

8 UNIFIED MODELING LANGUAGE (UML)	2
8.1 UN APPROCCIO VISUALE ALLA PROGETTAZIONE.	2
8.1.1 <i>I vantaggi dell'utilizzo di diagrammi nella fase di progettazione.</i>	2
8.2 COS'È UML	3
8.2.1 <i>Origini e breve storia di UML</i>	3
8.2.2 <i>Processo Unificato di sviluppo del software</i>	4
8.3 STRUTTURA DI UML.....	5
8.3.1 <i>Le viste</i>	5
8.3.2 <i>I diagrammi UML</i>	6
8.4 CASI D'USO	7
8.4.1 <i>Attore</i>	8
8.4.2 <i>Use case</i>	8
8.4.3 <i>Relazione tra attore e use case</i>	8
8.4.4 <i>Altri tipi di relazione ed associazione</i>	10
8.4.5 <i>Come produrre casi d'uso</i>	12
8.5 DIAGRAMMI DELLE CLASSI	13
8.5.1 <i>Diagrammi degli oggetti</i>	14
8.6 DIAGRAMMI DI COLLABORAZIONE	15
8.7 DIAGRAMMI DI SEQUENZA	16
8.8 DIAGRAMMI DI ATTIVITÀ	17
8.9 DIAGRAMMI DI STATO.....	18
8.10 DIAGRAMMI FISICI	19
8.10.1 <i>Diagrammi dei componenti</i>	19
8.10.2 <i>Diagrammi di deployment</i>	20

8 Unified Modeling Language (UML).

8.1 Un approccio visuale alla progettazione.

Per quale motivo è utile un approccio visuale alla progettazione?

Chi progetta un qualsiasi tipo di costruzione o artefatto utilizza sempre figure, schemi, diagrammi per svolgere la propria attività: ingegneri, architetti, ma anche stilisti utilizzano diagrammi e figure per visualizzare i propri progetti.

Anche i progettisti e gli analisti di sistemi informativi utilizzano figure e diagrammi per visualizzare il risultato del loro lavoro: il sistema software.

Ciò avviene anche se il tipo di prodotto finale che risulta dalla progettazione non è necessariamente visuale.

8.1.1 I vantaggi dell'utilizzo di diagrammi nella fase di progettazione.

Sia che si progetti un edificio sia che si progetti un sistema software il progettista ha la necessità di rappresentare i diversi aspetti del progetto, per fare ciò si utilizzano diagrammi differenti, ognuno focalizzato su uno o più aspetti.

Nel caso dell'edificio alcuni disegni rappresentano una visione completa da diversi punti di vista, altri alcuni particolari come ad esempio gli impianti tecnologici.

Allo stesso modo un diagramma può rappresentare i collegamenti tra le componenti di un sistema informativo, altri alcuni particolari come ad esempio la sequenza delle comunicazioni tra le componenti.

Si tratta di applicare il concetto di "astrazione" attraverso il quale una realtà anche molto complessa viene rappresentata semplificandola in un *modello*.

Durante le fasi di analisi e progettazione vengono generati dei modelli che consentono di identificare e separare le caratteristiche (utili al progetto) di un sistema reale (ad esempio una classe che modella l'oggetto cliente).

Il progettista dovrà quindi decidere quali caratteristiche sono rilevanti per il sistema che sta costruendo, inserirle nel modello e definire le relazioni tra gli elementi del modello.

Vi sono diversi tipi di relazione da considerare:

- Strutturali, tra elementi interdipendenti (ad esempio associazioni tra classi)
- Temporali, per rappresentare sequenze di eventi nel tempo (ad esempio messaggi sequenziali in un diagramma di interazione)
- Causa-effetto, per definire precondizioni ad una determinata funzione (ad esempio stati di un diagramma di stato)
- Organizzative, per raggruppare opportunamente elementi del sistema (ad esempio package di elementi)
- Evolutive, per rappresentare le derivazioni tra elementi del modello nel tempo (ad esempio dipendenze tra diagrammi del modello)

8.2 Cos'è UML

Unified Modeling Language (UML) è un linguaggio di modellazione visuale.

