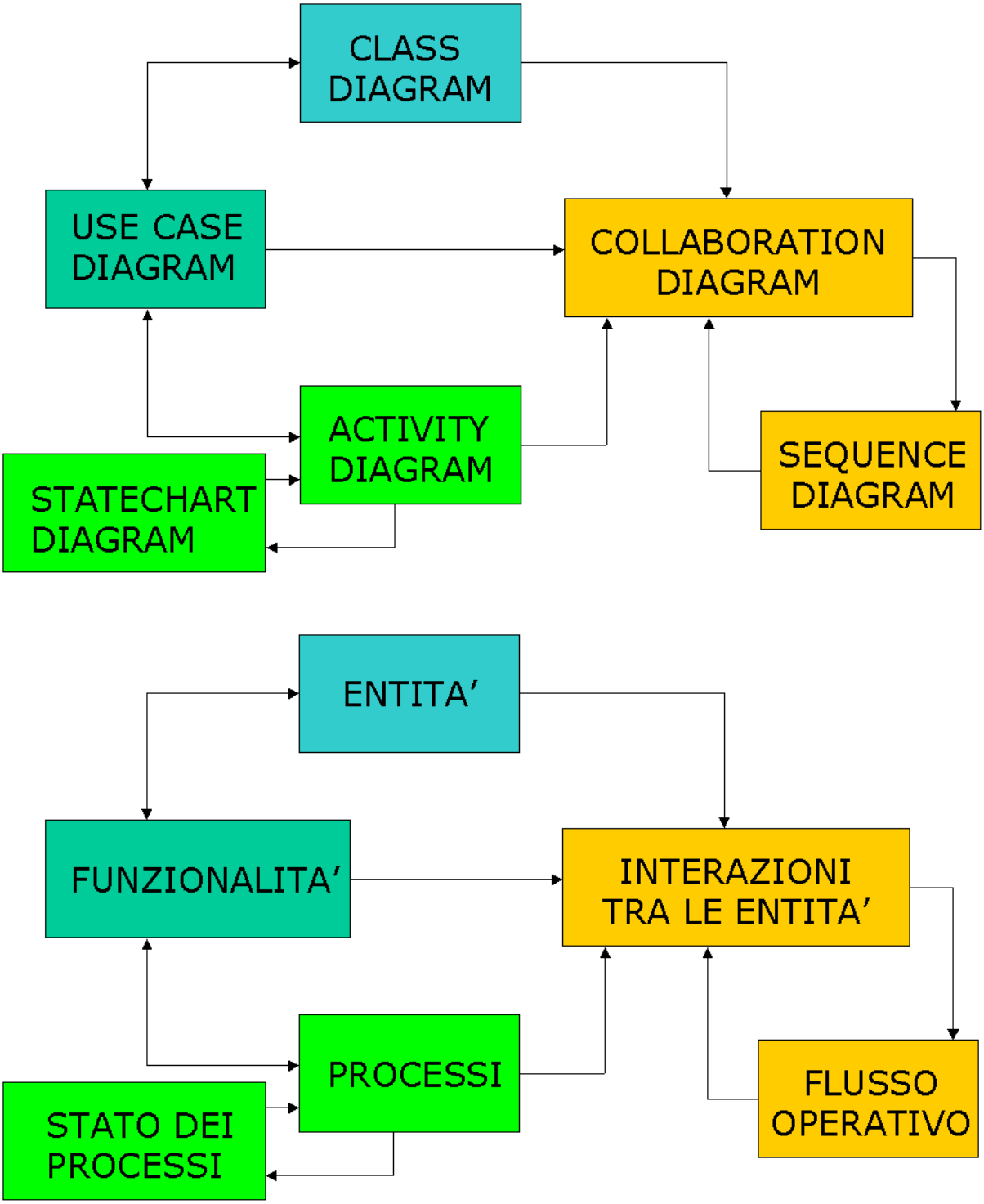


Fig. 4.32: I principali diagrammi UML, le relazioni che tra essi intercorrono ed i concetti che essi esprimono.

Gli strumenti CASE

CASE è l'acronimo di Computer Aided Software Engineering. Oltre alla metodologia di lavoro, questo termine individua anche una categoria di strumenti software ormai divenuti d'uso comune, almeno nella progettazione del software object-oriented.
Alcuni di questi strumenti permettono anche di generare il codice sorgente a partire dai class diagram. Ovviamente i metodi così ottenuti sono vuoti e devono essere riempiti manualmente dai programmatori. Il codice ottenibile è quindi lo scheletro delle classi.

Ormai sono disponibili moltissimi prodotti CASE sul mercato, alcuni dei più diffusi sono:
Commerciali:

- Rational Rose Modeler di IBM-Rational
- Visio di Microsoft
- PoseidonUML di Gentleware
- ModelioUML di ModelioSoft (disponibile anche in versione free http://www.modelio.org/downloads/download-modelio.html)
- VisualParadigm for UML (disponibile anche con versione ridotta community edition http://www.visual-paradigm.com/product/vpuml/whats-new/?edition=ce)
- Astah* (disponibile anche in versione ridotta libera per studenti)

Open source

- ArgoUML, progetto open source di Tiger
- StarUML, progetto open source di Tiger
- Umbrello UML Modeler
- Moduli UML plug-in per l'ambiente integrato Eclipse

Un elenco abbastanza completo degli strumenti disponibili sul mercato o come freeware è disponibile presso http://case-tools.org/uml.html.

Analisi e progettazione con UML

Il progetto software con UML

In quanto segue viene presentato un modello semplificato, tratto del ciclo fondamentale del Rational Unified Process di IBM, standard internazionale molto seguito nei grandi progetti [Web RUP].
Occorre ricordare sempre che i passi della analisi e della progettazione non devono mai essere vissuti come imposizioni, bensì come un ausilio al mantenimento della precisione e, conseguentemente, di un migliore controllo sull'andamento dei lavori e del progetto in toto.
Con queste premesse, i documenti di accompagnamento di un progetto software devono essere organizzati sulla base della successione di fasi che compongono il progetto stesso, che diventano:

1. Raccolta informale dei requisiti, comprensiva di tutti i requisiti funzionali e non
2. Stesura del Glossario, ossia del dizionario dei termini specifici del dominio del problema cui il progetto deve fornire una soluzione software
3. Stesura dell'analisi funzionale formalizzata con i casi d'uso
4. Definizione della cronologia delle interazioni (activity diagram), design delle interfacce utente e diagramma di navigazione
5. Definizione delle entità, delle relazioni che tra esse intercorrono e formalizzazione con il class diagram di analisi e la sua descrizione
6. Design della base di dati e sua costruzione (e generazione degli script di creazione tabelle)
7. Design architetturale del sistema
8. Progettazione software con la definizione delle classi e l'interazione
9. Pianificazione delle attività ed assegnamento degli incarichi, fatta attraverso gli strumenti del project management visti nel capitolo 2, che produce come risultato finale il diagramma degli incarichi
10. Sviluppo del software, seguendo le regole di stesura del codice definite a livello aziendale o di gruppo di sviluppo o uno degli standard presenti in letteratura
11. Debug del software stendendo le note sui problemi incontrati, in modo tale che possano servire d'aiuto in situazioni analoghe successive
12. Definizione delle procedure per l'installazione del software stesso.

In quanto segue sono definiti i dettagli di ciascuna sezione e dell'attività ad essa associata, comprensivi anche di alcuni esempi, tratti dai migliori progetti svolti per l'esame di Ingegneria del Software dei miei studenti dell'Università di Parma.

Prima raccolta dei requisiti

E' la prima fase del lavoro congiunto con il cliente e, a partire dalle sue intenzioni iniziali e dai suoi desiderata, ha lo scopo di produrre un documento informale, scritto in linguaggio naturale, che spieghi molto brevemente le intenzioni del cliente. In pratica questo primo documento serve a circoscrivere i limiti del lavoro successivo. Deve essere breve e il più possibile preciso e indicare con precisione l'argomento del progetto successivo.
Questo documento dovrebbe avere le seguenti caratteristiche:

1. Indicare chi è (o si immagina che sia, nel caso di un lavoro didattico) il cliente finale;
2. Indicare i requisiti che il cliente richiede;
3. Definire di che tipo di lavoro si tratta;
4. Indicare vincoli imposti da altri software o sistemi o ambienti esistenti con si debba interoperare o entro cui il risultato del progetto debba operare;
5. Indicare requisiti non funzionali (es. numero di accessi simultanei, dimensioni della base dati...);
6. Descrivere "informalmente" e a grandi linee il lavoro da svolgere.

Esempio di Documento di Raccolta dei Requisiti
Autore: Fabio Bettinazzi

Cliente : Cooperativa GruppoDue SNC con sede in via Gavardina di Sotto 130, 25100 Ponte S.Marco(BS).

Descrizione Del Contesto del Sistema
Situazione Aziendale: La cooperativa GruppoDue(fondata nel 2004) e' costituita da 40 soci, ognuno dei quali e' proprietario di un negozio di Ferramenta e/o Casalinghi. I soci provengono, per la maggior parte dalla provincia di Brescia ma sono presenti soci che provengono dalle provincie di Bergamo,Mantova. La cooperativa svolge la funzione di magazzino all'ingrosso per gli affiliati, cioè è in grado di ottenere i prodotti di consumo(ordinati dai soci) direttamente dalle aziende, senza passaggi intermedi, a prezzi favorevoli.
L'azienda al momento non presenta un sistema informatico in grado di raccogliere gli ordini dei soci in modo automatico, ma bensì al momento gli ordini vengono inviati alla cooperativa o tramite telefono o tramite fax. Questo modo di gestione degli ordini comporta perdite di tempo per la raccolta dei dati e la relativa gestione, a discapito della produttività, in quanto le uniche due segretarie si devono sobbarcare il compito di raccogliere gli ordini in arrivo via Fax e trasferire nel computer.

