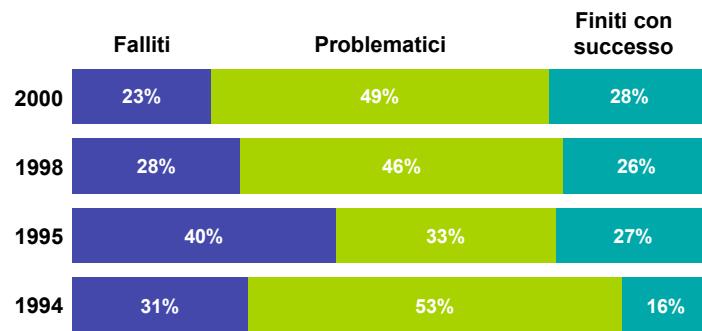


Agenda

Introduzione a a RUP (e UML)

prof. Paolo Ciancarini
Università di Bologna

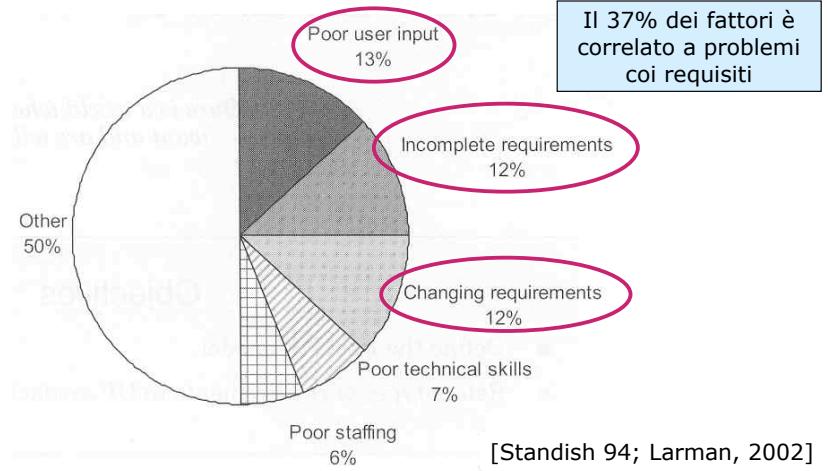
Tassi di fallimento dei progetti sw



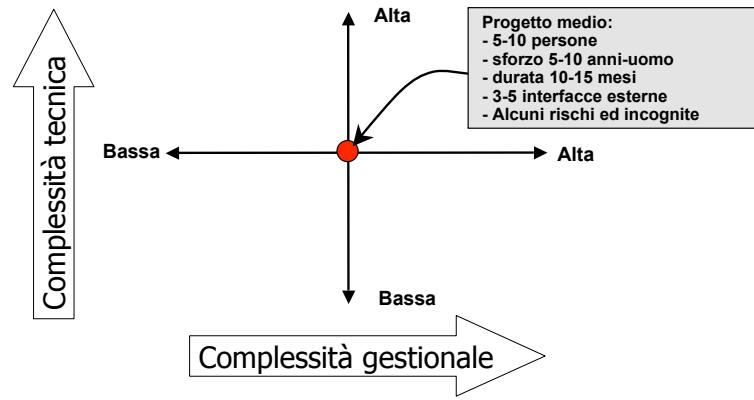
Risultato di circa 30.000 progetti sw in aziende USA piccole e grandi
Fonte: *Extreme Chaos, The Standish Group International, Inc., 2000*

- Le attività legate ai progetti software
- Il RUP
- Esempi di documentazione UML
- Strumenti di modellazione
- I workflow del RUP

