

Analisi Specifica dei requisiti

- Obiettivo
 - Specificare (cioè definire) le proprietà che il sistema dovrà avere senza descrivere una loro possibile realizzazione
- Approccio funzionale
Astrae sino al massimo livello il modo di funzionare di un calcolatore
- Approccio orientato agli oggetti
Si basa sulla stessa forma di astrazione applicata dagli uomini per affrontare la complessità del mondo
- Principio fondamentale: **Astrazione**
Permette di gestire la complessità intrinseca del mondo reale
 - Ignorare gli aspetti che non sono importanti per lo scopo attuale
 - Concentrarsi maggiormente su quelli che lo sono

Ingegneria del Software L-A

2.50

Analisi Approccio Funzionale

- Tecnica della Scomposizione Funzionale
- Strategia
 - Analisi *top-down* ► scegliere i passi ed i sotto-passi di elaborazione previsti per il sistema
 - Astrazione procedurale ► considerare ogni operazione come una singola entità, nonostante tale operazione sia effettivamente realizzata da un insieme di operazioni di più basso livello
- L'analista
 - concentra la sua attenzione sulla modularizzazione delle procedure – i moduli devono essere il più possibile indipendenti
 - specifica le elaborazioni e le interfacce delle procedure
- La scomposizione in funzioni è molto volatile (a causa del continuo cambiamento dei requisiti funzionali)

Ingegneria del Software L-A

2.51

Analisi Approccio Orientato agli Oggetti

- Si basa sugli oggetti che sono molto più stabili delle funzioni
 - Produce una specifica più resistente ai cambiamenti
- Fornisce una base efficace per
 - l'analisi del dominio
 - il riuso
- Fornisce una rappresentazione omogenea analisi ► progettazione ► programmazione in quanto si utilizzano
 - gli stessi principi
 - la stessa notazione
 - gli stessi oggetti

Ingegneria del Software L-A

2.52

Modello dei dati

- Analisi del dominio del problema al fine di individuare
 - Oggetti e classi rilevanti per il sistema da sviluppare
 - Limitarsi esclusivamente a quelle classi che fanno parte del vocabolario del dominio del problema
 - Relazioni tra le classi
 - Per ogni classe
 - Responsabilità
 - Attributi
 - Operazioni fondamentali cioè servizi forniti all'esterno (interfaccia)
- Attività non strettamente sequenziali, ma reiterate sino alla produzione di un modello coerente

Ingegneria del Software L-A

2.53

Modello dei dati

- Raggruppamento delle classi in sottosistemi
- Documentazione
 - Diagrammi delle classi e dei sottosistemi
 - Diagrammi di collaborazione tra le classi (opzionali)
 - Per ogni classe, descrizione che ne specifica scopo, responsabilità, attributi, operazioni
 - Per ogni attributo e ogni operazione, descrizione testuale accurata

Ingegneria del Software L-A

2.54

Analisi Individuazione delle classi

- Due analisti non produrranno mai due modelli delle classi identici per lo stesso dominio applicativo
- La letteratura è ricca di approcci raccomandati per l'individuazione delle classi
 - Approccio basato sulle frasi nominali
 - Approccio guidato dai casi d'uso
 - Approccio CRC
- La cosa migliore è usare un approccio misto

Ingegneria del Software L-A

2.55

Analisi Individuazione delle classi

- Fonti principali
 - Documento dei requisiti
 - Altri documenti di tutti i tipi che descrivono il sistema
- Altre fonti
 - Altri sistemi che funzionano nello stesso dominio o in domini analoghi
 - Enciclopedie, nomenclature e documenti tecnici che descrivono il dominio
- Riutilizzare classi, gerarchie e strutture ottenute da precedenti analisi nello stesso dominio

Ingegneria del Software L-A

2.56

Analisi Individuazione delle classi

- Elenare i nomi (semplici o composti) che compaiono nei documenti raccolti, convertendoli al singolare
- Eliminare i nomi che sicuramente
 - non si riferiscono a classi
 - indicano attributi (dati di tipo primitivo)
 - indicano operazioni
- Scegliere un solo termine significativo se più parole indicano lo stesso concetto (sinonimi)
- Il nome della classe deve essere un nome familiare
 - all'utente o
 - all'esperto del dominio del problema
 - non allo sviluppatore!

Ingegneria del Software L-A

2.57

Analisi Individuazione delle classi

- Attenzione agli aggettivi e agli attributi, possono
 - Indicare oggetti diversi
 - Indicare usi diversi dello stesso oggetto
 - Essere irrilevanti
- Ad esempio:
- "Studiante bravo" potrebbe essere irrilevante
 - "Studiante fuori corso" potrebbe essere una nuova classe
- Attenzione alle frasi passive, impersonali o con soggetti fuori dal sistema devono essere rese attive ed esplicite, perché potrebbero mascherare entità rilevanti per il sistema in esame

Ingegneria del Software L-A

2.58

Analisi Individuazione delle classi

- Individuare Attori con cui il sistema in esame deve interagire
 - Persone
Docente, Studente, Esaminatore, Esaminando, ...
 - Sistemi esterni
ReteLocale, Internet, DBMS, ...
 - Dispositivi
attuatori, sensori, ...
- Individuare Modelli e loro elementi specifici, cioè oggetti descrittivi e istanze specifiche
 - Insegnamento – "Ingegneria del Software L-A"
 - CorsoDiStudio – "Ingegneria Informatica"
 - Facoltà – "Ingegneria"

Ingegneria del Software L-A

2.59

Analisi Individuazione delle classi

- Individuare Cose tangibili, cioè oggetti reali appartenenti al dominio del problema
 - Banco, LavagnaLuminosa, Schermo, Computer, ...
- Individuare Contenitori (fisici o logici) di altri oggetti
 - Facoltà, Dipartimento, Aula, SalaTerminali, ...
 - ListaEsame, CommissioneDiLaurea, OrdineDegliStudi, Window, Form, ...
- Individuare Eventi o Transazioni che il sistema deve gestire e memorizzare
 - possono avvenire in un certo istante (ad es., una vendita) o
 - possono durare un intervallo di tempo (ad es., un affitto)
 - Appello, EsameScritto, Registrazione, AppelloDiLaurea, ...