Obbiettivo: Il sistema che la cooperativa ha richiesto è un sistema di gestione informatico Web per l'invio e la ricezione degli ordini che, in particolare, permetta ad ogni utente del servizio (i soci) di accedere al sistema, previa autenticazione, e gli consenta di controllare i prodotti che il magazzino ha a disposizione o che, in caso di momentanea indisponibilità, è in grado di avere nell'arco di pochi giorni. Inoltre al sistema è richiesta anche una funzione di amministrazione più generale del magazzino, cioè è richiesta la possibilità di gestire le anagrafiche dei clienti, dei fornitori e di ogni prodotto presente in magazzino. In particolare l'amministratore del sistema deve poter controllare automaticamente quali prodotti presenti in magazzino sono esauriti o quali presentano una quantità minore di quella minima che deve essere sempre presente in magazzino, per evitare lunghe attese al socio che l'ha richiesta.
La gestione dell'ordine del cliente, da parte dell'amministratore deve essere eseguita nella seguente procedura: l'amministratore quando decide di gestire l'ordine deve poter marcare l'ordine in modo speciale e trasparente, poiché anche il cliente deve avere la possibilità di controllare se un suo ordine è già stato gestito, e quindi se, entro pochi giorni, gli arriveranno a domicilio i prodotti richiesti.

Inoltre la segnatura dell'ordine deve portare ad un automatico aggiornamento delle quantità di magazzino, dei prodotti presenti nell'ordine del cliente, in modo da poter controllare in tempo reale le quantità di ogni prodotto presenti in magazzino, e quindi avere un risparmio di tempo e lavoro.
Inoltre il cliente (GruppoDue) richiede che sia implementata la parte delle anagrafiche anche come applicazione a maschere tradizionale e non solo come applicativo WEB, poiché le segretarie sono più abituate ad interagire tramite un'interfaccia tradizionale rispetto all'ambiente WEB.

Vincoli: Il sistema deve poter essere implementato in modo da poter essere eseguito su calcolatori già presenti in azienda e che possa funzionare anche sui computer dei singoli soci senza bisogno di apportarvi modifica. Inoltre il funzionamento del Programma deve essere di facile comprensione vista l'avversità di molti soci alla tecnologia informatica. Quindi il sistema deve rispettare i seguenti vincoli:
Database:
Postgres v.8.1.3 con Gestore Grafico pgAdmin
Server Aziendale:
Processore Pentium 4 3GHz
Memoria Ram DDR2 512 MB
Hard Disk 60GB
Server Web:
Apache/Tomcat
Computer Soci:
Pentium 3 1GHz
Ram DDR 128 MB
Modem 56Kb

Vincoli Software: Sui computer già presenti in azienda è installato il Sistema Operativo LINUX con Kernel 2.6.10

Fonti Informative: I referenti per la parte di raccolta della informazioni necessarie per la produzione del sistema informatico sono: il

- Sig. Marco Rossi segretario
  - e-mail Marco.Rossi@gruppoDue.it
  - cellulare: 333 88888888
- Sig. Egidio Bianchi magazziniere
  - e-mail Egidio.Bianchi@gruppoDue.it
  - cellulare: 338 77777777

della cooperativa GruppoDue.

Stesura del Glossario

In questa fase si deve definire la terminologia del progetto, identificando con precisione le entità (persone, ruoli, luoghi, oggetti materiali, eventi, strutturazioni ecc...) coinvolte nel sistema del mondo reale (ovvero del dominio di business) che hanno importanza per il sistema informatico obiettivo del progetto. E' importante identificare con precisione le entità, allo scopo sia di definire meglio i loro scenari d'uso (Passo 3, gli Use Case), sia di individuare le Classi Entità (Passo 4, il Class Diagram d'analisi).
Il risultato di questa fase è il documento Glossario, che definisce con precisione tutti i termini corrispondenti alle entità coinvolte, evitando ambiguità.

Esempi di Glossario

Magazzino - Autore: Fabio Bettinazzi
Amministratore: E' la persona che ha l'accesso alla parte di gestione del sistema. Gestisce gli ordini in arrivo e ha il controllo sulle anagrafiche.In pratica è il super-utente del sistema.
Clienti: Nel sistema verranno indicati con la parola cliente gli utenti del sistema che possono solo inviare gli ordini al magazzino e controllare gli ordini inviati in passato. Quindi nel sistema reale i clienti saranno i soci della cooperativa.
Fornitori: I fornitore non hanno accesso al sistema ma sono importanti perché sono i rifornitori del magazzino.
Prodotti: Sono gli oggetti (i prodotti) fondamentali per il magazzino, sono gli oggetti reali di cui si occupa la cooperativa.
Ordini: Sono l'elemento fondamentale del sistema in quanto sono le entità che si scambiano i clienti con l'amministratore, cioè sono una raccolta di codici di prodotti con le relative quantità che i clienti richiedo al magazzino. Sono inoltre identificati attraverso un codice univoco e la data corrente in cui vengono creati.
Listino: E' l'entità che raggruppa tutti i prodotti che il magazzino della cooperativa ha a disposizione. Ogni prodotto presente in listino ha due parametri fondamentali:
Quantità Magazzino: Numero di Prodotti presenti al momento in magazzino.
Quantità Minima: E' la quantità minima che il magazzino dovrebbe sempre avere, è il parametro che identifica se un prodotto dovrà essere richiesto al fornitore o meno.

Ricettario Web - Autore: Davide Masi

Ricetta

Insieme di informazioni necessarie per la preparazione di un piatto.
Una ricetta e' caratterizzata da:
1. **Nome**;
2. Numero di **porzioni**;
3. **Attributi di classificazione**;
4. Lista di **ingredienti**;
5. Lista di **passi per la preparazione**;
6. **Tempo di preparazione**;
7. **Voto** medio degli utenti: solo se sono presenti commenti per la ricetta;
8. Lista dei **commenti** degli utenti;

Porzioni

Ogni ricetta e' dimensionata su un numero fissato di coperti. Assume valori compresi tra 1 e 20.

Ingrediente

Componente atomico di una ricetta; caratterizzato da:
1. **Nome**;

2. **Unità di misura**: relativa alla specifica ricetta in cui l'ingrediente compare;
3. **Quantità**: relativa alla specifica ricetta in cui l'ingrediente compare, espressa in funzione dell'unità di misura. Quando non significativa, può essere sostituita dall'indicazione "Quanto Basta", codificata ponendo quantità a -1. In tal caso l'unità di misura non è significativa;
4. **Note**: eventuali caratteristiche particolari richieste per la specifica ricetta; quando non specificata assume il valore "nessuna". Quando questo attributo assume il valore di default non e' visualizzato nell'interfaccia web.

Unità di misura

E' caratterizzata da:
1. **Nome**: nome dell'unità di misura, deve essere univoco;
2. **Sigla**: abbreviazione di al più tre lettere del nome dell'unità di misura; quando impostata ad "na" non è significativa;

Può assumere i seguenti valori:
- bicchiere
- cucchiaio
- decilitro (dl)
- ettogrammo (hg)
- grammo (g)
- chilogrammo (kg)
- litro (l)
- millilitro (ml)
- pizzico
- tazza
- unitaria

Preparazione

Istruzioni in linguaggio naturale per la preparazione di una ricetta. Le istruzioni sono suddivise in più passi, per facilitare la lettura. Ad ogni ricetta è associata una preparazione costituita di almeno un passo.
Ogni passo della preparazione ha una dimensione massima prefissata, in termini di numero di caratteri, che sarà definita successivamente.

Tempo necessario alla preparazione

Tempo approssimativamente necessario alla preparazione di una ricetta, espresso in ore.

Commento

Ogni commento è creato da un solo utente, e può essere cancellato solo dall'amministratore. Ogni commento ha una dimensione massima prefissata, in termini di numero di caratteri, che sarà definita successivamente.
Caratterizzato da:
1. **Utente** che lo ha inserito;
2. **Voto** alla ricetta [1..10];
3. **Testo** del commento;
4. **Timestamp** relativo all'inserimento;

Utente

Un utente è caratterizzato:
1. **Indirizzo** e-mail;
2. **Username**;
3. **Password**;

Attributi di classificazione

Gli **attributi di classificazione** per una ricetta sono:

1. **Portata**;
2. **Costo Stimato**;
3. **Alimento Base**;

Ogni ricetta ha una e una sola classificazione.

Portata

Assume i valori:

- antipasto
- primo
- secondo
- contorno
- salsa
- frutta
- dolce
- zuppa
- piatto base
- piatto unico
- pizza/focaccia

Costo Stimato

1. Indicazione approssimata del costo di una ricetta.

Assume i valori:

- basso
- medio
- alto
- molto alto

Alimento Base

Indica l'alimento che caratterizza il piatto. Assume i valori:

- nessuno/vari
- carne
- pesce
- pasta
- riso
- verdura
- uova
- formaggio
- frutta

Analisi funzionale con i casi d'uso e relative descrizioni

In questa fase devono essere individuati con precisione gli scenari d'uso del sistema, ovvero, in modo più generale, gli scenari di interazione fra il sistema e gli attori, ovvero le entità esterne al sistema con cui esso interagisce e comunica. I passi necessari in questa fase possono essere così suddivisi:

1. Definizione esatta del boundary o confine del sistema (entro sistemi particolarmente complessi questa fase può anche essere applicata a sotto-sistemi);
2. Identificazione e definizione degli attori, ossia delle entità esterne con cui il sistema (o i sotto-sistemi) oggetto dell'analisi interagiscono e comunicano;
3. Individuazione dei vari scenari di uso/interazione fra sistema ed attori, che corrisponderanno ai singoli casi d'uso, identificati dalle singole ellissi nel diagramma; si ricordi che i casi d'uso descrivono cosa si vuole che il sistema faccia, non come questo comportamento deve essere implementato (modello della black box);
4. Definizione delle interazioni entro i singoli casi d'uso; tali interazioni, strutturate nella forma della richiesta dell'attore cui corrisponde una risposta del sistema (tenendo conto anche di eventuali comunicazioni asincrone o autonome del sistema quali ad esempio allarmi), andranno a costituire i campi descrizione dei singoli casi d'uso (operazione detta in gergo "srotolamento" dello use case);
5. Esame dei diagrammi così ottenuti e delle loro descrizioni per potere procedere alla raccolta a fattore comune di parti fra i sigoli use case entro diagrammi, facendo uso delle relazioni extends ed include definibili tra i vari casi d'uso;
6. Il passo 5 può essere iterato più volte; occorre tenere conto della granularità del problema e del grado di definizione e precisione che si vuole raggiungere; inoltre occorre tenere presente che un singolo caso d'uso spesso dà origine ad una singola maschera (sia essa maschera testuale, singola window in ambiente GUI o pagina Web); infine si tenga presente che spesso da un caso d'uso deriverà anche un caso di test durante la fase di test del sistema informatico realizzato.