E' uno strumento per analisti e progettisti di sistemi orientati agli oggetti che consente di modellare, rappresentare e documentare sistemi software.

Non è un linguaggio di programmazione, non è uno strumento di CASE.

Vi sono strumenti di CASE che possono generare codice in diversi linguaggi a partire da modelli UML. Si tratta del codice di struttura di oggetti che poi richiedono da parte dello sviluppatore la scrittura manuale del codice che implementa i metodi.

UML non è una metodologia di sviluppo del software. Molte metodologie di analisi e progettazione, pur mantenendo diversi procedimenti, hanno standardizzato la loro notazione per rappresentare visivamente i modelli di un sistema con UML.

UML è un linguaggio, un insieme di elementi e di regole, di specifica formale.

Gli elementi sono forme grafiche (linee, rettangoli, ecc.) che rappresentano ciò che si sta modellando.

Le regole che spiegano come combinare gli elementi sono di tre tipi:

- sintassi astratta, espressa tramite diagrammi e linguaggio naturale
- regole sintattiche, espresse tramite Object Constraint Language (OCL) e linguaggio naturale
- semantica, espressa in linguaggio naturale con il supporto di diagrammi

8.2.1 Origini e breve storia di UML

La Rational Software Corporation e l'Object Management Group (OMG) hanno unificato gli elementi rappresentativi di tre diverse metodologie di rappresentazione di diagrammi orientati agli oggetti.

Le tappe principali dell'evoluzione di UML.

- 1990-1994, affermazione di tre metodologie per lo sviluppo ad oggetti: Rumbaugh (Object Modeling Technique - OMT), Booch, Jacobson (Object-Oriented Software Engineering – OOSE)
- 1994, unificazione delle metodologie Rumbaugh e Booch insieme alla Rational Software Corporation
- 1995, le due metodologie danno origine allo Unified Method
- 1995, unione della compagnia di Jacobson (Objectory) alla Rational, inizio dello sviluppo di UML
- 1996, formazione del consorzio UML Partners (Rational Software Corporation, IBM, HP, Microsoft, Oracle)
- 1997, versione 1.1 di UML che viene aggiunto alle tecnologie adottate dall'Object Management Group (OMG)
- 1998, versione 1.2 delle specifiche UML
- 2001, versione 1.4

UML è il linguaggio standard di modellazione più diffuso nello sviluppo di software a livello industriale.

E' uno standard in evoluzione riconosciuto dall'Organizzazione Internazionale per la Standardizzazione (International Organization for Standardization, ISO).

8.2.2 Processo Unificato di sviluppo del software

UML è un linguaggio che ha lo scopo di definire in modo formale un sistema, gli sviluppatori di UML hanno messo a punto per gli sviluppatori di sistemi anche un modo di procedere nel processo di sviluppo utilizzando UML: il Processo Unificato di sviluppo del software (Unified Software Development Process – USDP).

Il Processo Unificato coinvolge persone, progetti, strumenti, processi, prodotti: i partecipanti e gli sviluppatori coinvolti nel progetto di sviluppo di un sistema, seguono un determinato processo, utilizzando strumenti di ausilio nello sviluppo, generando prodotti software.

Il processo parte dai requisiti dell'utente, che vengono raccolti nei cosiddetti "casi d'uso", delle sequenze di esecuzione del sistema in grado di fornire un valore all'utente. Dai casi d'uso gli sviluppatori producono i modelli del sistema e le implementazioni che li realizzano.

Le caratteristiche del processo:

- architetto-centrico: l'architettura del sistema viene sviluppata in modo da soddisfare i requisiti dei casi d'uso più importanti (piattaforma e struttura, sottosistemi)
- iterativo: il progetto viene scomposto in sottoprogetti che costruiscono parti del sistema finale
- incrementale: il sistema viene costruito incrementalmente unendo le singole parti sviluppate nei sottoprogetti

Fasi del ciclo di vita (un progetto può essere costituito da più cicli): Inizio, elaborazione, costruzione, transizione (ogni fase può essere costituita da più iterazioni).