Ingegneria del Software L-A

2.60

Analisi Individuazione delle classi

- Per determinare se includere una classe nel modello, porsi le seguenti domande:
 - il sistema deve interagire in qualche modo con gli oggetti della classe?
 - utilizzare informazioni (attributi) contenute negli oggetti della classe
 - utilizzare servizi (operazioni) offerti dagli oggetti della classe
 - quali sono le responsabilità della classe nel contesto del sistema?

Ingegneria del Software L-A

2.61

Analisi Individuazione delle classi

- Attributi e operazioni devono essere applicabili a tutti gli oggetti della classe
- Se esistono
 - attributi con un valore ben definito solo per alcuni oggetti della classe e/o
 - operazioni applicabili solo ad alcuni oggetti della classe siamo in presenza di ereditarietà
- Esempio: dopo una prima analisi, la classe *Studente* potrebbe contenere un attributo *booleano* *inCorso*, ma un'analisi più attenta potrebbe portare alla luce la gerarchia:
 - Studente
 - StudenteInCorso
 - StudenteFuoriCorso

Ingegneria del Software L-A

2.62

Analisi Individuazione delle responsabilità

- Responsabilità di una classe
 - Descrizione ad alto livello dello scopo per cui è stata definita la classe
 - Vengono descritti solo i servizi disponibili pubblicamente, non quelli privati interni alla classe, la cui definizione sarebbe prematura
- Obiettivo – aiutare nell'identificazione di
 - Classi
 - Attributi
 - Operazioni
 - Relazioni tra le classi

Ingegneria del Software L-A

2.63

Analisi Individuazione delle responsabilità

- Principali sorgenti di informazione
 - Documento dei requisiti
 - Classi già identificate
- Cercare verbi che rappresentano azioni fatte da un oggetto del sistema
- Utilizzare le classi già identificate per il semplice fatto di esistere, devono avere almeno una responsabilità
- Annotare tutte le informazioni che gli oggetti devono mantenere e gestire

Ingegneria del Software L-A

2.64

Analisi Individuazione delle responsabilità

- Una volta identificate, le responsabilità vanno assegnate alle classi in base ai seguenti criteri:
 - Distribuire le responsabilità in modo bilanciato (molti oggetti che si scambiano messaggi)
 - Assegnare le responsabilità al livello più generale possibile salendo la gerarchia delle classi
 - Mantenere il comportamento collegato alle informazioni ad esso necessarie

Ingegneria del Software L-A

2.65

Analisi Individuazione delle responsabilità

- Metodo di analisi CRC (*Class-Responsibility-Collaboration*) di Cunningham e Beck

Nome della classe	
Lista di responsabilità	Lista di collaboratori

- I collaboratori sono le altre classi con le quali la classe in esame deve interagire
- Si utilizzano schede di cartoncino di 10x15 cm

Ingegneria del Software L-A

2.66

Analisi Individuazione delle relazioni

- La maggior parte delle classi (degli oggetti) interagisce con altre classi (altri oggetti) in vari modi
- Quando si modella un sistema è necessario individuare:
 - Non solo le entità coinvolte nel sistema
 - Ma anche le relazioni tra tali entità
- Una relazione (*relationship*) è una connessione tra entità

Ingegneria del Software L-A

2.67

Analisi Individuazione delle relazioni

- Nella modellazione *object-oriented* le relazioni più importanti sono:
 - Ereditarietà
 - Associazione
 - Associazione generica
 - Aggregazione
 - Composizione
 - Dipendenza
 - Collaborazione (relazione usa)
 - Relazione Istanza – Classe
 - Istanza di classe generica – Classe generica (istanza di template – template)
 - Relazione Classe – Metaclassa

Ingegneria del Software L-A

2.68

Analisi Individuazione delle relazioni

- In ogni tipo di relazione, esiste un cliente C che **dipende** da un fornitore di servizi F
- C ha bisogno di F per lo svolgimento di alcune funzionalità che C non è in grado di effettuare autonomamente
- Conseguenza per il corretto funzionamento di C è indispensabile il corretto funzionamento di F

Ingegneria del Software L-A

2.69

Analisi Individuazione delle relazioni

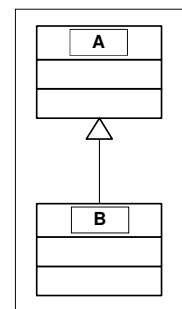
Tipo di relazione	Cliente	Fornitore
Ereditarietà	sottoclasse	superclasse
Associazione	contenitore	contenuto
Dipendenza	classe dipendente (che usa)	classe da cui si dipende (che viene usata)
	istanza	classe
	classe	metaclassa

Ingegneria del Software L-A

2.70

Analisi Individuazione dell'ereditarietà

- La classe B (*specializzazione o sottoclasse*) è in relazione isA con la classe A (*generalizzazione o superclasse*) e ne *eredita*
 - Relazioni
 - Attributi
 - Operazioni



Ingegneria del Software L-A

2.71

Analisi Individuazione dell'ereditarietà

- L'ereditarietà deve rispecchiare una tassonomia effettivamente presente nel dominio del problema
 - Non usare l'ereditarietà dell'implementazione (siamo ancora in fase di analisi!)
 - Non usare l'ereditarietà solo per riunire caratteristiche comuni ad es., Studente e Dipartimento hanno entrambi un indirizzo, ma non per questo c'è ereditarietà!
- La ricerca delle relazioni di ereditarietà contribuisce a chiarire il significato delle varie classi e può portare alla scoperta di nuove classi