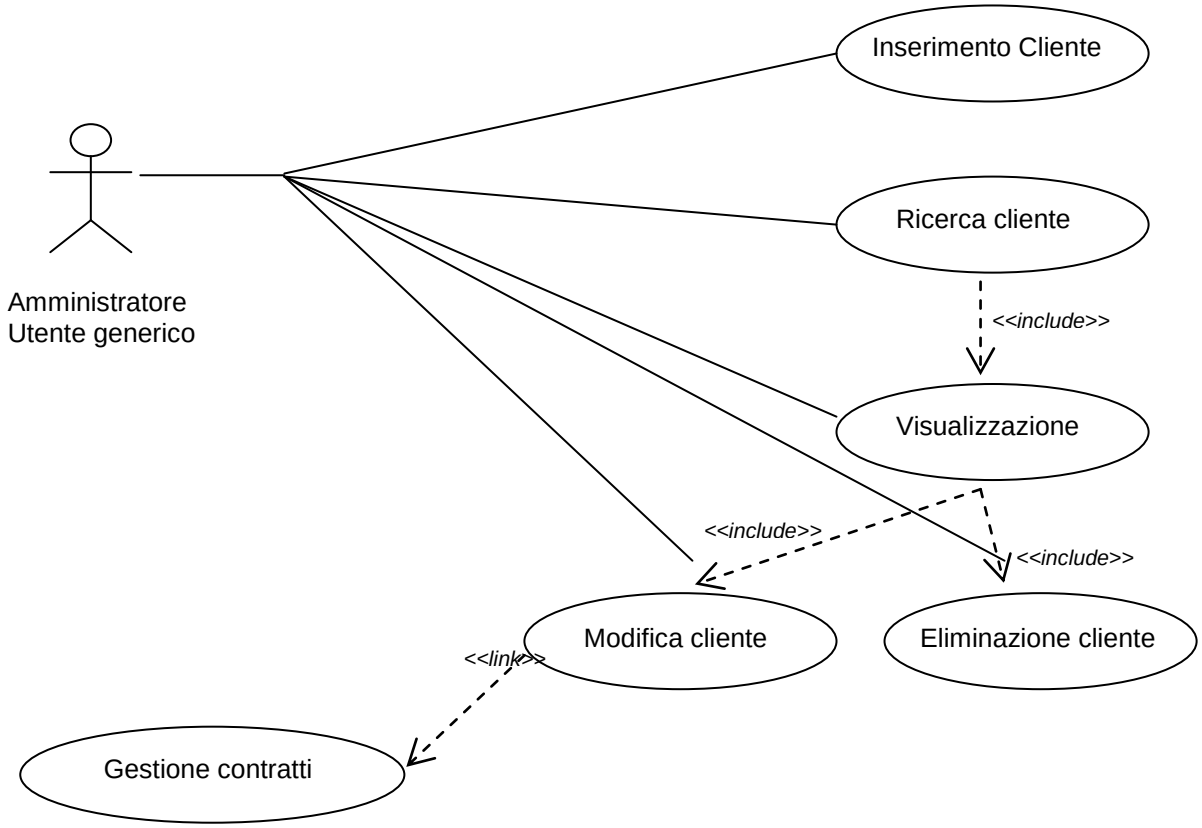
Il prodotto di questo passo è l'insieme completo degli use case inseriti entro uno o più use case diagram, ognuno corredato di adeguata descrizione, strutturata chiaramente in forma di request-response e considerando sia il percorso principale di interazione (basic course) sia gli eventuali percorsi alternativi (alternative courses), quali quelli che si verificano in presenza di errori nei dati introdotti ecc... Il diagramma e le descrizioni devono essere ben strutturati, chiari ed esaurienti, in quanto tutti i passi successivi si baseranno su di essi.

Siccome gli use case diagram non esprimono direttamente relazioni di flusso logico/temporale fra i loro componenti, può essere utile esplicitare tali relazioni attraverso un activity diagram derivato, che definisca le attività associate ai singoli use case (potrebbero in tal caso essere necessarie ulteriori scomposizioni od aggregazioni) e le relazioni logiche e temporali che tra esse intercorrono. Da questo diagramma deriva, più o meno direttamente, anche il diagramma di navigazione fra le finestre o maschere che costituiscono l'interfaccia esterna utente dell'applicazione in progetto.

I diagrammi da soli non sono sufficienti e si devono aggiungere ulteriori descrizioni, che rendano più preciso il tutto.

Esempi di Use Case con descrizione

Gestionale, gestione anagrafica clienti – Autori: Giulio Destri e Alberto Picca



1. Macro-Caso d'uso: "**gestione anagrafica clienti**"		
Attori coinvolti:	Utente	
Breve descrizione:	L'utente gestisce i nominativi presenti in un archivio dati esterno. Sono previste le funzioni di visualizzazione, inserimento, modifica ed eliminazione.	
Precondizione:		
Scenario principale :		
	Sistema	*Utente*
	Visualizza l'elenco dei clienti	Sceglie una delle seguenti opzioni di gestione : ○ Inserimento di un nuovo cliente ○ Modifica di un cliente ○ Ricerca di un cliente ○ Eliminazione di un cliente
Alternative:		
Eccezioni:		
Osservazioni:	Viene avviato il relativo use case in base alla scelta effettuata.	

1.1. Caso d'uso: "**Gestione anagrafica clienti - Inserimento**"		
Attori coinvolti:	Utente	
Breve descrizione:	L'utente ha la possibilità di inserire i dati per un nuovo cliente. Il sistema può procedere alla sincronizzazione dei dati con un archivio esterno.	
Presupposti:		
Descrizione del procedimento :		
	Sistema	*Utente*
	Visualizza la maschera per l'inserimento dei dati del cliente Attraverso una libreria interna viene richiamato il metodo per salvare i dati in una base di dati esterna. Se l'utente ha premuto "Annulla" si riporta nello stato precedente	Inserisce i dati del cliente Preme il pulsante "OK", altrimenti il pulsante "Annulla"
Osservazioni:	In caso di errore, ad esempio se i dati obbligatori non sono stati inseriti correttamente allo viene visualizzato un messaggio d'errore e l'inserimento non viene effettuato fino a che non sono stati corretti gli errori.	

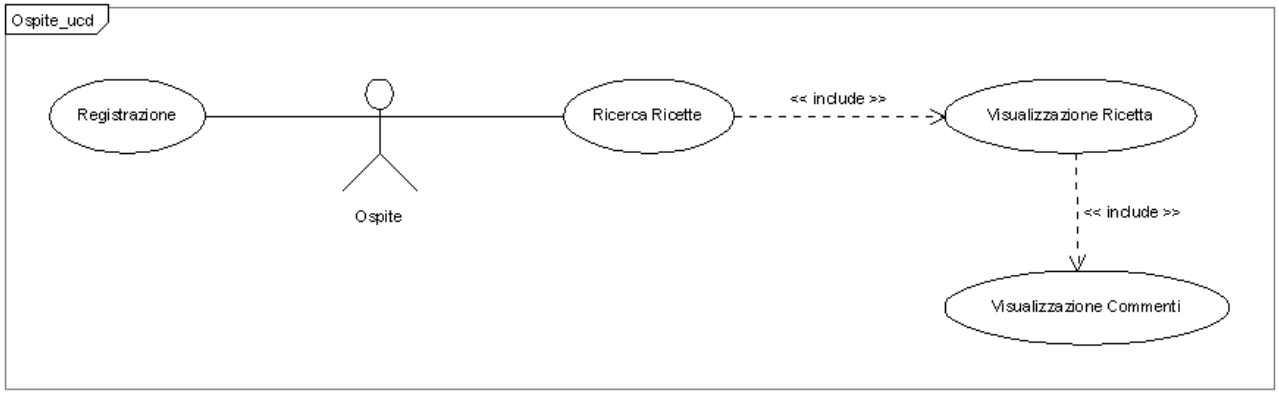
1.2. Caso d'uso: "**Gestione anagrafica clienti – Ricerca cliente**"		
Attori coinvolti:	Utente	
Breve descrizione:	L'utente ha la possibilità di ricercare un cliente attraverso alcuni criteri	
Presupposti:		
Descrizione del procedimento :		
	Sistema	*Utente*
	Visualizza l'elenco di tutti i clienti Mostra l'elenco dei clienti che soddisfano la ricerca	Inserisce i parametri per la ricerca Preme il pulsante "OK", altrimenti il pulsante "Annulla"
Osservazioni:	In caso di errore, ad esempio se i dati obbligatori non sono stati inseriti correttamente allo viene visualizzato un messaggio d'errore e la ricerca non viene avviata fino a che non sono stati corretti gli errori.	

1.3. Caso d'uso: "**Gestione anagrafica clienti - Visualizzazione**"		
Attori coinvolti:	Utente	
Breve descrizione:	L'utente ha la possibilità di visualizzare i clienti	
Presupposti:		
Descrizione del procedimento :		
	Sistema	*Utente*
	Visualizza la maschera con la griglia contenente l'elenco dei clienti In base alla scelta fatta dall'utente si viene rimandati al relativo use-case	Può effettuare le seguenti operazioni: - Modifica del cliente selezionato - Eliminazione del cliente selezionato
Osservazioni:		

1.4. Caso d'uso: "**Gestione anagrafica clienti – Modifica** "		
Attori coinvolti:	Utente	
Breve descrizione:	L'utente ha la possibilità di modificare i dati del cliente o di visionare altre informazioni ad esso correlate, come ad esempio le scadenze o i contratti stipulati. Il sistema può procedere alla sincronizzazione dei dati con un archivio esterno.	
Presupposti:	L'utente deve aver selezionato un nominativo dallo stato precedente	
Descrizione del procedimento :		
	Sistema	*Utente*
	Visualizza la maschera di modifica del nominativo in precedenza selezionato Visualizza le altre informazioni correlate al cliente Attraverso una libreria interna viene richiamato il metodo per salvare i dati in una base di dati esterna. Se l'utente ha premuto "Annulla" si riporta nello stato precedente	Può modificare i dati del cliente e premere il pulsante "OK", altrimenti il pulsante "Annulla"
Osservazioni:	In caso di errore, ad esempio se i dati obbligatori non sono stati inseriti correttamente allo viene visualizzato un messaggio d'errore e il salvataggio non viene avviato fino a che non sono stati corretti gli errori.	