I prodotti intermedi del processo sono i modelli (astrazioni delle caratteristiche del sistema). Ogni modello definisce il sistema da un certo punto di vista, i principali:

- dei Casi d'Uso
- di Analisi
- di Progetto
- di Deployment
- di Implementazione
- di Test

8.3 Struttura di UML

In un approccio top-down all'UML si possono distinguere gli elementi fondamentali della sua struttura:

- Viste
- Diagrammi
- Elementi del modello

Le viste mostrano i differenti aspetti di un sistema attraverso la realizzazione di un certo numero di diagrammi, si tratta di astrazioni, ognuna delle quali analizza il sistema da modellare con un'ottica diversa (funzionale, non funzionale, organizzativa, ecc.), la somma di queste viste fornisce il quadro d'insieme.

I diagrammi permettono di esprimere le viste logiche per mezzo di grafici.

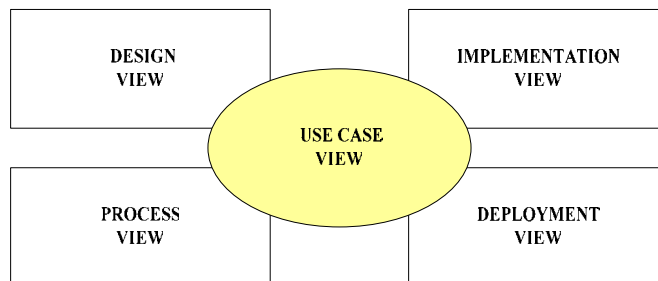
Vi sono diversi tipi di diagrammi destinati ad essere utilizzati ognuno per una particolare vista.

Gli elementi del modello sono i concetti che permettono di realizzare i vari diagrammi, indicano gli attori, le classi, i packages, gli oggetti, ecc.

8.3.1 Le viste

Si possono individuare cinque tipi di viste:

- Use Case view
- Design view
- Process view
- Implementation view
- Deployment view



La use case view serve per analizzare i requisiti utente: cosa il sistema dovrà fare.

Si tratta di una vista ad alto livello di importanza fondamentale, sia perché guida lo sviluppo delle rimanenti, sia perché stabilisce le funzionalità che il sistema dovrà realizzare.

Queste funzionalità potranno essere individuate assieme al cliente grazie proprio all'ausilio di tale vista che coadiuva il reperimento dei requisiti.

A questo livello di analisi, è opportuno studiare il sistema considerandolo come una scatola nera, concentrarsi sul cosa fare astraendosi il più possibile dal come verrà fatto.

Design view

La design view descrive come le funzioni debbono essere realizzate.

In questa vista si analizza il sistema dall'interno, si trova sia la struttura statica del sistema (diagramma delle classi e diagramma degli oggetti) e sia la collaborazione dinamica dovuta alle interazioni tra gli oggetti del sistema.

Implementation view

La implementation view descrive come il codice contenuto nelle classi viene aggregato in moduli (package) e le relative interdipendenze.

Process view

La process view serve per analizzare gli aspetti non funzionali del sistema, e consiste nell'individuare i processi.

Lo scopo è un utilizzo efficiente delle risorse, poter stabilire l'esecuzione parallela di determinati oggetti, poter gestire correttamente eventuali eventi asincroni.

Deployment view

La deployment view mostra l'architettura fisica del sistema e l'ubicazione delle componenti software all'interno della struttura stessa.

8.3.2 I diagrammi UML

I diagrammi di UML sono dei grafici che visualizzano una particolare proiezione del sistema analizzato da una specifica prospettiva.

Le varie tipologie sono di seguito elencate:

- Use Case Diagram (Diagramma dei casi d'uso);
- Class Diagram (Diagramma delle classi);
- Object Diagram (Diagramma degli oggetti);
- Sequence Diagram (Diagramma di sequenza);
- Collaboration Diagram (Diagramma di collaborazione);
- Statechart diagram (diagramma degli stati);
- Activity Diagram (Diagramma delle attività);
- Component diagram (diagramma dei componenti);
- Deployment diagram (diagrammi di dispiegamento).

8.4 Casi d'uso

I casi d'uso costituiscono un ottimo strumento per ottenere una visione d'insieme del sistema che si stà analizzando e sono quindi importanti nelle prime fasi del progetto.