Ingegneria del Software L-A

2.72

Analisi Individuazione delle associazioni

- Un'associazione rappresenta una relazione strutturale tra due istanze di classi diverse o della stessa classe
- Un'associazione può
 - Rappresentare un contenimento logico (aggregazione)
 - Una lista d'esame contiene degli studenti
 - Rappresentare un contenimento fisico (composizione)
 - Un triangolo contiene tre vertici
 - Non rappresentare un reale contenimento
 - Una fattura si riferisce a un cliente
 - Un evento è legato a un dispositivo

Ingegneria del Software L-A

2.73

Analisi Individuazione delle associazioni

- **Aggregazione**
Un oggetto x di classe X è associato a (contiene) un oggetto y di classe Y in modo non esclusivo x può condividere y con altri oggetti anche di tipo diverso (che a loro volta possono contenere y)
- Una lista d'esame contiene degli studenti
 - Uno studente può essere contemporaneamente in più liste d'esame
 - La cancellazione della lista d'esame non comporta l'eliminazione "fisica" degli studenti in lista

Ingegneria del Software L-A

2.74

Analisi Individuazione delle associazioni

- **Composizione**
Un oggetto x di classe X è associato a (contiene) un oggetto y di classe Y in modo esclusivo y esiste solo in quanto contenuto in x
- Un triangolo contiene tre punti (i suoi vertici)
 - L'eliminazione del triangolo comporta l'eliminazione dei tre punti
- Se la distruzione del contenitore comporta la distruzione degli oggetti contenuti, si tratta di composizione, altrimenti si tratta di aggregazione

Ingegneria del Software L-A

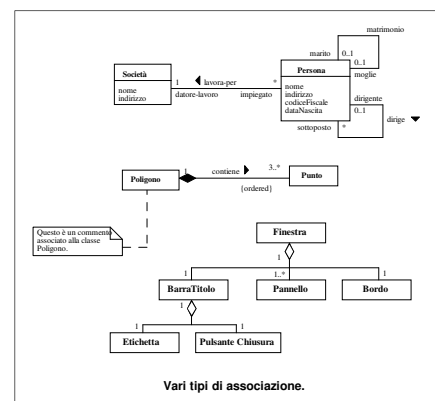
2.75

Analisi Individuazione delle associazioni

- **In UML**
 - La composizione è indicata con un rombo nero dalla parte del contenitore
 - L'aggregazione è indicata con un rombo bianco dalla parte del contenitore
- Ad ogni associazione è possibile aggiungere:
- Un nome (e una direzione di lettura del nome)
 - I ruoli delle classi coinvolte
 - La molteplicità, cioè il numero di oggetti che possono partecipare all'associazione
 - Il contenimento (aggregazione o composizione)
 - ...

Ingegneria del Software L-A

2.76



Ingegneria del Software L-A

2.77

Analisi Individuazione delle associazioni

- Per individuare potenziali aggregazioni o composizioni, considerare
 - Insieme-PartiComponenti
 - Contenitore-Contenuto
 - Collezione-Membri
 - ...
- Se contenitore e contenuto hanno un legame forte e uno stesso ciclo di vita, si tratta di composizione altrimenti, si tratta di aggregazione

Ingegneria del Software L-A

2.78

Analisi Individuazione delle associazioni

- Per individuare potenziali associazioni generiche, per ogni oggetto, porsi le domande
 - A quali altri oggetti è collegato nel dominio del problema?
 - Da chi deve ottenere informazioni per soddisfare alle sue responsabilità?

Ingegneria del Software L-A

2.79

Analisi Individuazione delle associazioni

Una volta determinata la presenza di un'associazione

- Tracciare la linea di connessione, eventualmente col simbolo di aggregazione o composizione dalla parte del contenitore
- Se possibile, dare un nome all'associazione e ai due ruoli, scegliendolo tra i termini del dominio del problema
- Definire il valore, l'intervallo o l'insieme di valori e/o intervalli del
 - limite inferiore
 - la connessione è opzionale? il limite inferiore è 0
 - la connessione è obbligatoria? il limite inferiore è ≥ 1
 - limite superiore
 - la connessione è singola? il limite superiore è 1
 - la connessione è multipla? il limite superiore è > 1

Ingegneria del Software L-A

2.80

Analisi Individuazione delle associazioni

• Molteplicità

Simbolo	Significato	Esempi
n	valore singolo	1
*	da 0 a ∞	*
n..m	intervallo	0..1 1..*
n,m	insieme di valori e/o intervalli	1,3..5,7

Ingegneria del Software L-A

2.81

Analisi Individuazione delle associazioni

- Attenzione alle associazioni molti a molti possono nascondere una classe (**classe di associazione**) del tipo "evento da ricordare"
- Esempio: la connessione proprietà tra Proprietario e Veicolo nasconde una classe **CompraVendita**, che lega Proprietari e Veicoli
- Esempio: la connessione matrimonio tra Persona e Persona nasconde una classe **Matrimonio**, che lega due Persone

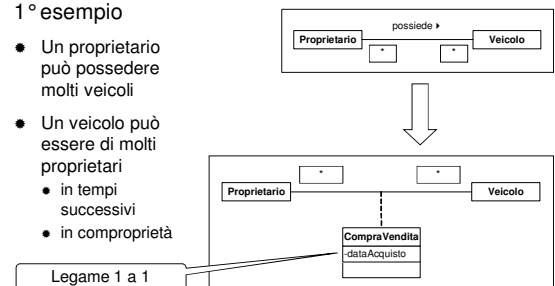
Ingegneria del Software L-A

2.82

Analisi Individuazione delle associazioni

1° esempio

- Un proprietario può possedere molti veicoli
- Un veicolo può essere di molti proprietari
 - in tempi successivi
 - in comproprietà