<table>
<tr><td colspan="3">1.5. Caso d’uso: “Gestione anagrafica clienti – Eliminazione ”</td></tr>
<tr><td>Attori coinvolti:</td><td colspan="2">Utente</td></tr>
<tr><td>Breve descrizione:</td><td colspan="2">L’utente ha la possibilità di eliminare un cliente. Il sistema può procedere alla sincronizzazione dei dati con un archivio esterno.</td></tr>
<tr><td>Presupposti:</td><td colspan="2">L’utente deve aver selezionato un nominativo dallo stato precedente</td></tr>
<tr><td colspan="3">Descrizione del procedimento :</td></tr>
<tr><td></td><td>Sistema</td><td>Utente</td></tr>
<tr><td></td><td>Chiede conferma all’utente della reale volontà di eliminazione

Se l’utente ha premuto “OK” allora attraverso una libreria interna viene richiamato il metodo per cancellare il dato dalla base di dati esterna.
Se l’utente ha premuto “Annulla” si riporta nello stato precedente</td><td>Può premere “OK” per confermare o “Annulla” per cancellare l’azione</td></tr>
<tr><td>Osservazioni:</td><td colspan="2">Qualora al nominativo che si desidera eliminare siano presenti altri elementi ad esso correlato, come ad esempio una o più pratiche, allora il sistema avviserà l’utente che eliminando il cliente gli altri elementi non saranno più accessibili e quindi verranno automaticamente eliminati dall’archivio.</td></tr>
</table>



Visualizzazione ricetta

Attori coinvolti : **Ospite**.
Descrizione breve : un Ospite vuole visualizzare i dettagli di una ricetta.
Presupposti : nessuno.

Descrizione del procedimento

1. Il **Sistema** mostra gli attributi della ricetta da visualizzare. Gli attributi sono elencati secondo il seguente ordine:
    1. Nome.
    2. Classificazione della ricetta: tipo di portata e alimento caratterizzante, costo.
    3. Tempo totale di preparazione.
    4. Ingredienti: sono presentati in forma tabulare. La prima colonna contiene il nome dell'ingrediente, la seconda la quantità, la terza l'unità di misura. Se la quantità codifica l'indicazione "quanto basta" l'unità di misura non è mostrata. Se sono presenti le note per un particolare ingrediente (cioè se il loro valore e' diverso da quello di default…) sono mostrare come quarta colonna. Gli ingredienti sono preceduti dall'indicazione del numero di porzioni.
    5. Preparazione: sotto forma di elenco numerato sono mostrati tutti i passi necessari alla preparazione.
    6. Commenti: sono mostrati gli ultimi 3 commenti inseriti, secondo l'ordine cronologico. I commenti sono presentati in forma tabulare. Ogni commento occupa due righe: la prima contiene il voto alla ricetta, la data di inserimento e lo username dell'utente che ha scritto il commento, la seconda riga contiene il testo.
2. L'**Ospite** sceglie tra:
    1. Visualizzare tutti i commenti associati alla ricetta:.
    2. Terminare lo use case;
3. **Caso 1:** viene avviato lo use case "Visualizzazione Commenti" per la ricetta corrente.

Effetti : nessuno.
Osservazioni : quando uno degli attributi opzionali non è disponibile non viene visualizzato.

In alternativa alla sopra indicata forma tabellare per la descrizione si può usare anche la forma di elenco, di cui qui riportiamo un esempio completo

Caso d'Uso/Attività: Nome del caso d'uso/dell'attività

Breve descrizione

Breve descrizione. Esempio: L'utente gestisce i nominativi presenti in un archivio dati esterno. Sono previste le funzioni di visualizzazione, inserimento, modifica ed eliminazione.

Trigger

Quale evento o serie di eventi fa iniziare questo caso d'uso?

Prerequisiti

Quali sono i prerequisiti? Che operazioni dovrebbero essere già state compiute?

Input

Quali sono gli input, provenienti dagli stadi precedenti, ovvero quali risultati di operazioni precedenti vengono ad essere l'ingresso di questa sequenza di azioni?

Attori coinvolti con indicazione se attivi o passivi

Chi sono gli attori (utenti o sistemi esterni) che compiono azioni sul sistema entro questo caso d'uso, o che compiono azioni in questa attività?
Esempio: Utente, Amministratore

Basic course

Percorso principale, ordinato come richiesta / risposta alternando utente e sistema
Esempio:

Sistema:

- Visualizza l'elenco dei clienti

Utente:

- Sceglie una delle seguenti opzioni di gestione :
  - Inserimento di un nuovo cliente
  - Modifica di un cliente
  - Ricerca di un cliente
  - Eliminazione di un cliente

Alternative course

Percorsi alternativi, sempre ordinati come richiesta / risposta

Errori / Eccezioni

Cosa succede in caso di errore? Come deve comportarsi il sistema? Che segnalazioni devono essere date all'utente?

Output

Dati o altro risultanti dall'attività compiuta

Tecniche / linee guida / buone pratiche

Eventuali buone pratiche provenienti da esperienze simili precedenti

Stesura del Diagramma di navigazione

L'insieme dei casi d'uso, pur esprimendo perfettamente l'insieme delle interazioni tra utente e sistema ed i legami logici che fra esse intercorrono, manca dell'aspetto cronologico della navigazione fra le finestre od interfacce, per cui, attraverso l'uso di un activity diagram, occorre dare un legame anche cronologico alla successione delle interazioni.

Non è detto che le singole attività abbiano sempre una corrispondenza uno a uno con i singoli casi d'uso: in alcuni casi è opportuno suddividere un singolo use case in più di un'attività o, viceversa, accorpare in una sola attività più di un caso d'uso.

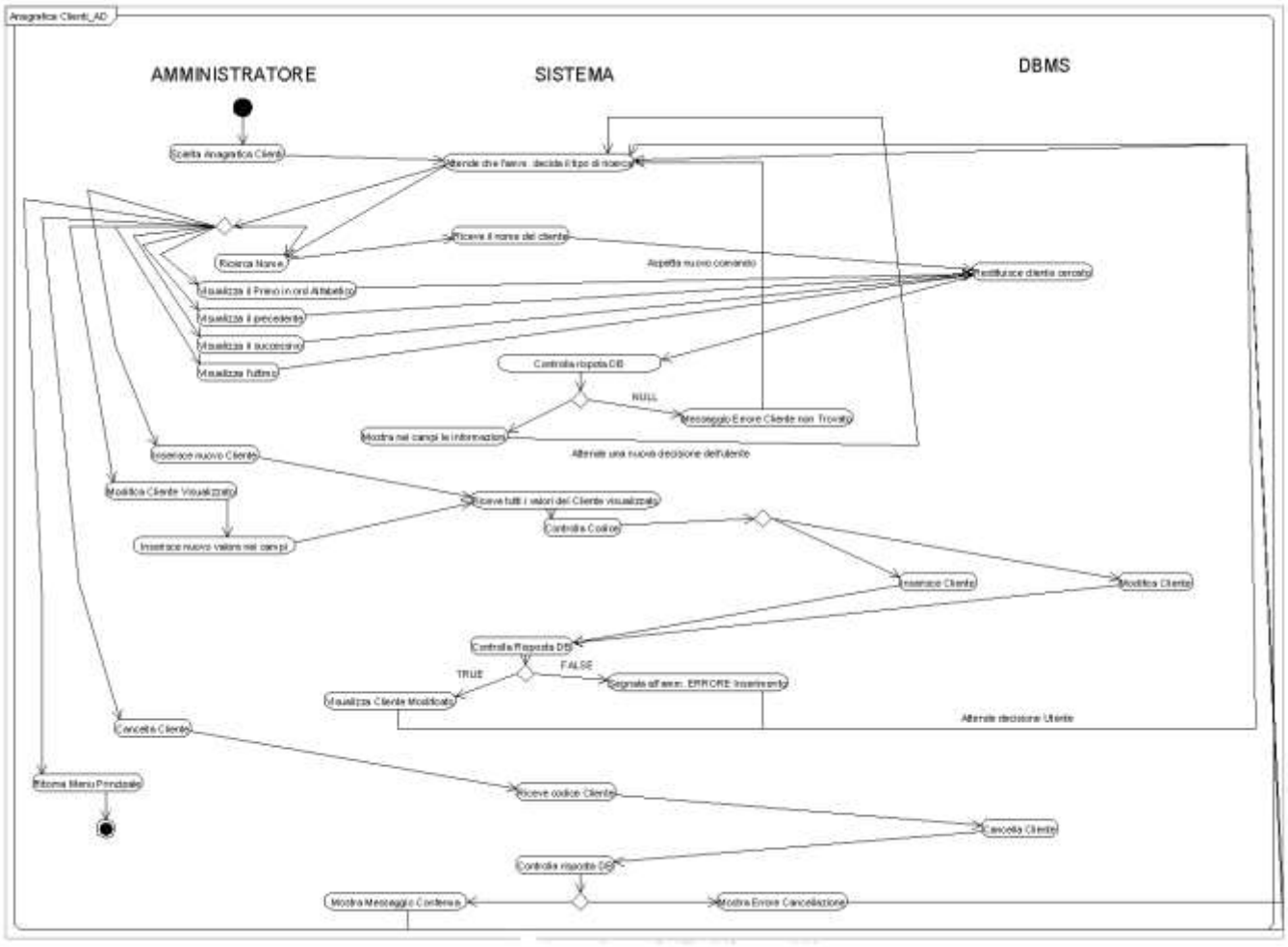
In ogni caso il risultato di questa fase viene essere espresso attraverso un activity diagram che definisca la successione cronologica di attività, ciascuna delle quali corrisponde ad un tipo particolare di interazione tra gli attori ed il sistema.

Da questo deriva il diagramma di navigazione, entro il quale da ciascuna attività è derivata una interfaccia utente (maschera Form o Web) e che definisce con precisione la navigazione fra le maschere utente.