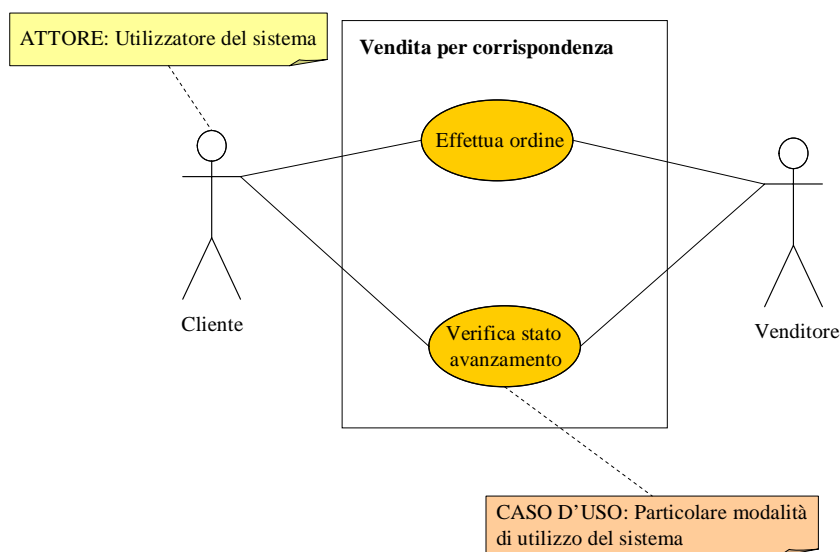
I casi d'uso rappresentano una vista esterna (utilizzatore) del sistema e sono finalizzati a modellare il dialogo tra utilizzatore e sistema:

- descrivono l'interazione tra attori e sistema, non la "logica interna" della funzione
- sono espressi in forma testuale, comprensibile anche per i non "addetti ai lavori"
- possono essere definiti a livelli diversi (sistema o parti del sistema)
- rappresentano le modalità di utilizzo del sistema da parte di uno o più utilizzatori (attori)

Ragionare sui casi d'uso aiuta a scoprire i requisiti funzionali: i casi d'uso possono essere un valido ausilio nel dialogo con l'utente ed in generale con esponenti non tecnici del progetto.

I diagrammi dei casi d'uso rappresentano i casi d'uso stessi, gli attori e le associazioni che li legano.

Esempio di diagramma di caso d'uso



Nei diagrammi dei casi d'uso sono rappresentate le modalità di utilizzo del sistema da parte di uno o più utilizzatori che vengono definiti attori.

I singoli use case non descrivono la "logica interna" delle funzioni ma si occupano di descrivere le iterazioni tra sistema e attori.

L'attore è un'entità esterna al sistema che fornisce lo stimolo a cui il sistema risponde.

8.4.1 Attore

L'attore è un utilizzatore del sistema (essere umano oppure altro sistema) che interagisce con i casi d'uso.

Rappresenta un ruolo di un oggetto oppure un oggetto esterno al sistema che interagisce con lo stesso come parte di un unità di lavoro (work unit) a realizzare un use case.

Un oggetto fisico (o classe) può giocare più ruoli differenti ed essere modellato da diversi attori.

Notazione



Esempi di attore per un sistema di prenotazioni: agente di viaggi, addetto check-in, ecc.

E' bene ricordare nel caso di attori umani che il nome di un attore identifica il ruolo che l'attore svolge in relazione al sistema, non il suo incarico.

Esempio: un impiegato (incarico) gestisce ed archivia un documento svolgendo il ruolo di protocollatore.

8.4.2 Use case

Il caso d'uso è una particolare modalità di utilizzo del sistema.

Rappresenta alcune funzioni visibili all'attore.

Raggiunge alcuni obiettivi per l'attore, in concreto può essere una scrittura, lettura, modifica di informazioni.

Notazione



Esempio per un sistema di prenotazioni: checking per un volo, assegnazione di un posto, ecc.

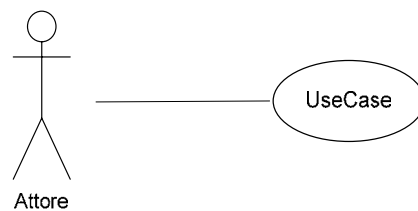
8.4.3 Relazione tra attore e use case

La relazione indica l'esistenza di un'associazione tra un attore ed un caso d'uso.