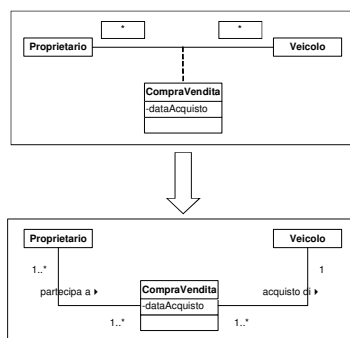
Ingegneria del Software L-A

2.83

Analisi Individuazione delle associazioni

1° esempio

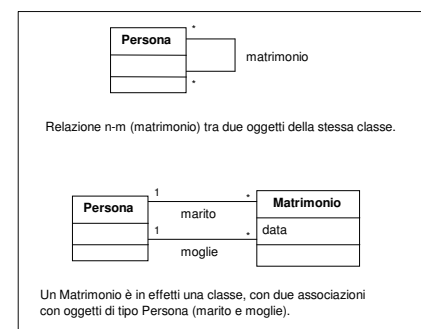
- Un proprietario può partecipare a più compravendite
- Un veicolo può essere contemporaneamente di più proprietari
- Una compravendita è relativa a un solo veicolo



Ingegneria del Software L-A

2.84

Analisi Individuazione delle associazioni



Ingegneria del Software L-A

2.85

Analisi Individuazione delle collaborazioni

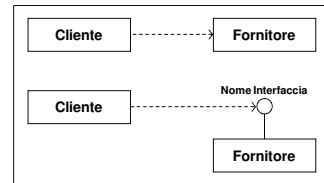
- Una classe A è in relazione USA con una classe B (A USA B) quando A ha bisogno della collaborazione di B per lo svolgimento di alcune funzionalità che A non è in grado di effettuare autonomamente
 - Un'operazione della classe A ha bisogno come argomento di un'istanza della classe B
 - void fun1(B b) { ... usa b ... }
 - Un'operazione della classe A restituisce un valore di tipo B
 - B fun2(...) { B b; ... return b; }
 - Un'operazione della classe A utilizza un'istanza della classe B (ma non esiste un'associazione tra le due classi)
 - void fun3(...) { B b = new B(...); ... usa b ... }

Ingegneria del Software L-A

2.86

Analisi Individuazione delle collaborazioni

- La relazione non è simmetrica
A dipende da B, ma B non dipende da A
- Evitare situazioni in cui una classe, tramite una catena di relazioni USA, alla fine dipende da se stessa



Ingegneria del Software L-A

2.87

Analisi Individuazione degli attributi

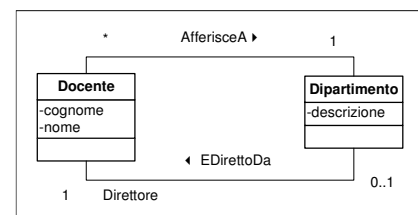
- Ogni attributo modella una proprietà atomica di una classe
 - Un valore singolo
 - Una descrizione
 - Un importo
 - ...
 - Un gruppo di valori strettamente collegati tra loro
 - Un indirizzo
 - Una data
 - ...

Ingegneria del Software L-A

2.88

Analisi Individuazione degli attributi

- Proprietà non atomiche di una classe devono essere modellate come associazioni



Ingegneria del Software L-A

2.89

Analisi Individuazione degli attributi

- Il nome dell'attributo
 - Deve essere un nome familiare
 - all'utente o
 - all'esperto del dominio del problema
 - non allo sviluppatore!
 - Non deve essere il nome di un valore ("qualifica" sì, "ricercatore" no)
 - Deve iniziare con una lettera minuscola
- Esempi
 - cognome
 - dataDiNascita
 - annoDiImmatricolazione

Ingegneria del Software L-A

2.90

Analisi Individuazione degli attributi

- Esprimere tutti i vincoli applicabili all'attributo
 - Tipo (semplice, struttura, enumerativo)
 - cognome: string
 - sesso: {maschio, femmina}
 - Valori ammessi
 - Valore di *default*
 - sesso: {maschio, femmina} = maschio
 - Vincoli di creazione o di accesso
 - Vincoli dovuti ai valori di altri attributi

Ingegneria del Software L-A

2.91

Analisi Individuazione degli attributi

- Esprimere tutti i vincoli applicabili all'attributo
 - Opzionalità
 - votoFinale [0..1]: Votazione
 - Unità di misura, precisione
 - Visibilità (opzionale in fase di analisi)
 - Pubblica +
 - Protetta #
 - Privata -
- **Attenzione:**
gli attributi devono essere sempre privati!

Ingegneria del Software L-A

2.92

Analisi Individuazione degli attributi

- Esprimere tutti i vincoli applicabili all'attributo
 - Appartenenza alla classe (e non all'istanza)
Attributi (e associazioni) possono essere di classe, cioè essere unici nella classe e quindi non fare parte della struttura dati delle istanze
 - -numIstanze: int = 0

Ingegneria del Software L-A

2.93

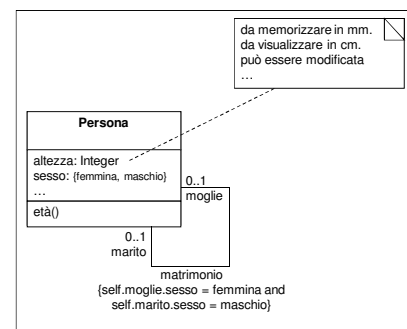
Analisi Individuazione degli attributi

- I vincoli possono essere scritti
 - Utilizzando direttamente UML
 - Tipo
 - Opzionalità
 - Visibilità
 - ...
 - Utilizzando *Object Constraint Language (OCL)* descritto in "The Unified Modeling Language Reference Manual"
 - Come testo in formato libero in un commento UML

Ingegneria del Software L-A

2.94

Analisi Individuazione degli attributi



Ingegneria del Software L-A

2.95

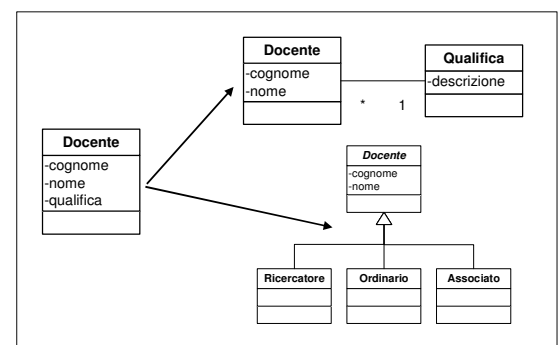
Analisi Individuazione degli attributi

- In un certo istante, ogni oggetto avrà un valore specifico per ogni attributo della classe di appartenenza: informazione di stato
- **Attenzione:** attributi
 - con valore "non applicabile" o
 - con valore opzionale o
 - a valori multipli (enumerazioni) possono nascondere
 - ereditarietà o
 - una nuova classe