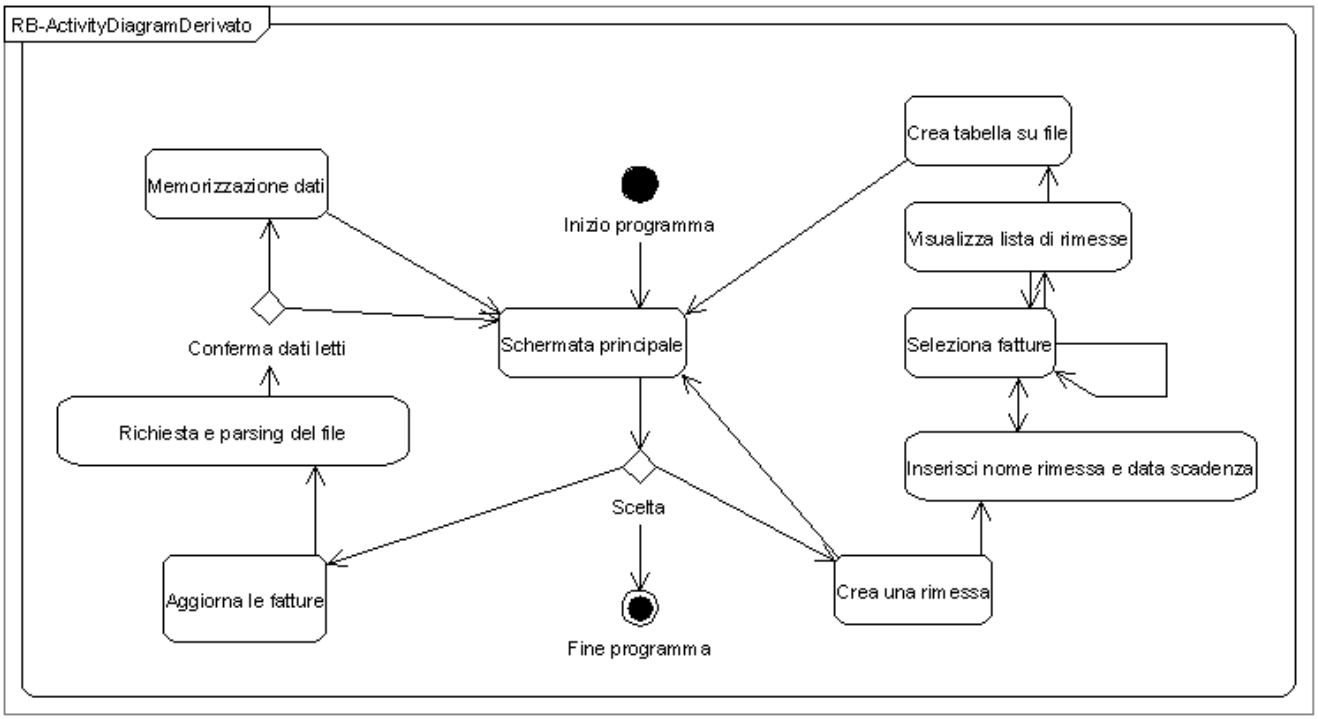
I diagrammi vanno ovviamente corredati di descrizione di ciascuna delle interfacce utente.

Esempi di Activity Diagram e Diagrammi di navigazione

Gestione Magazzino - Autore: Fabio Bettinazzi



Gestione Fatture e Rimesse Bancarie – Autore: Alessandro Vincenzi



Descrizione
Schermata Principale
La schermata principale è la finestra che viene presentata all'avvio dell'applicazione e deve fornire all'utente una scelta:

1. Aggiornare i dati delle fatture e dei clienti
2. Creare le rimesse
3. Uscire dal programma

Il tutto deve essere gestito con 3 pulsanti. Nei primi due casi l'azione da eseguire corrisponde all'apertura di una nuova finestra grafica mentre nel terzo la chiusura dell'applicazione

Aggiornamento delle fatture
La finestra di aggiornamento delle fatture si apre in seguito all'azione sul pulsante corrispondente nella schermata principale. Tale finestra deve offrire la possibilità all'utente di inserire il nome del file. Questo lo si può fare in due modi:

1. inserendo il path del file in modo diretto
2. selezionando il file con una navigazione nell'albero delle directory del sistema ospitante.

In seguito alla selezione del file si deve eseguire il parsing dello stesso, modo da leggere tutti i dati presenti.
Il parsing si esegue seguendo il modello di fattura e leggendo una fattura alla volta (riga per riga). Man mano che i dati si leggono, vanno salvati in oggetti del tipo corrispondente (fattura, cliente ...) e al termine della singola fattura si caricano tutti gli oggetti in un vettore. Nel caso si verifichino errori che compromettano la lettura del file, il processo di parsing viene arrestato e si comunica l'errore all'utente. Nel caso sulla fattura non si trovino i dati aspettati, tale informazione deve essere salvata per una comunicazione successiva.
Al termine della lettura del file, si deve visualizzare una finestra di riassunto dei dati letti (solo il numero di fattura) e dei dati mancanti. Sempre con questa finestra si chiude all'utente se procedere
con la memorizzazione dei dati o con l'annullamento dell'operazione. In quest'ultimo caso i dati vanno persi e si rientra alla schermata principale. Se l'utente sceglie di procedere con la

memorizzazione allora si crea una connessione al database in modalità di scrittura e senza commit automatico e si inseriscono i dati nelle corrispondenti tabelle seguendo il seguente ordine:

1. descrizioni dei pagamenti
2. banche
3. clienti
4. fatture

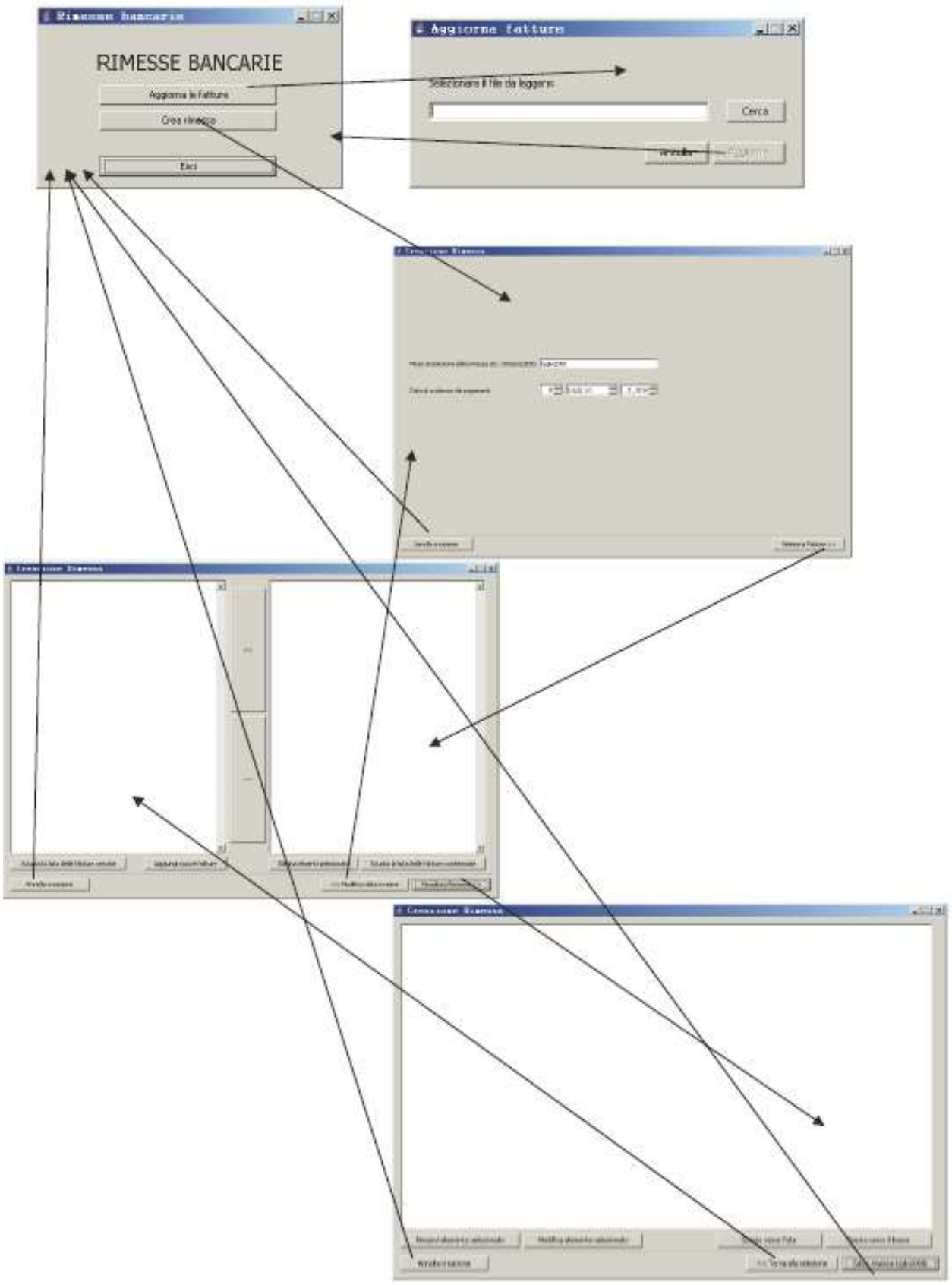
in modo da garantire i vincoli di chiave esterna tra le fatture. Se durante tale operazione non si verificano errori da parte del database, allora si esegue il commit delle operazioni, altrimenti si esegue il rollback e si comunica l'insuccesso all'utente.
In ogni caso si rientra nella schermata principale.

Creazione delle rimesse
La finestra di creazione della rimessa si apre in corrispondenza dell'azione sul pulsante della schermata principale. Il processo di creazione delle rimesse consta di quattro fasi:

1. inserimento del nome della rimessa e della data di scadenza dei pagamenti
2. selezione delle fatture
3. eventuale modifica dei dati delle rimesse
4. creazione delle tabelle con le rimesse

Alle prime tre fasi deve essere associata un'interfaccia grafica, mentre per la quarta non occorre.
La finestra di creazione delle rimesse deve quindi avere tre pannelli che si alternano sulla finestra in base alle operazioni dell'utente su due pulsanti di avanzamento e retrocessione della fase.

- Nel primo punto occorre solo fornire la possibilità di inserire un nome ed una data.
- Nel secondo punto occorre fornire due liste: la lista a sinistra conterrà l'elenco delle fatture temporaneamente scelte (con interrogazioni al database) mentre la lista a destra conterrà l'elenco delle fatture da trasformare in rimesse. Da tali liste deve essere possibile aggiungere ed eliminare elementi in modo semplice ed intuitivo tramite selezioni e azioni su pulsanti.
- Il passaggio alla terza componente del processo di creazione delle rimesse consiste nel prendere l'elenco delle fatture presenti sulla lista di destra (quelle confermate) e creare una terza lista di rimesse leggendo i dati necessari dal database. Se durante la creazione della lista si trovano fatture dello stesso cliente, queste vanno unite nella stessa rimessa. La lista deve offrire la possibilità di selezione per l'eliminazione dalla lista o per la modifica dei dati (un elemento alla volta). Eventuali modifiche fatte non devono corrispondere a memorizzazioni di dati sul database.
- La quarta ed ultima parte di questo processo consiste nel prendere la lista delle rimesse e creare un file pdf con tabelle riassuntive dei dati. Tali pagine devono seguire lo standard mostrato come esempio nel documento di analisi. In seguito al salvataggio del file il controllo torna alla schermata principale.