Quindi una particolare persona (o sistema) che si trova in un certo ruolo comunica con specifiche istanze del caso d'uso, partecipando agli eventi rappresentati dal caso.

In concreto poi il caso d'uso è realizzato come una funzione software utilizzata dagli attori trattando (inserendo o ricevendo) informazioni.

Nel diagramma gli attori sono collegati ai casi d'uso con i quali interagiscono attraverso una linea che rappresenta la relazione tra loro esistente.



Ogni caso d'uso costituisce una sequenza di attività che generano un risultato per l'attore che con esso interagisce.

La sequenza di attività viene descritta in una specifica comportamentale, che può consistere in un diagramma di sequenza, o di collaborazione, o di stato, oppure in istruzioni di un linguaggio di programmazione.

Normalmente viene prodotta una specifica informale nella forma di descrizione del caso d'uso.

La descrizione dei casi d'uso non ha una sintassi formale, a differenza di altri diagrammi UML con precise regole sintattiche, che rendono possibile la simulazione del sistema e la conversione del modello in un programma (C++ oppure Java).

La scelta delle modalità di scrittura della descrizione è quindi lasciata al progettista, normalmente in un certo contesto (ad es. software house) vi sono delle regole alle quali attenersi.

I due approcci più diffusi per la descrizione:

- uno o più paragrafi che descrivono la sequenza di attività che si verifica nel caso d'uso
- elencare, su due colonne, le attività compiute dall'attore e le risposte date dal sistema a tali attività

8.4.4 Altri tipi di relazione ed associazione

Altri tipi di associazioni e relazioni rappresentabili nel diagramma dei casi d'uso:

- generalizzazione tra casi d'uso

- generalizzazione tra attori
- relazione di inclusione (include) tra casi d'uso
- relazione di estensione (extend) tra casi d'uso

Generalizzazione tra casi d'uso.

Può esistere più di una versione di un caso d'uso, aventi ognuna alcune azioni in comune ed altre uniche per ciascun caso.

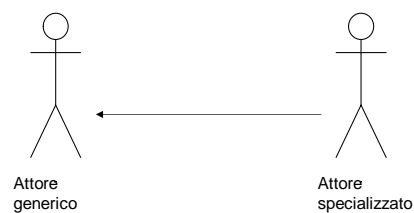
Nel diagramma l'associazione di generalizzazione viene indicata da una freccia puntata verso il caso d'uso più generale



Il caso d'uso specializzato eredita (vedi diagrammi delle classi) parte delle funzionalità dal caso d'uso generico.

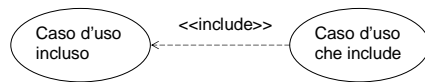
Generalizzazione tra attori.

L'attore specializzato eredita parte delle caratteristiche dell'attore generico.



Relazione di inclusione (include) tra casi d'uso.

Un caso d'uso comprende le funzionalità di un altro caso



Relazione di estensione (extend) tra casi d'uso.

Un caso d'uso può essere esteso nella sua funzionalità ad un altro caso



8.4.5 Come produrre casi d'uso

Nelle prime fase di progetto il progettista/analista lavora tipicamente su appunti e trascrizioni da interviste.

Normalmente il processo di analisi e definizione dei casi d'uso avviene per fasi.

- Definire attori e casi d'uso
 - tipiche domande alle quali dare risposta:
 - *Chi sono le persone che utilizzeranno questo sistema per inserire informazioni?*
 - *Chi sono i destinatari delle informazioni fornite dal sistema?*
 - *Quali altri sistemi interagiscono con questo?*
- Organizzare secondo ordine di priorità i casi d'uso
 - I casi d'uso più importanti vanno sviluppati prima
- Sviluppare ciascun caso d'uso (secondo priorità)
 - Generare la specifica dettagliata di ogni caso d'uso (analista del sistema)
- Strutturare il modello dei casi
 - Aggiunta della struttura al diagramma, attraverso generalizzazione, inclusione, estensione ed attraverso il raggruppamento dei casi in "package".