Ingegneria del Software L-A

2.96

Analisi Individuazione degli attributi



Ingegneria del Software L-A

2.97

Analisi Individuazione degli attributi

- **Attenzione**
nel caso di attributi con valore "sì o no",
il nome dell'attributo potrebbe essere uno dei valori
di un'enumerazione
ad esempio:
 - tassabile (sì o no) potrebbe diventare
tipoTassazione (tassabile, esente, ridotto, ecc.)

Ingegneria del Software L-A

2.98

Analisi Individuazione degli attributi

- **Attenzione**
nel caso di attributi calcolabili (ad esempio, età),
specificare
 - sempre l'operazione di calcolo
 - mai l'attributo
- Se memorizzare oppure no un attributo calcolabile
è una decisione progettuale, un compromesso tra
 - tempo di calcolo
 - spazio di memoria

Ingegneria del Software L-A

2.99

Analisi Individuazione degli attributi

- **Attenzione**
non definire attributi che indicano il tipo (o categoria)
di un oggetto
- Deve essere sempre possibile ottenere il tipo di un
oggetto mediante un'operazione, ad esempio
 - `nomeClasse()`
 - `GetType()` – in .NET
 - `GetType().ToString()`
per avere il nome della classe

Ingegneria del Software L-A

2.100

Analisi Individuazione degli attributi

- **Applicare l'ereditarietà:**
 - Posizionare attributi e associazioni più generali
più in alto possibile nella gerarchia
 - Posizionare attributi e associazioni specializzati
più in basso

Ingegneria del Software L-A

2.101

Analisi Individuazione delle operazioni

- Per completare la fase di analisi,
è necessario descrivere in modo dettagliato gli
aspetti più volatili del sistema
 - Le operazioni (o servizi) che ogni classe deve
offrire
 - La sequenza temporale di tali operazioni

Ingegneria del Software L-A

2.102

Analisi Individuazione delle operazioni

- **Operazione:** azione, o insieme di azioni, che tutti gli
oggetti della stessa classe devono essere in grado di
compiere per raggiungere un determinato scopo
- Sono in stretto rapporto con le responsabilità di una
classe, escluse le responsabilità di mantenere le
informazioni di stato
- Si eseguono coinvolgendo uno specifico oggetto
della classe (invio di un messaggio all'oggetto)
- Possono anche essere assegnate a una classe e
non alle sue istanze

Ingegneria del Software L-A

2.103

Analisi Individuazione delle operazioni

- Il nome dell'operazione
 - Deve appartenere al vocabolario standard del dominio del problema
 - Potrebbe essere un verbo
 - all'imperativo (scrivi, esegui, ...) o
 - in terza persona (scrive, esegue, ...) in modo consistente in tutto il sistema
 - Dovrebbe iniziare con una lettera maiuscola (convenzione .NET)

Analisi Individuazione delle operazioni

- Notazione UML
 - visibilità nomeOperazione(lista-parametri): tipoRestituito
 - +GetCognome(): string
 - +SetCognome(nuovoValore: string)
 - +GetNumIstanze(): int
 - +Visualizza(dispatch: OutputDevice)

Analisi Individuazione delle operazioni

- Operazioni standard
 - Operazioni che tutti gli oggetti hanno per il semplice fatto di esistere e di avere degli attributi e delle relazioni con altri oggetti
 - Sono implicite e, di norma, non compaiono nel diagramma delle classi
- Altre operazioni
 - Devono essere determinate
 - servizi offerti agli altri oggetti
 - Compaiono nel diagramma delle classi

Analisi Individuazione delle operazioni

Operazioni standard	
Costruttore	Inizializza un nuovo oggetto (.ctor) Invocato automaticamente dalla new
Distruttore	Effettua tutte le azioni necessarie per rilasciare le risorse possedute dall'oggetto Invocato automaticamente da GC o delete
Accessor	Get - restituisce il valore di un attributo atomico o il riferimento a un oggetto (nel caso di associazioni) Set - modifica il valore di un attributo atomico oppure collega/scollega un oggetto a un altro oggetto (nel caso di associazioni)

Analisi Individuazione delle operazioni

Classi contenitori

- Operazioni standard
 - Aggiungi, rimuovi, conta, itera, ...
- Altre operazioni – riguardano l'insieme degli oggetti, non il singolo oggetto
 - Calcoli da effettuare sugli oggetti contenuti
 - CalcolaSulleParti(), Totalizza()
 - Selezioni da fare sugli oggetti contenuti
 - TrovaPartiSpecifiche()
 - Operazioni del tipo
 - InviaComandoATutteLeParti()

Analisi Individuazione delle operazioni

- Distribuire in modo bilanciato le operazioni nel sistema
- Mettere ogni operazione "vicino" ai dati ad essa necessari
- Applicare l'ereditarietà
 - Posizionare le operazioni più generali più in alto possibile nella gerarchia
 - Posizionare le operazioni specializzate più in basso

Analisi Individuazione delle operazioni

- Descrivere tutti i vincoli applicabili all'operazione
 - Parametri formali, loro tipo e significato
 - Pre-condizioni e post-condizioni
 - Invarianti di classe
 - Eccezioni sollevate
 - Eventi che attivano l'operazione
 - Applicabilità dell'operazione
 - ...