RIMESSE BANCARIE
Aggiorna le Fatture
Crea rimessa
Esci
Aggiorna fatture
Seleziona il file da leggere
Cerca

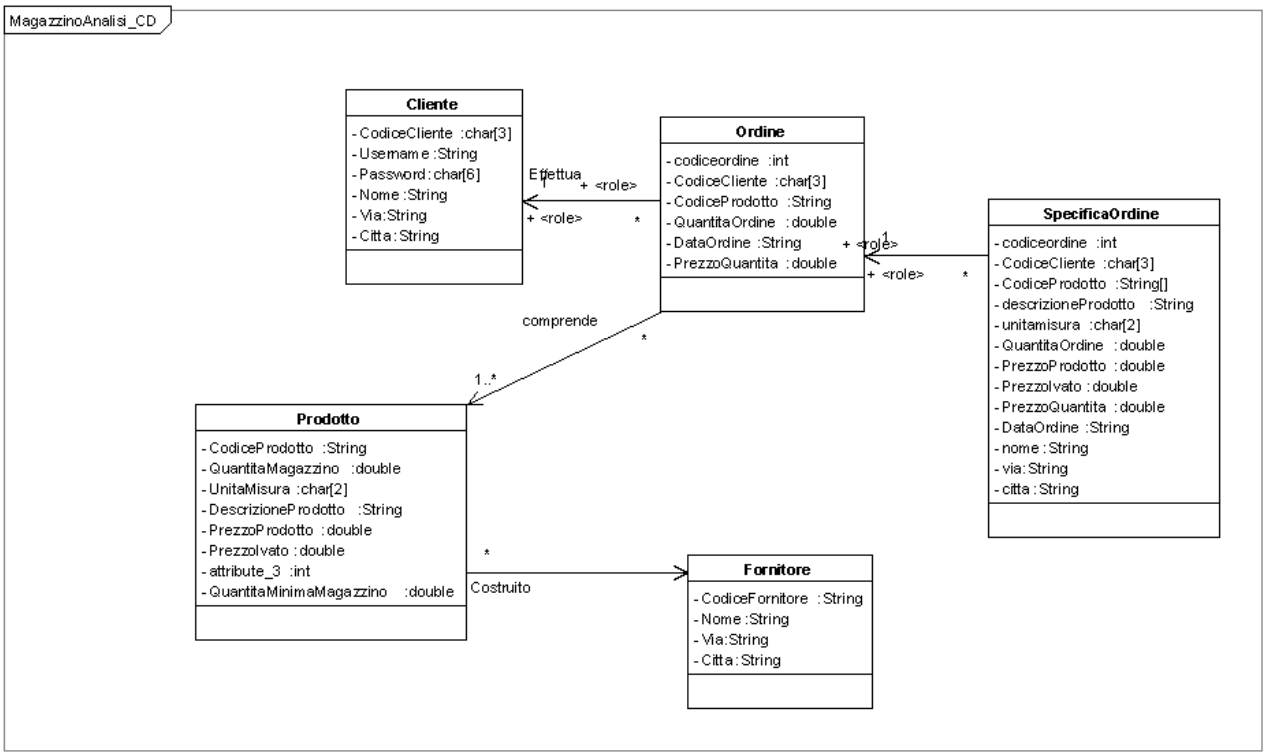
Definizione delle entità attraverso il Class Diagram di Analisi

In questa fase, che parte dal Glossario e dallo/dagli Use Case Diagram (corredati anche degli Activity Diagram), deve essere realizzato il diagramma delle classi di analisi. Tale diagramma deve indicare chiaramente tutte le classi entità, ossia le classi definibili come "proiezioni" nel dominio della applicazione software delle entità del dominio del problema dove la applicazione software andrà ad operare. Ad esse si possono aggiungere eventuali altre classi individuate nel corso dell'analisi, che siano di importanza per i concetti funzionali che definiscono i requisiti del progetto. Questo diagramma è l'equivalente da un punto di vista del ruolo (e l'evoluzione da un punto di vista storico e metodologico) del diagramma Entità-Relazioni (ER) usato nelle metodologie di sviluppo più tradizionali. In pratica entro il diagramma devono essere chiaramente indicate:

1. Tutte le classi entità che fanno parte del dominio del problema;
2. Gli attribuiti caratteristici di tali classi, eventualmente procedendo alla individuazione dei singoli attributi o dei gruppi che consentano una identificazione univoca delle istanze delle classi, ovvero dei singoli oggetti; tali attributi costituiscono le chiavi;
3. Le associazioni che tra tali classi intercorrono, ossia tutti i legami logici che tra esse intercorrono; queste associazioni (che corrispondono alle relazioni dei diagrammi ER) sono importanti perchè in sede implementativa di codice indicheranno anche la visibilità necessaria tra le classi, cioè quali altre classi (eventualmente appartenenti ad altri package o namespace) una certa classe dovrà vedere, definendo quindi la loro interdipendenza;
4. I versi di tali associazioni (ad esempio, se la classe magazzino deve conoscere la classe prodotto, non è sempre vero il viceversa);
5. Le molteplicità di tali associazioni (es. uno-a-molti, molti-a-molti), l'eventuale necessità di definire classi di associazione (si ricordi ad esempio la proprietà dell'auto che svolge il ruolo di classe di associazione fra proprietario ed auto);
6. Eventuali rapporti di inclusione legati a tali associazione, suddivisi fra aggregazione e composizione; si ricordi che l'eliminazione di una composizione, indicata con il diamante nero, elimina anche tutti i suoi elementi componenti, mentre l'eliminazione di una aggregazione, indicata con il diamante bianco, non elimina anche i componenti, che hanno anche una natura indipendente;
7. Eventuali rapporti di ereditarietà fra le classi, ottenuti applicando i principi di generalizzazione e specializzazione, ovvero "raccogliendo a fattor comune" attributi e metodi o aggiungendone di nuovi;
8. Si ricordi che da questo diagramma, eventualmente passando attraverso un diagramma EER, deriverà anche la base dati relazionale dell'applicazione: i rapporti di molteplicità devono essere chiari perchè dalle associazioni derivano le relazioni tra le chiavi che collegano le tabelle entro la base dati; tale derivazione, in base alle caratteristiche specifiche del DBMS che conterrà la base dati e alla necessità di ottimizzazione dei tempi di accesso ai dati stessi, potrebbe portare ad una implementazione della base dati anche molto diversa dal modello;
9. I metodi delle classi possono ancora non essere completamente definiti in questa fase.

Il processo che conduce al diagramma finale è ovviamente iterativo e può dirsi stabilizzato quando tutte le relazioni (in senso ampio) fra le classi sono chiaramente individuate. Il Class Diagram di Analisi descrive graficamente il dominio di business e le sue entità ed è fondamentale per tutti i passi di progetto che seguono. Normalmente vanno incluse anche descrizioni più o meno ampie delle caratteristiche delle classi e delle relazioni che tra loro intercorrono.

Esempio di Class Diagram di Analisi (modello del dominio business)
Gestione Magazzino - Autore: Fabio Bettinazzi



Design della base di dati

L'obiettivo deve essere la costruzione degli script per la generazione della base dati entro un DBMS relazionale, che dovranno essere salvati a parte.
In pratica si parte dal modello del dominio di business formalizzato attraverso il Class Diagram di Analisi, per giungere al modello di tabelle della base di dati. Si può eventualmente tradurre il Class Diagram in un diagramma Entità-Relazione (ER), oppure giungere per passaggi successivi alla forma finale.
Devono essere considerate le regole di ottimizzazione e l'insieme delle tabelle nella base dati può essere anche molto diverso dallo schema delle classi iniziale. Si vedano [ACPT 2002] e [Camuso5 2012] per approfondimenti.

Scelta architetturale e definizioni conseguenti

La scelta architetturale è un passo fondamentale, in quanto i passi successivi sono da essa condizionati. Esistono comunque regole generali importanti che aiutano nello svolgimento, quali il pattern Model-View-Controller ed il conseguente approccio multicanale alla realizzazione delle interfacce utenti. Seguendo tale metodo si separa nettamente la interfaccia utente vera e propria (View), che ha lo scopo di presentare semplicemente dati all'utente ed è ovviamente soggetta ai vincoli dal tipo di mezzo o canale utilizzato (interfaccia a finestre grafiche, Web, PDA, cellulare, Set-Top Box TV...), dal reattore agli eventi trasmessi dall'utente (Controller), che usa i metodi forniti dagli strati interni dell'applicazione (Model e relativi Adapter) per garantire all'utente i servizi associati agli eventi inviati dall'utente stesso. Grazie all'approccio multicanale, eventualmente corredato dall'uso di altri strati di Adapter, diviene possibile riutilizzare (almeno in buona parte) il controller (ed ovviamente gli strati sottostanti) cambiando solo la view quando si cambia canale, passando, ad esempio, da una applicazione Window ad una Web sostituendo alla finestra il servlet.
Nel Web in particolare, seguendo il metodo MVC-2, il servlet fa il ruolo di adapter del controller, al cui interno stanno poi gli adapter del model, mentre la view è implementata con una pagina JSP, permettendo riadattamenti grafici estremamente rapidi.
La scelta dell'architettura deve anche segnalare limiti e criticità nel sistema che sarà realizzato.
L'output di questa fase sono documenti tecnici architetturali, che saranno poi corredati da eventuali Component Diagram e Deployment Diagram solo al termine della fase di progetto vera e propria.
Per approfondimenti sulle architetture, si veda [DFP 2009].