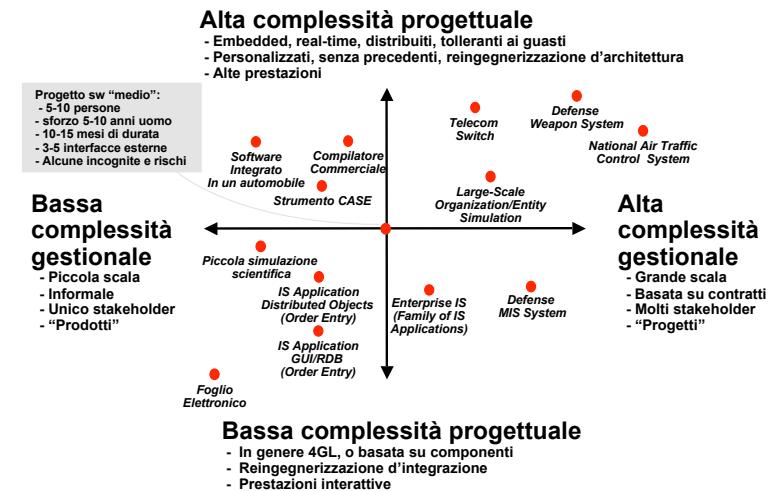
Fattori che ostacolano i progetti sw



La complessità dei progetti sw



Sistemi “software intensive”



Modellare il processo di sviluppo

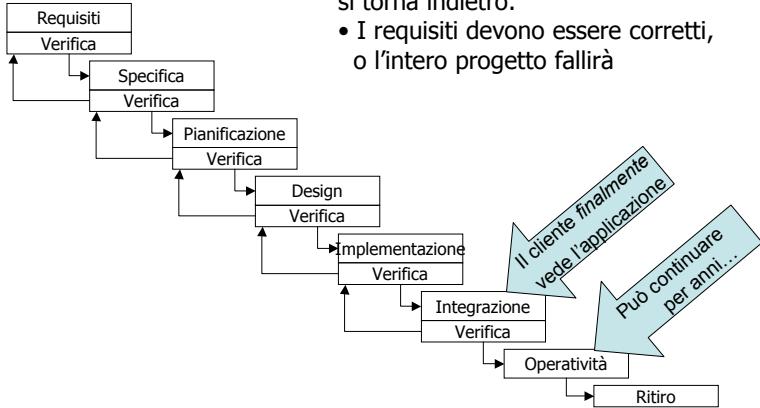
- Siccome ogni prodotto sw è *invisibile e intangibile*, chi gestisce i progetti di costruzione di tali prodotti ha bisogno di controllarne il processo di sviluppo in modo speciale
- Il controllo si basa sulla definizione esplicita delle attività e dei documenti da produrre: questo si chiama “**modellare il processo**”
- I documenti (“artefatti”) prodotti dal progetto permettono di controllare l’*avanzamento* del processo e di valutarne la qualità

I documenti di processo

- I diversi *modelli* di processo, e le loro *istanze*, differiscono per i documenti prodotti
- Il management adora la documentazione, i progettisti di solito la snobbano: sono sbagliati entrambi i punti di vista
- Bisogna produrre i documenti necessari al progetto!

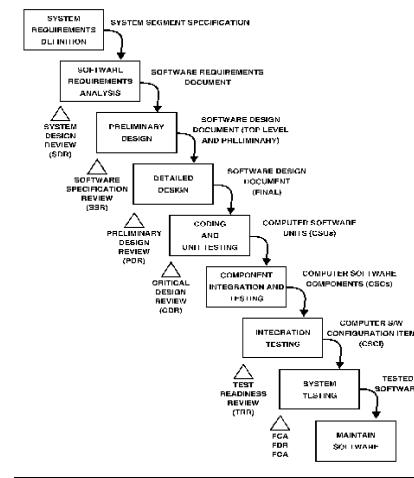
Il modello di processo a “Cascata”

- Quando una fase è finita non si torna indietro.
- I requisiti devono essere corretti, o l'intero progetto fallirà



Modello a cascata (Royce)

- Molto dettagliato
- Molto rigido
- Orientato alla documentazione
- Orientato agli standard
- Adatto per organizzazioni gerarchizzate
- Rimanda il rischio alla fine del processo

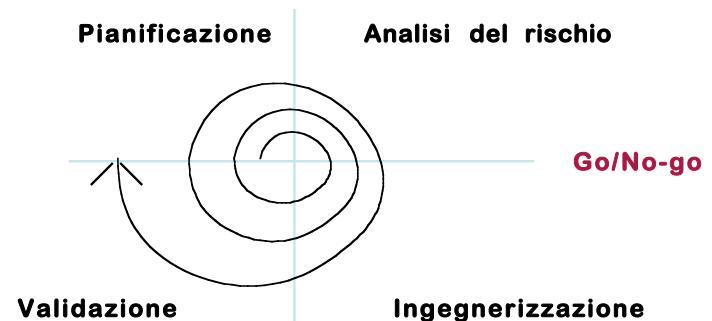


Il modello di processo a “Spirale”