Ingegneria del Software L-A

2.110

Analisi Individuazione delle operazioni

- **PRE-CONDIZIONE**
Espressione logica riguardante le aspettative sullo stato del sistema prima che venga eseguita un'operazione
- Ad esempio, per l'operazione **CalcolaRadiceQuadrata (valore)**, la pre-condizione potrebbe essere: "**valore** \geq 0"
- Esplicita in modo chiaro che è responsabilità della procedura chiamante controllare la correttezza degli argomenti passati

Ingegneria del Software L-A

2.111

Analisi Individuazione delle operazioni

- **POST-CONDIZIONE**
Espressione logica riguardante le aspettative sullo stato del sistema dopo l'esecuzione di un'operazione
- Ad esempio, per l'operazione **CalcolaRadiceQuadrata (valore)**, la post-condizione potrebbe essere: "**valore** == **risultato** * **risultato**"

Ingegneria del Software L-A

2.112

Analisi Individuazione delle operazioni

- **INVARIANTE di classe**
Vincolo di classe (espressione logica) che deve essere sempre verificato
 - sia all'inizio
 - sia alla finedi tutte le operazioni pubbliche della classe
- Può non essere verificato solo durante l'esecuzione dell'operazione

Ingegneria del Software L-A

2.113

Analisi Individuazione delle operazioni

- In caso di ridefinizione di un'operazione in una sotto-classe
 - Le pre-condizioni devono essere identiche o meno stringenti
 - Le post-condizioni devono essere identiche o più stringenti
 - Gli invarianti di classe devono essere identici o più stringenti

Ingegneria del Software L-A

2.114

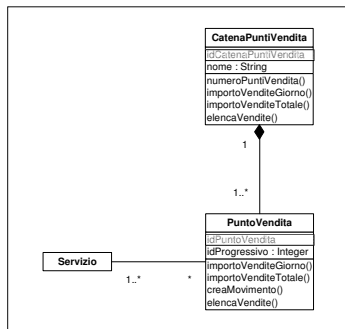
Analisi Individuazione delle operazioni

- **ECCEZIONE**
Si verifica quando un'operazione
 - viene invocata nel rispetto delle sue pre-condizioni
 - ma non è in grado di terminare la propria esecuzione nel rispetto delle post-condizioni

Ingegneria del Software L-A

2.115

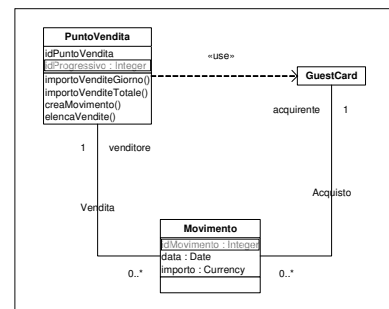
Esempio: Villaggio Turistico Modello dei dati



Ingegneria del Software L-A

2.116

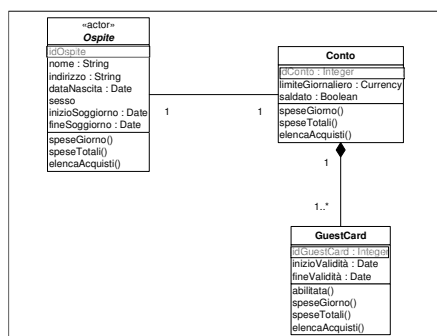
Esempio: Villaggio Turistico Modello dei dati



Ingegneria del Software L-A

2.117

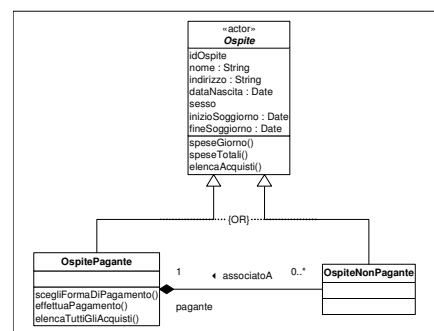
Esempio: Villaggio Turistico Modello dei dati



Ingegneria del Software L-A

2.118

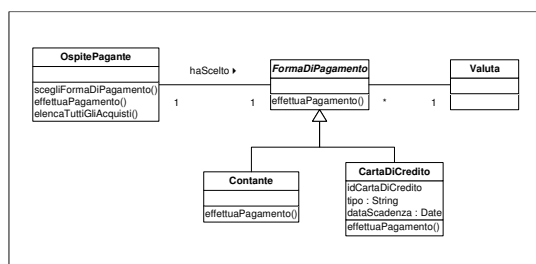
Esempio: Villaggio Turistico Modello dei dati



Ingegneria del Software L-A

2.119

Esempio: Villaggio Turistico Modello dei dati



Ingegneria del Software L-A

2.120

Analisi Modello dinamico

- Descrive il sistema durante il suo funzionamento
- Consiste nella definizione di scenari di funzionamento del sistema, in risposta a eventi esterni o in fasi particolarmente significative
- Diagrammi dei casi d'uso - evidenziano
 - le risposte a eventi esterni
- Diagrammi di stato - evidenziano
 - i possibili stati che un oggetto può assumere
 - gli eventi che attivano transizioni da uno stato all'altro

Ingegneria del Software L-A

2.121

Analisi Modello dinamico

- Diagrammi di interazione - descrivono lo scambio di messaggi tra oggetti
- Diagrammi di sequenza - evidenziano
 - l'ordine in cui i messaggi vengono scambiati tra gli oggetti
- Diagrammi di collaborazione - evidenziano
 - le connessioni tra gli oggetti e i messaggi scambiati
- Entro certi limiti, un diagramma di sequenza può essere convertito nel suo corrispondente diagramma di collaborazione e viceversa

Ingegneria del Software L-A

2.122

Analisi Modello dinamico

- Non conviene spingere la definizione del modello dinamico sino ai minimi dettagli
- Utilizzare i diagrammi dinamici solo per descrivere il funzionamento del sistema nei casi più critici
- La definizione dettagliata di tutte le operazioni del sistema può essere fatta realizzando un prototipo funzionante