8.5 Diagrammi delle classi

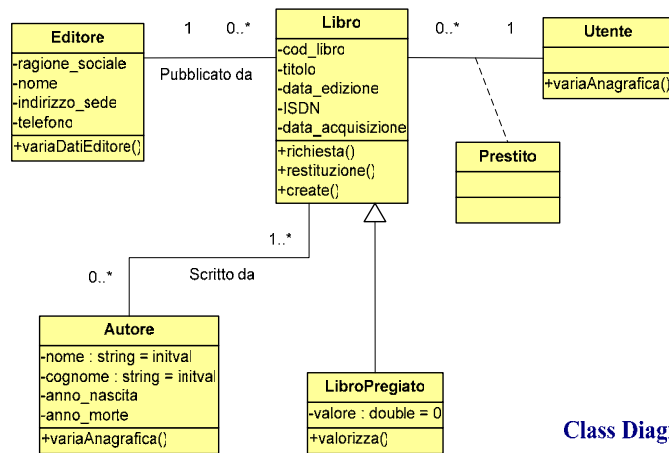
I diagrammi delle classi rappresentano le classi e gli oggetti, con i relativi attributi ed operazioni, che compongono il sistema.

L'obiettivo di tali diagrammi è visualizzare la parte statica del sistema.

Il diagramma delle classi specifica, mediante le associazioni, i vincoli che legano tra loro le classi.

Può essere definito a diversi livelli (analisi, disegno di dettaglio).

Può rappresentare diverse tipologie di oggetti (boundary, control, entità ...).



Class Diagram

8.5.1 Diagrammi degli oggetti

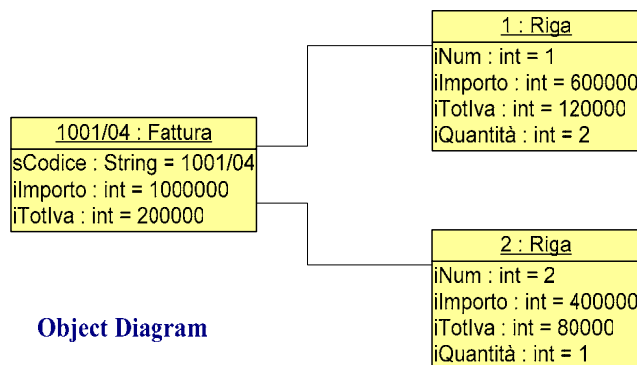
Il diagramma degli oggetti rappresenta una variante del diagramma delle classi, la notazione utilizzata è pressoché equivalente, con le sole differenze che i nomi degli oggetti vengono sottolineati e le relazioni vengono dettagliate.

Il diagramma degli oggetti mostra un numero di oggetti istanze delle classi e i relativi legami espliciti.

Anche questa tipologia di diagramma si occupa della parte statica del sistema.

Mentre un diagramma delle classi è sempre valido, un diagramma degli oggetti rappresenta un'istantanea del sistema.

In sostanza lo si utilizza per esemplificare diagrammi delle classi, o suoi frammenti, ritenuti poco chiari o particolarmente complicati.



8.6 Diagrammi di collaborazione

Il diagramma di collaborazione modella il comportamento dinamico del sistema.

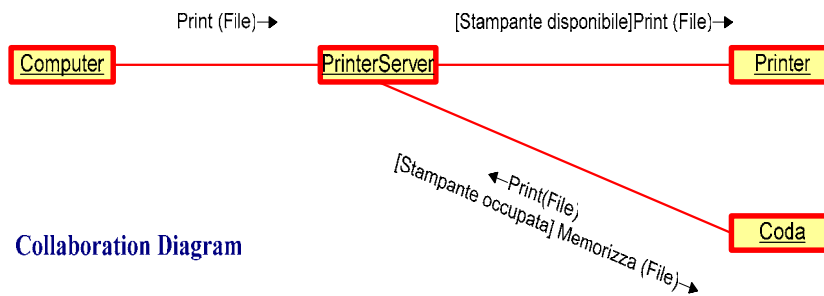
Il diagramma di collaborazione specifica gli oggetti (ed i messaggi che questi si indirizzano) che collaborano tra loro in un certo scenario.