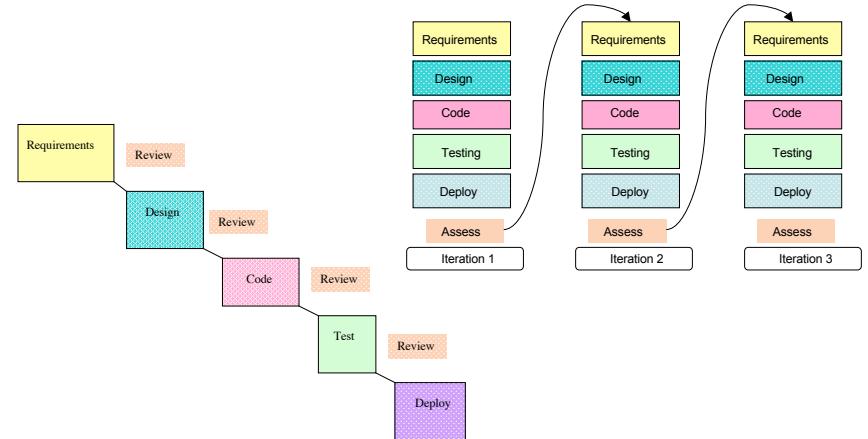
- Non lineare, ma iterativo
- Flessibile
- Valuta continuamente il rischio
- Involge il “cliente” sin dalle prime fasi
- Adatto se requisiti instabili o poco noti

Modello a spirale (Boehm)