Progettazione software e Class Diagram di Progetto

In questa fase occorre definire chiaramente tutte le classi che fanno parte dell'applicazione software da implementare. Il Class Diagram di Progetto è l'elenco completo delle classi, con tutte le loro relazioni e su di esso si basa anche il dimensionamento della fase di sviluppo (ovvero scrittura vera e propria del software).
Il processo che permette di giungere al diagramma delle classi di progetto è necessariamente iterativo. Si parte dal diagramma delle classi di analisi e devono essere inserite tutte le classi di servizio, ossia le classi infrastrutturali, non necessariamente derivate dalla fase di analisi, che permettono al programma nel suo insieme di operare

correttamente ed in modo efficiente. Le classi di servizio sono ovviamente fortemente dipendenti nella loro struttura dall'architettura scelta e da eventuali framework utilizzati nel progetto. Se un diagramma di analisi ben fatto può essere spesso utilizzato con diverse tecnologie ad oggetti, ovvero essere punto di partenza per progetti analoghi realizzati su piattaforme diverse, un diagramma di progetto è chiaramente molto più influenzato dalla tecnologia usata. Il processo usa anche altri diagrammi UML.

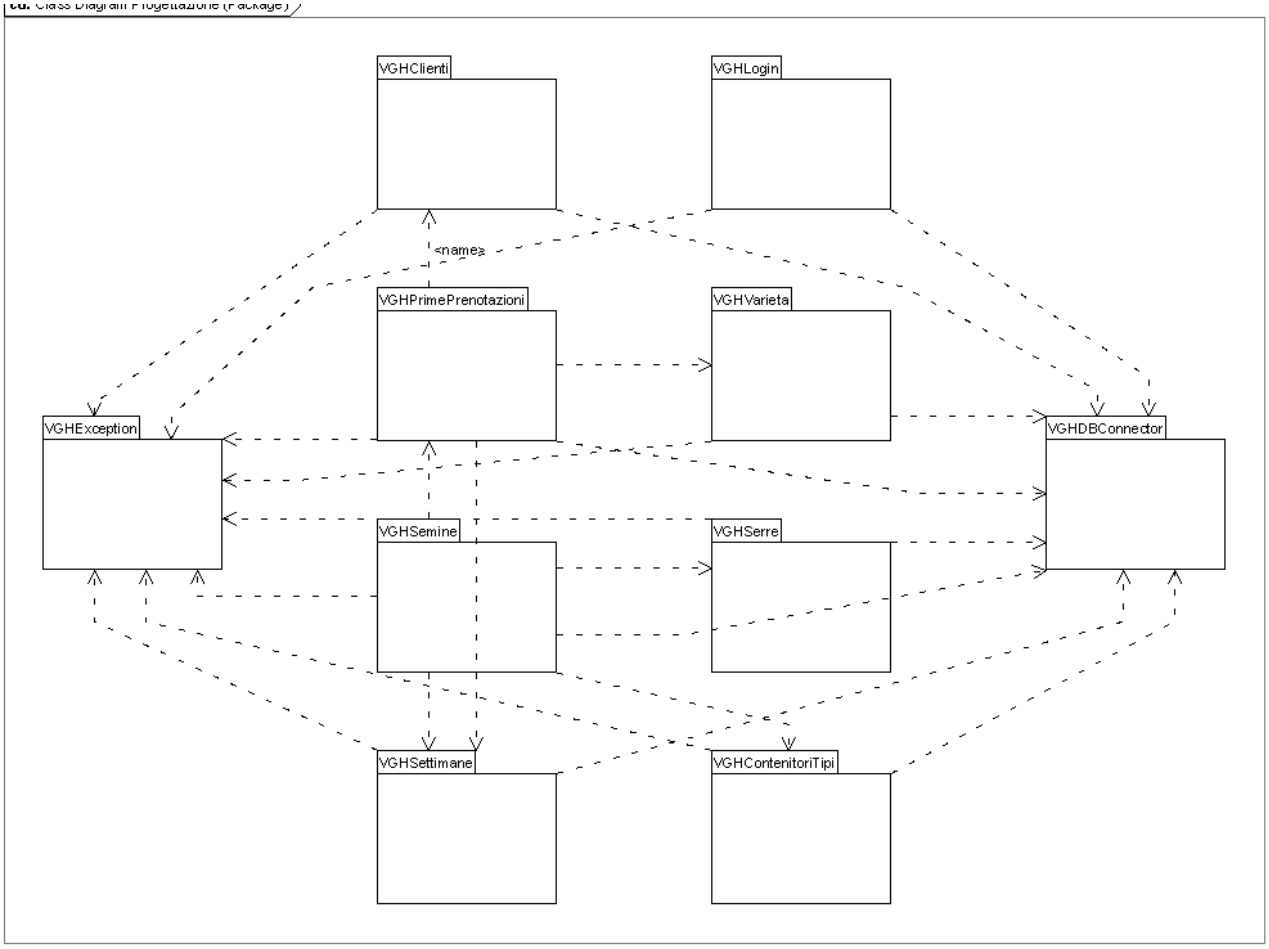
1. I diagrammi di interazione (sequence, che pone enfasi sulla sequenza temporale delle interazioni, e collaboration, che pone enfasi sulla dipendenza fra le classi) sono di importanza fondamentale sia per la definizione dei metodi che le classi offrono le une alle altre (e dei loro argomenti e valori di ritorno), sia per l'individuazione di eventuali "colli di bottiglia" che vengono risolti con l'inserimento di nuove classi. In teoria ad ogni use case corrisponde almeno un sequence o collaboration diagram: infatti ogni corso di eventi individuato nell'analisi con gli use case dovrebbe produrre una precisa sequenza temporale di invocazione di metodi all'interno dell'insieme delle classi costituenti il sistema software. Non sempre è però indispensabile realizzarli tutti, specie nei casi di corsi di eventi molto simili, nel qual caso bastano le opportune descrizioni di accompagnamento.
2. I diagrammi di attività, che derivano anch'essi dagli Use Case, dando ad essi una sequenza temporale e logica, possono aiutare molto nella definizione della Mappa di Navigazione fra le finestre, consentendo di definire completamente l'interfaccia utente di un applicativo ed eventualmente di realizzare i prototipi d'analisi (finestre vuote vere e proprie o gli schematics).
3. I diagrammi di stato sono anch'essi molto importanti per valutare l'evoluzione temporale delle singole classi (o meglio degli oggetti da esse istanziati) o di sotto-sistemi che esse vanno a costituire, aiutando ad individuare eventuali condizioni critiche o colli di bottiglia.

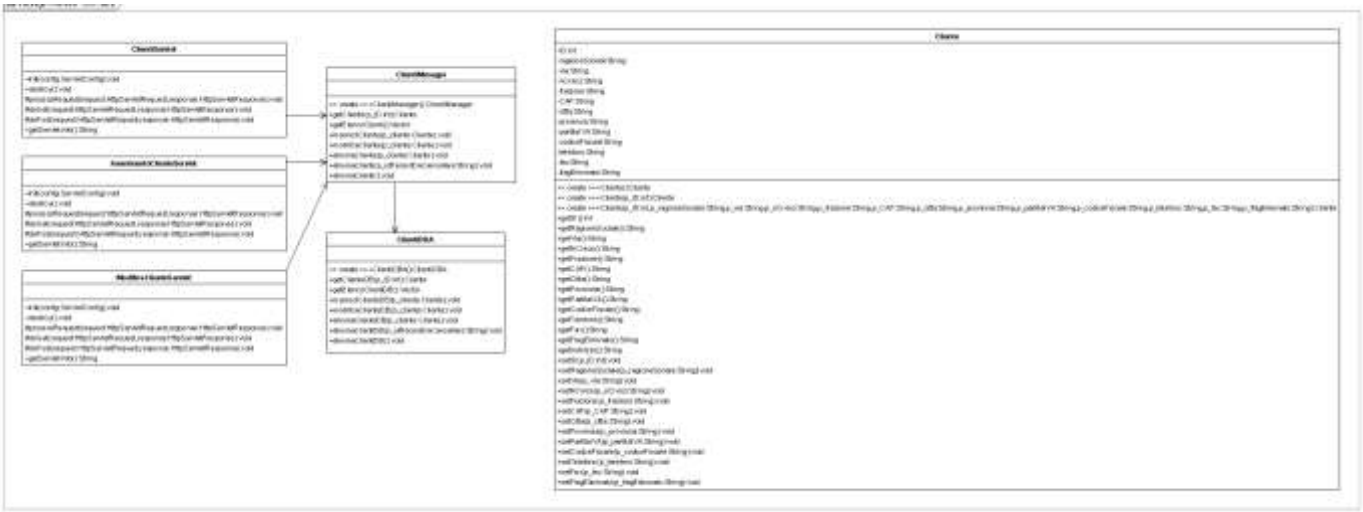
L'obiettivo finale è comunque la realizzazione del Class Diagram di Progetto, completo di tutte le classi. Spesso per motivi di chiarezza (specialmente in progetti grandi dove le classi sono molto numerose) il diagramma viene diviso in package, associazioni di classi corrispondenti ad unità funzionali, indicando esternamente ad essi solo i legami che fra i singoli package intercorrono. Ciascun package viene poi rappresentato completamente entro un diagramma di secondo livello. Quasi sempre questa suddivisione funzionale viene anche portata a livello implementativo servendosi delle aggregazioni tipiche dei linguaggi (package del Java, namespace di C#). L'obiettivo deve essere sempre quello di avere un diagramma leggibile, che serve come mappa per lo sviluppo. Da questo diagramma possono anche essere generati gli scheletri delle classi attraverso opportuni strumenti CASE (ad esempio ArgoUML), oppure essere ottenuti i Fogli di Specifica, ossia i documenti che descrivono ciascuna classe con attributi, metodi, vincoli e controlli da implementare.

Dal punto di vista didattico, il prodotto di questa fase deve essere il class diagram di progetto, corredato dalla descrizione di tutte le classi coinvolte. Qualora le classi siano molte conviene applicare una struttura a due livelli, suddividendo il progetto in package, come nell'esempio sotto riportato.