Ingegneria del Software L-A

2.123

Analisi Diagrammi di interazione

- Obiettivo
Evidenziare le interazioni (e quindi lo scambio di messaggi) tra gli oggetti del sistema reale
- Modellare solo gli scenari veramente cruciali al funzionamento del sistema
- Partire dai casi d'uso, cioè considerare ciò che deve fare il sistema in risposta a
 - richieste dei suoi utenti o
 - a eventi esterni

Ingegneria del Software L-A

2.124

Analisi Diagrammi di interazione

- L'invio di un messaggio (o richiesta di un servizio)
 - da un oggetto mittente (o cliente)
 - a un oggetto destinatario (o fornitore)deve corrispondere a un'operazione della classe del destinatario o di una sua superclasse
- Pertanto, ogni messaggio comprenderà:
 - il nome dell'operazione invocata
 - gli eventuali parametri attuali
 - un eventuale valore restituito

Ingegneria del Software L-A

2.125

Analisi Definizione di una sequenza di messaggi

- Considerare l'attore, l'evento esterno o il generico oggetto che scatena la sequenza di operazioni
- Determinare il primo servizio richiesto e il fornitore di tale servizio – specificare il primo messaggio
- Immedesimarsi nell'oggetto (o nella classe) che riceve il messaggio e chiedersi
 - Sono in grado di eseguire in modo autonomo l'operazione richiesta?in caso negativo, chiedersi
 - A quali altri oggetti devo inviare messaggi per ottenere i dati e/o i (sotto) servizi mancanti?

Ingegneria del Software L-A

2.126

Analisi Definizione di una sequenza di messaggi

- Iterare il procedimento, sino all'identificazione di tutta la sequenza dei messaggi
- Identificare modelli comportamentali standard di interazione tra oggetti (*pattern*)
- Ricordarsi che un oggetto (cliente) può inviare un messaggio (chiedere un servizio) ad un altro oggetto (fornitore)
 - Solo durante l'esecuzione di un proprio metodo
 - Solo se conosce il fornitore

Ingegneria del Software L-A

2.127

Analisi Definizione di una sequenza di messaggi

- Il fornitore è un oggetto globalmente accessibile (una classe o un oggetto globale)
- Il fornitore è contenuto nel cliente (per valore o per riferimento)
- Il fornitore è stato passato come parametro attuale al metodo del cliente
- Il fornitore è il risultato dell'invio di un precedente messaggio dentro il metodo del cliente due possibilità:
 - Il fornitore esisteva già
 - Il fornitore viene creato, utilizzato (e distrutto)

Ingegneria del Software L-A

2.128

Analisi Definizione di una sequenza di messaggi

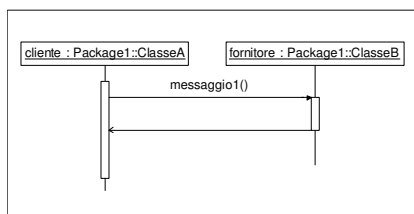
- Se il traffico di messaggi necessari per accedere a un fornitore è troppo alto, *forse* è opportuno aggiungere una connessione permanente tra cliente e fornitore
- Se ci sono oggetti che dominano lo scenario, facendo quasi tutto, *forse* esiste un problema di distribuzione delle responsabilità

Ingegneria del Software L-A

2.129

Analisi Diagrammi di sequenza

- Obiettivo: porre in evidenza la sequenza (e la durata temporale) dei messaggi scambiati tra gli oggetti

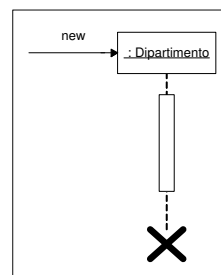


Ingegneria del Software L-A

2.130

Analisi Diagrammi di sequenza

- Gli oggetti sono rettangoli con dentro i nomi sottolineati dell'oggetto e/o della relativa classe



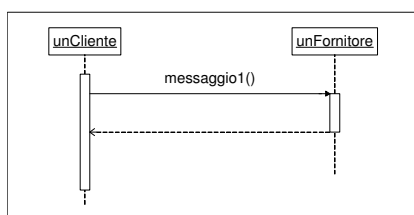
- Le linee verticali tratteggiate (*lifeline*) indicano il tempo di vita degli oggetti
- Il flusso temporale scorre dall'alto in basso
- Gli ispessimenti delle linee verticali denotano un'elaborazione in corso
- È possibile indicare esplicitamente quando un oggetto viene creato e quando viene distrutto

Ingegneria del Software L-A

2.131

Analisi Diagrammi di sequenza

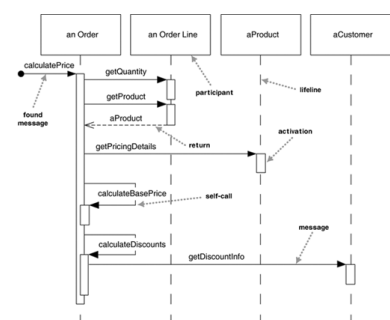
- Tra gli oggetti vi possono essere invii di messaggi, denotati da frecce, e ritorni (opzionali)



Ingegneria del Software L-A

2.132

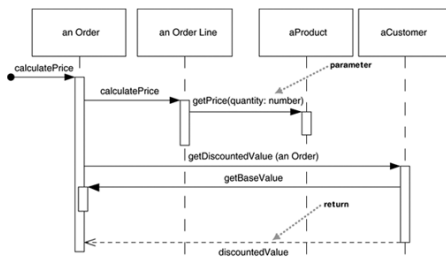
Analisi Diagrammi di sequenza



Ingegneria del Software L-A

2.133

Analisi Diagrammi di sequenza

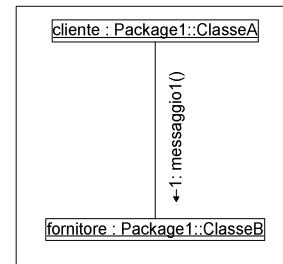


Ingegneria del Software L-A

2.134

Analisi Diagrammi di collaborazione

- Obiettivo: porre in evidenza come gli oggetti sono connessi tra di loro e come collaborano