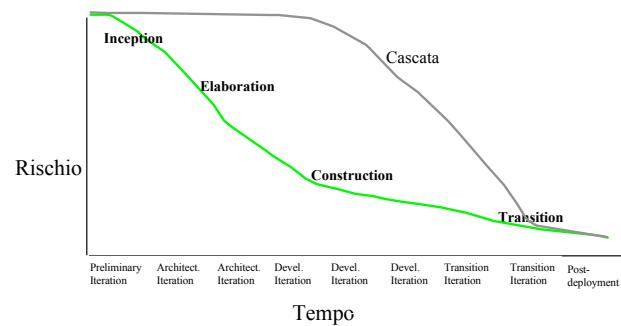


I processi iterativi diminuiscono i rischi

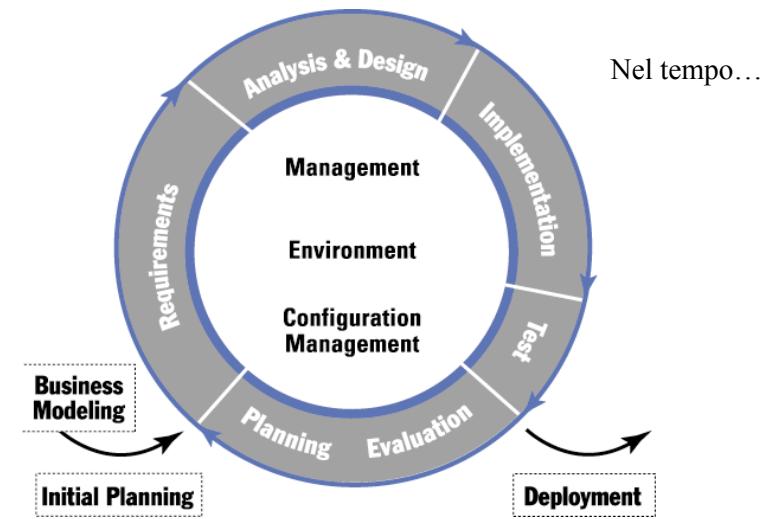
Cascata vs iterativi



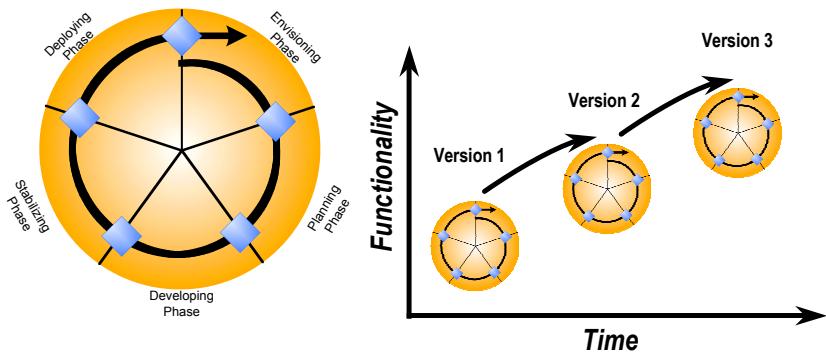
Cascata vs Iterativi



Il RUP è un processo iterativo



Anche MSF è un processo iterativo



Che cos'è la progettazione OO?

Orientato agli oggetti:

- Decomposizione di un sistema mediante astrazioni orientate al dominio
- Diversa dalla decomposizione funzionale/procedurale

OO Design, Analysis e Programming:

OOA: Esaminare il dominio e decomporre il problema...*analysis*

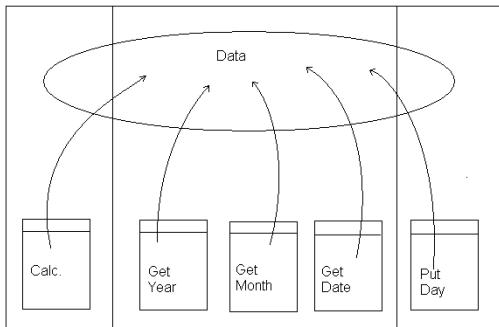
OOD: Costruire un modello del sistema...*design*

OOP: Realizzare il modello...*programming*



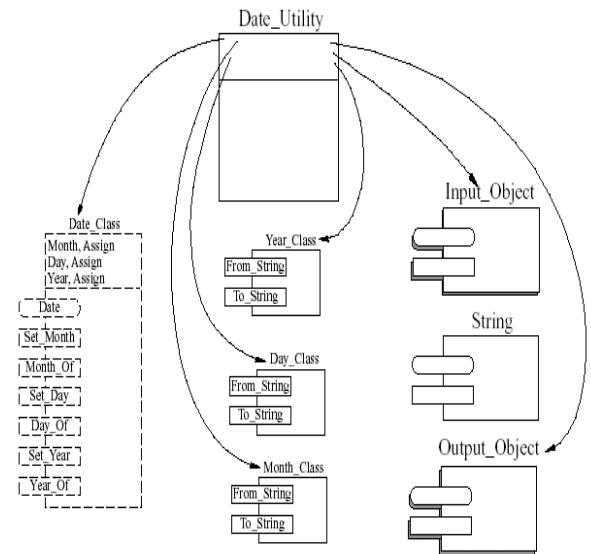
Approccio funzionale (non OO)

- Strutture dati globali
- I componenti sono blocchi funzionali
- Astrazione e encapsulamento assenti o poveri
- Non modella il dominio del problema
- Sequentiale (thread singolo)

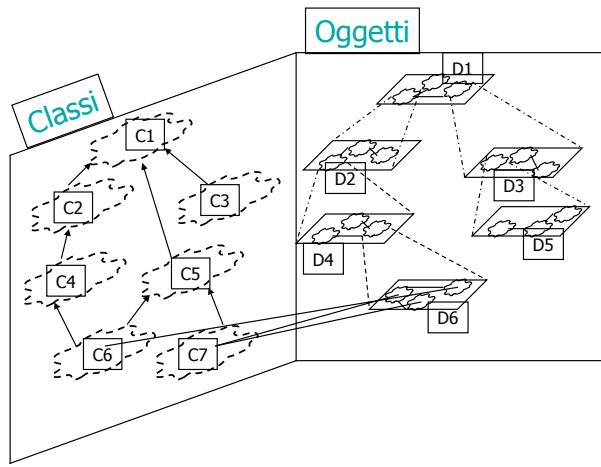


- I componenti sono classi e oggetti ricavati dal dominio
- I meccanismi principali di design sono l'astrazione e l'incapsulamento
- Il sistema finale rispecchia il dominio del problema
- Le strutture dati sono “logicamente distribuite”
- Concorrenza (thread multipli)