Esempi di Class Diagram di Progetto

Gestione Vivaio – Autore: Irene Bacchi



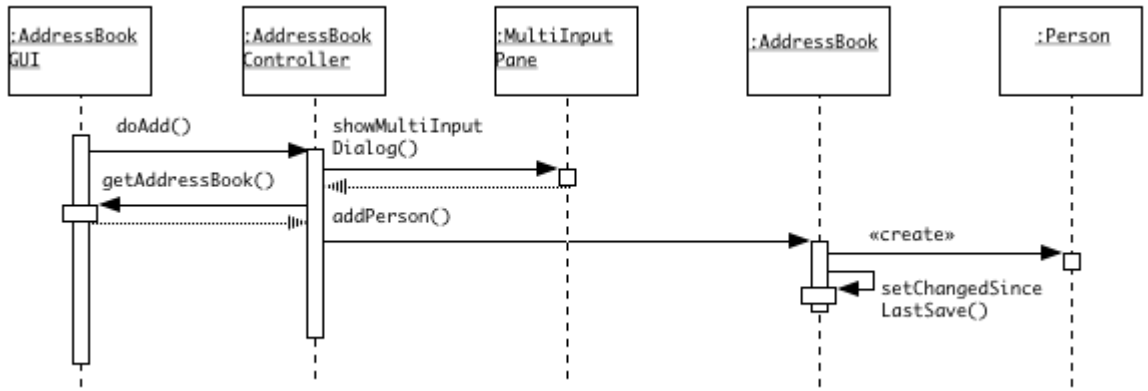


Esempi di Sequence Diagram

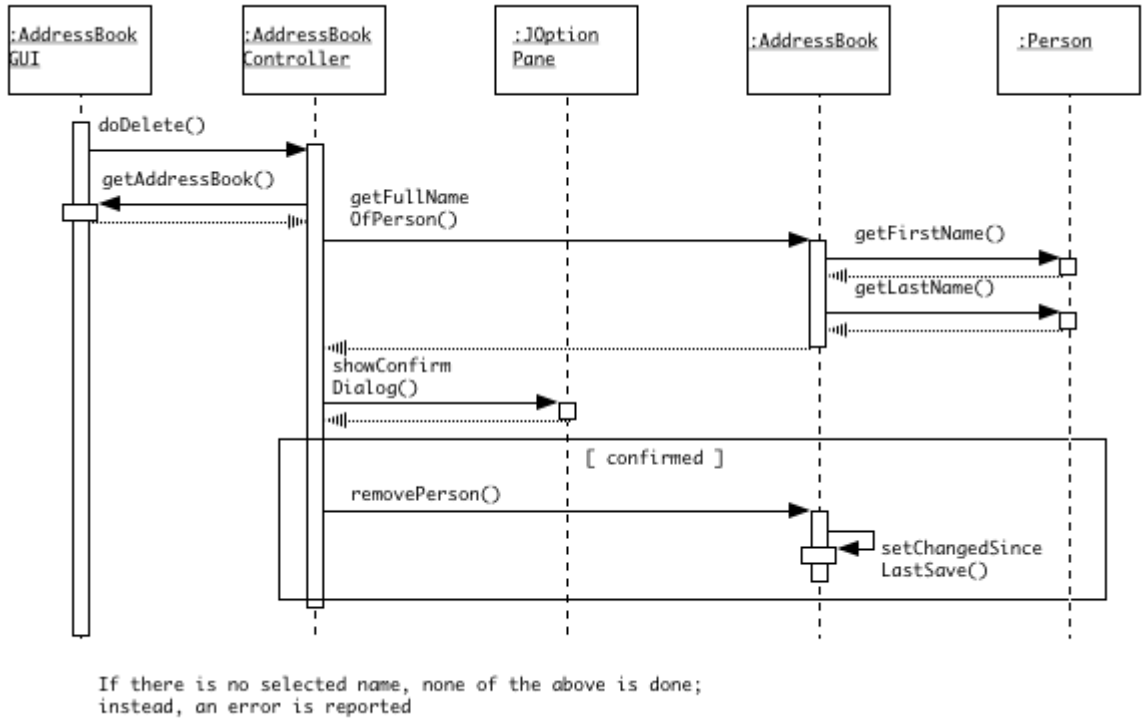
I seguenti esempi sono tratti da [Bjork 2005], che si raccomanda come esempio completo e ben fatto di progetto informatico con UML. Sono molto significativi in quanto rappresentano le interazioni tipiche con un programma di anagrafica: aggiunta nuovo elemento, cancellazione e modifica di elementi esistenti.

Le classi indicate che collaborano fra di loro per svolgere l'azione sono

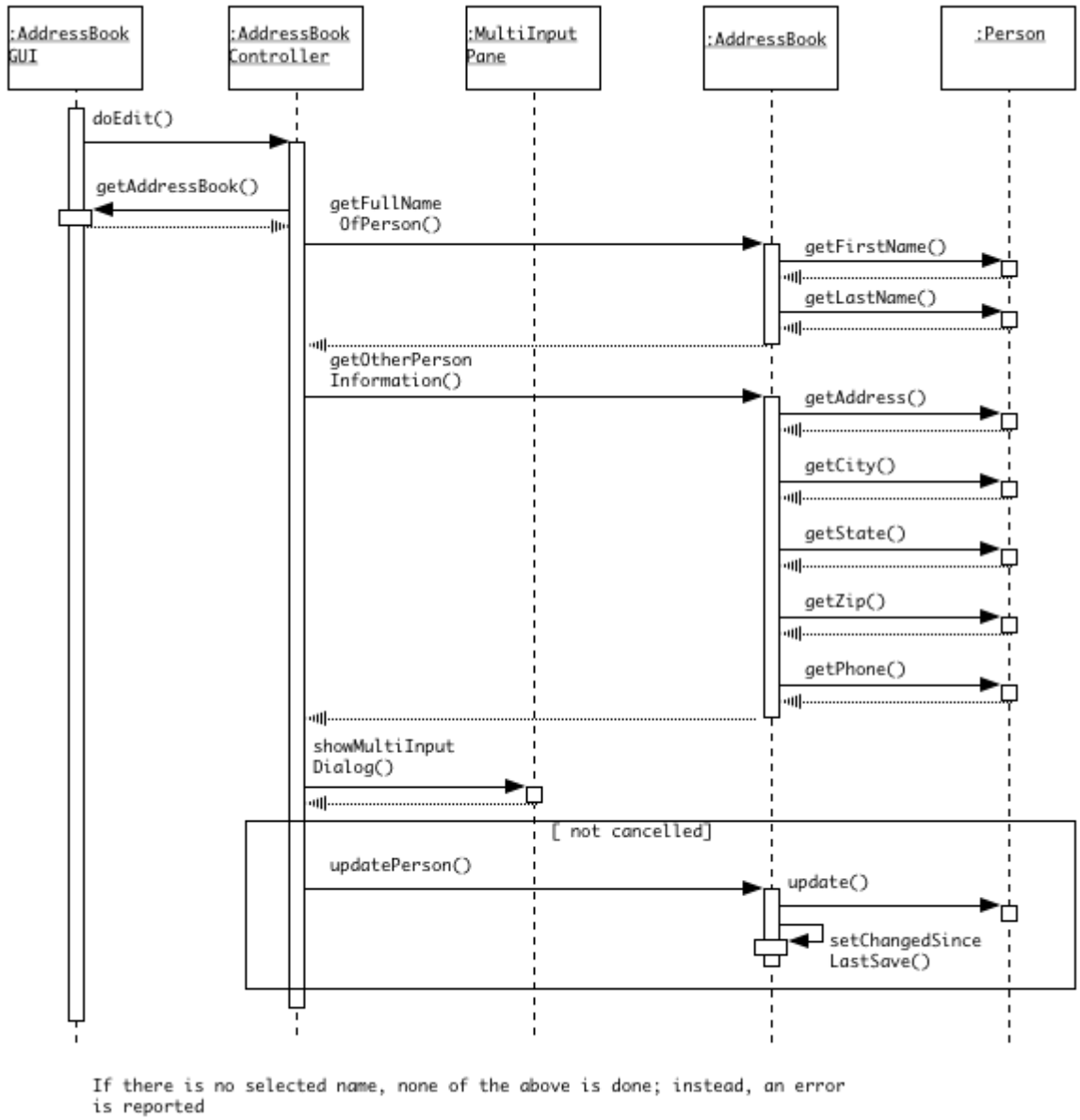
- GUI, Controller, Panel – classi che formano o gestiscono gli eventi della interfaccia utente;
- AddressBook – classe che rappresenta il "contenitore principale" dei dati;
- Person – classe entità che rappresenta la persona, ossia l'unità base della anagrafica.



Aggiunta nuovo elemento, ossia una nuova persona



Cancellazione elemento esistente, ossia persona esistente



Modifica di elemento esistente, ossia di persona esistente.

Bibliografia

[ACPT 2002] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone - Basi di Dati: Modelli e Linguaggi di Interrogazione – Ed. McGraw-Hill Italia, Milano, 2002

[ACFPT 2003] P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone - Basi di Dati: Architetture e Linee di Evoluzione – Ed. McGraw-Hill Italia, Milano, 2003

[BABoK 2009] International Institute of Business Analysis – *Business Analysis Body of Knowledge v.2.0,* 2009

[Bjork 2005] R. Bjork – Address Book project example – http://www.math-cs.gordon.edu/courses/cs211/AddressBookExample/Intro.html

[Camuso3 2012] F. Camuso – *Informatica per la classe Terza ITIS* http://www.camuso.it/3a/dispense/Dispense%20informatica%20III%202012-13.zip

[Camuso5 2012] F. Camuso – Materiali didattici di *Informatica per la classe Quinta ITIS* http://www.camuso.it/5A/index.asp

[Destri 2013] G. Destri. S*istemi Informativi: il pilastro digitale di aziende ed organizzazioni*. Ed. Franco Angeli, 2013

[DFP 2009] G. Destri, O.Figus, A. Picca – *AreaMVC: Linee guida per l'applicazione pratica di MVC in .NET e Java* – su Web http://www.areaprofessional.net/documenti/AreaMVC.pdf

[EP 2000] H.E. Eriksson, M. Penker - *Business Modeling with UML* - Ed. Wiley and Sons, 2000

[Fowler 2010] M. Fowler - *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Fourth Edition* – Ed. Addison-Wesley, 2010

[PMBoK 2008] PMI – *Guide to Project Management Body of Knowledge, 4th edition*, 2008, disponibile anche in italiano.

[PMIWBS 2008] PMI - *Practice Standard for Work Breakdown Structures, 2nd Edition, 2008*

[PRINCE 2009] *Managing Successful Projects with PRINCE2, 2009 edition* – Ed. TSO, 2009

[Web Agile] Sito Web dell'Agile Manifesto - http://agilemanifesto.org/iso/it/

[Web IPMA] Sito Web istituzionale della International Project Management Association – http://www.ipma.ch.

[Web OMG] Sito Web dell'*Object Management Group* - http://www.omg.org