Ingegneria del Software L-A

2.135

Analisi Diagrammi di collaborazione

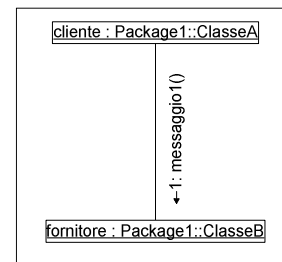
- Una connessione tra oggetti:
 - Modella le dipendenze di un oggetto per quanto riguarda le elaborazioni, indicando la necessità di richiedere servizi per soddisfare le sue responsabilità
 - Mostra graficamente l'accesso di un oggetto (il cliente) ad un altro oggetto (il fornitore del servizio) e la richiesta di un servizio
 - È in stretto rapporto con le collaborazioni di un oggetto

Ingegneria del Software L-A

2.136

Analisi Diagrammi di collaborazione

- Gli oggetti sono rettangoli con dentro i nomi sottolineati dell'oggetto e/o della relativa classe (come nei diagrammi di sequenza)
- Le connessioni tra oggetti sono mostrate come linee continue tracciate tra gli oggetti

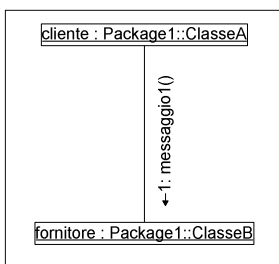


Ingegneria del Software L-A

2.137

Analisi Diagrammi di collaborazione

L'invio di un messaggio è indicato da:



- una freccia posta parallelamente alla connessione
- un numero progressivo che fornisce la posizione nella sequenza di invio dei messaggi
- una condizione (opzionale) tra parentesi quadre
- il nome del messaggio ed eventualmente i suoi argomenti

Ingegneria del Software L-A

2.138

Analisi Diagrammi di stato

- Lo stato di un oggetto è dato dal valore dei suoi attributi e delle sue associazioni
- In molti domini applicativi, esistono oggetti che, a seconda del proprio stato, rispondono in maniera diversa ai messaggi ricevuti
 - Dispositivi (spento, in attesa, operativo, guasto, ecc.)
 - Transazioni complesse (in definizione, in esecuzione, completata, fallita, ecc.)
 - ...
- In questi casi, è opportuno disegnare un diagramma di stato per l'oggetto, mostrando i possibili stati e gli eventi che attivano transizioni da uno stato all'altro

Ingegneria del Software L-A

2.139

Analisi Diagrammi di stato

• STATO

Situazione, che si verifica durante la vita di un oggetto, nella quale l'oggetto

- Soddisfa certe condizioni
- Esegue certe attività
- Aspetta il verificarsi di certi eventi

Ingegneria del Software L-A

2.140

Analisi Diagrammi di stato

• EVENTO

Fatto che avviene in un intervallo di tempo e in una porzione di spazio così ristretti da poterlo considerare caratterizzato da un istante e/o da un punto

• Può essere:

- Esterno – generato nell'ambito del mondo reale (mouse, tastiera, arrivo ordine, passaggio qualifica, ...)
- Interno – generato da un oggetto del sistema

Ingegneria del Software L-A

2.141

Analisi Diagrammi di stato

• Call Event

Occurs when an element receives a call for an operation

• Signal Event

Occurs when an element receives an explicit signal from another element

• Change Event

Occurs when a designated condition (usually described as a Boolean operation) becomes true

• Time Event

Occurs after a designated period of time or at a specific time or date

Ingegneria del Software L-A

2.142

Analisi Diagrammi di stato

• TRANSIZIONE

Relazione tra due stati
S1 (stato di partenza) e S2 (stato di arrivo)
indicante che

- un oggetto inizialmente nello stato S1
- in seguito al verificarsi di un particolare evento
- e se sono soddisfatte certe condizioni,
- effettuerà certe azioni ed
- entrerà nello stato S2

Ingegneria del Software L-A

2.143

Analisi Diagrammi di stato

• AZIONE

Computazione atomica (non interrompibile)

- associata a una transizione

• ATTIVITÀ

Computazione non atomica (interrompibile)

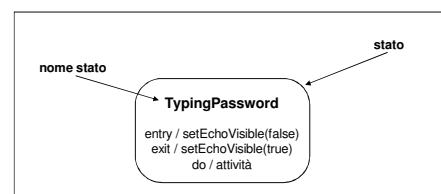
- associata a uno stato
- insieme di azioni

Ingegneria del Software L-A

2.144

Analisi Diagrammi di stato

• Notazione UML

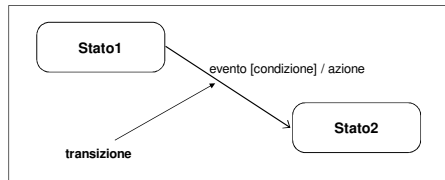


Ingegneria del Software L-A

2.145

Analisi Diagrammi di stato

• Notazione UML



Ingegneria del Software L-A

2.146

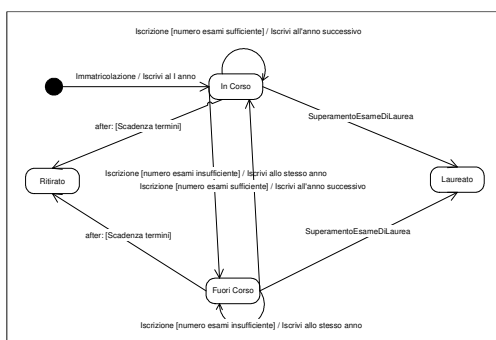
Analisi Identificazione degli stati

- Esaminare i possibili valori degli attributi di un oggetto
- Determinare se le responsabilità del sistema prevedono
 - comportamenti diversi
 - per valori diversi degli attributi
- Descrivere mediante un diagramma di stato
 - tutti i possibili stati che un oggetto può assumere durante la sua vita e
 - come cambia lo stato dell'oggetto in conseguenza degli eventi

Ingegneria del Software L-A

2.147

Analisi Diagrammi di stato



Ingegneria del Software L-A

2.148