Vengono evidenziati i legami tra gli oggetti, la sequenza dei messaggi viene specificata nel diagramma di sequenza.

Può essere utilizzato a diversi livelli (analisi, disegno) e rappresentare diverse tipologie di oggetti.

Diagrammi di collaborazione e diagrammi di sequenza sono molto simili, si può passare agevolmente dall'una all'altra rappresentazione (sono isomorfi), e si differenziano per l'aspetto dell'interazione che enfatizzano.

I diagrammi di collaborazione mettono in risalto l'organizzazione degli oggetti che si scambiano i messaggi; la sequenza dei messaggi è meno evidente che nel diagramma di sequenza, mentre sono più evidenti i legami tra gli oggetti.



Collaboration Diagram

8.7 Diagrammi di sequenza

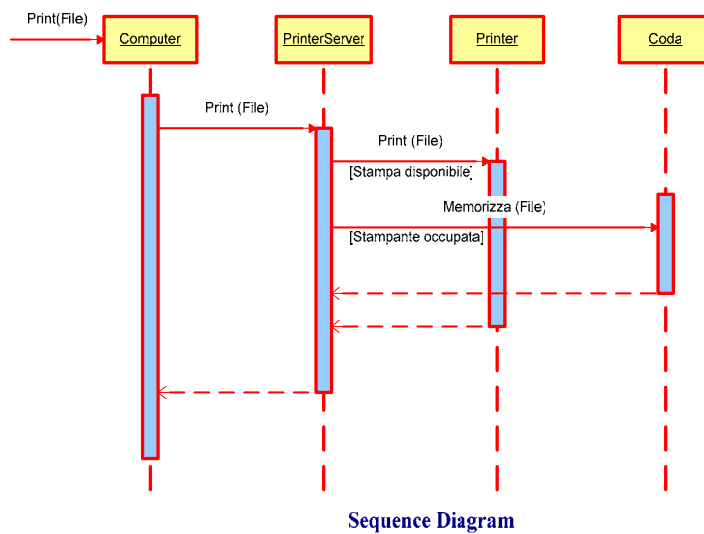
Il diagramma di sequenza evidenzia il modo in cui uno scenario (uno specifico percorso in un caso d'uso) viene risolto dalla collaborazione tra un insieme di oggetti.

Specifica la sequenza dei messaggi scambiati tra gli oggetti.

Può specificare nodi decisionali e iterazioni.

Questo tipo di diagramma viene anche detto di interazione in quanto mostra le interazioni tra oggetti del sistema, modella il comportamento dinamico del sistema, evidenziando in particolare l'ordine temporale dello scambio di messaggi.

Il diagramma di sequenza ed il diagramma di collaborazione esprimono informazioni simili, ma con diversa evidenza sui rispettivi dettagli.



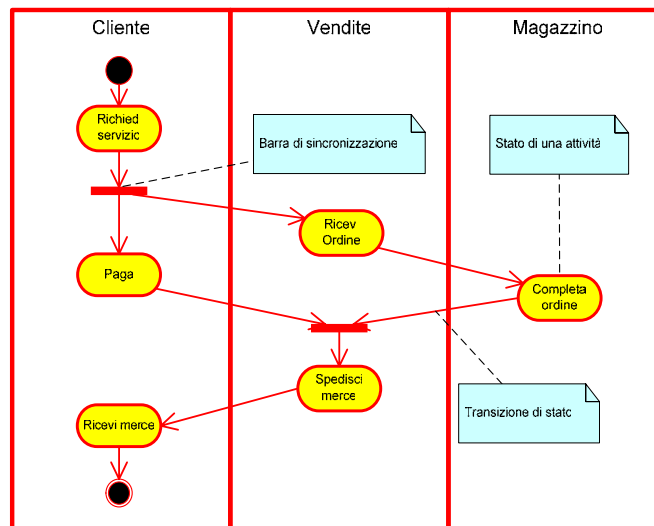
8.8 Diagrammi di attività

Il diagramma di attività rappresenta sistemi di workflow, la logica interna di un processo a diversi livelli (di business, di dettaglio).

Anche se dissimili nell'aspetto sono l'evoluzione dei familiari flow chart, e, come tali, si occupano di evidenziare il flusso di attività.

Permette la rappresentazione di processi paralleli e della relativa sincronizzazione.

Il diagramma di attività è un caso particolare del diagramma di stato nel quale ogni stato è uno stato di attività.



Activity Diagram

8.9 Diagrammi di stato

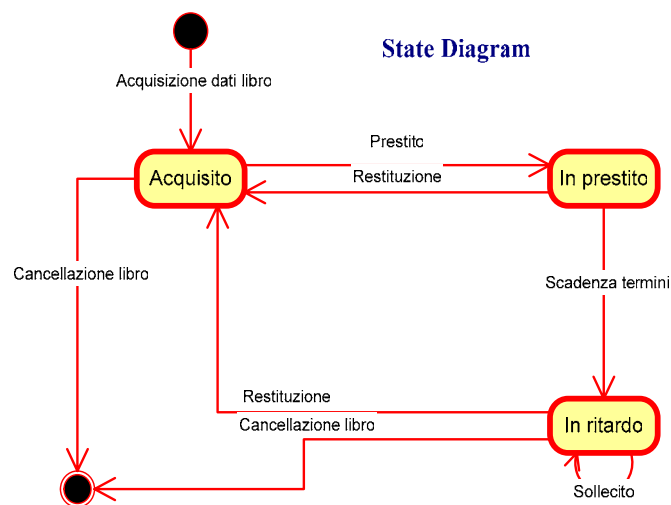
Il diagramma di stato specifica il ciclo di vita degli oggetti di una classe e definisce le regole che lo governano.

Quando un oggetto è in un certo stato può essere interessato da determinati eventi (e non altri), un evento può far passare l'oggetto ad un nuovo stato (transizione).

Il diagramma di stato mostra un automa a stati finiti, è pertanto costituito da stati, transizioni, eventi e attività.

Viene utilizzato principalmente come completamento della descrizione delle classi.

Si occupa di modellare la parte dinamica del sistema, e viene utilizzato per illustrare in dettaglio il comportamento delle sole classi che possono transitare per gli elementi di un ben definito insieme di stati.



8.10 Diagrammi fisici

I diagrammi fisici servono alla modellazione degli aspetti implementativi di un sistema informatico e si distinguono in:

- diagrammi dei componenti
- diagrammi di deployment

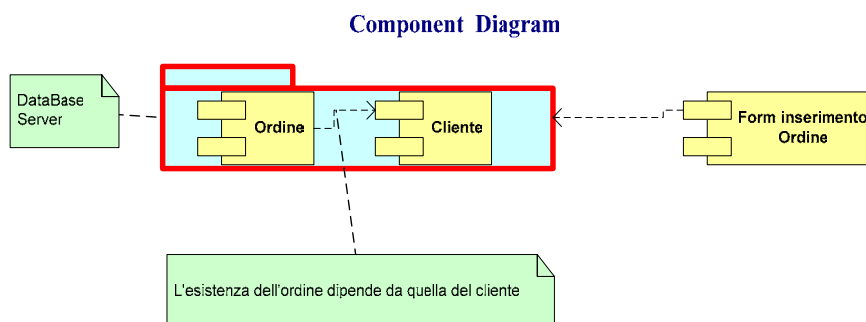
8.10.1 Diagrammi dei componenti

Il diagramma dei componenti evidenzia l'organizzazione delle componenti e le dipendenze tra le stesse.

I componenti sono moduli software, dotati di identità e con un'interfaccia ben specificata.

I componenti possono essere raggruppati in package (come a livello logico le classi).

Il diagramma dei componenti è utile nel rappresentare la vista statica del sistema ed è pertanto strettamente connesso al diagramma delle classi.



8.10.2 Diagrammi di deployment

Il diagramma di distribuzione permette la rappresentazione dell'architettura fisica del sistema (hardware e software) a diversi livelli di dettaglio.

In particolare illustra gli elementi fisici del sistema (computer, reti, device fisici,...) e i moduli software da installare su ciascuno di essi.

Il diagramma di distribuzione evidenzia la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time), dei componenti, dei processi, degli oggetti localizzati nei nodi.