Approccio OO



Forma canonica di un sistema OO



Il paradigma OO secondo i “tre amigos”

- Booch: Definizione e classificazione di nozioni base
- Rumbaugh: Modelli e notazioni diagrammatiche
- Jacobson: modello di processo Objectory
- Verso il 1994-95 definiscono per Rational le basi di UML e UP
- Nel 2001 Rational viene acquisita da IBM



The three amigos:

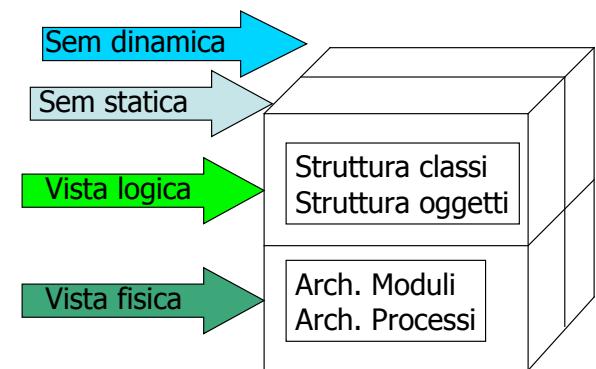
Grady Booch, Jim Rumbaugh, Ivar Jacobson

Principali elementi del paradigma OO secondo Booch

- **Astrazione** Il progettista crea le classi e gli oggetti derivandole dal dominio
- **Incapsulamento** Le strutture dati entro un oggetto non sono accessibili dall'esterno dell'oggetto stesso
- **Modularità** Decomposizione in moduli ad accoppiamento lasco
- **Gerarchia** ordinamento di astrazioni mediante ereditarietà

Il “cubo” di Booch

- Vista **logica** e vista **fisica**
- Semantica **statica** e semantica **dinamica**



Unified Sw Development Process

Jacobson propose all'inizio degli anni 90 un metodo per progettazione ad oggetti chiamato Objectory

Verso il 1996 Objectory venne scelto da Rational come processo di UML col nome USDP; Rational mise il copyright su questo processo col nome di RUP.

Nel 2001 IBM ha comprato Rational e continua a supportare RUP™. A volte viene chiamato UP

Dunque

$$\text{Objectory} = \text{USDP} = \text{RUP}^{\text{TM}} = \text{UP}$$

Le fasi principali di RUP

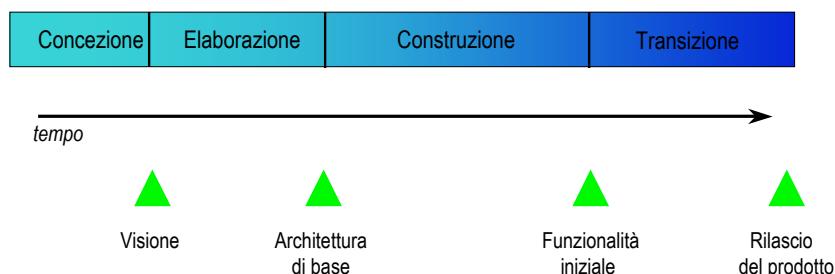
Il ciclo di vita di RUP è suddiviso in una serie di *iterazioni*

Ogni iterazione è composta da una serie di *fasi*

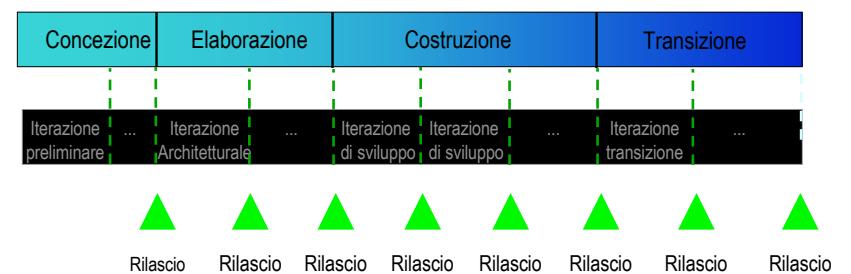
1. Concezione
2. Elaborazione
3. Costruzione
4. Transizione



RUP: Consegnate Principali

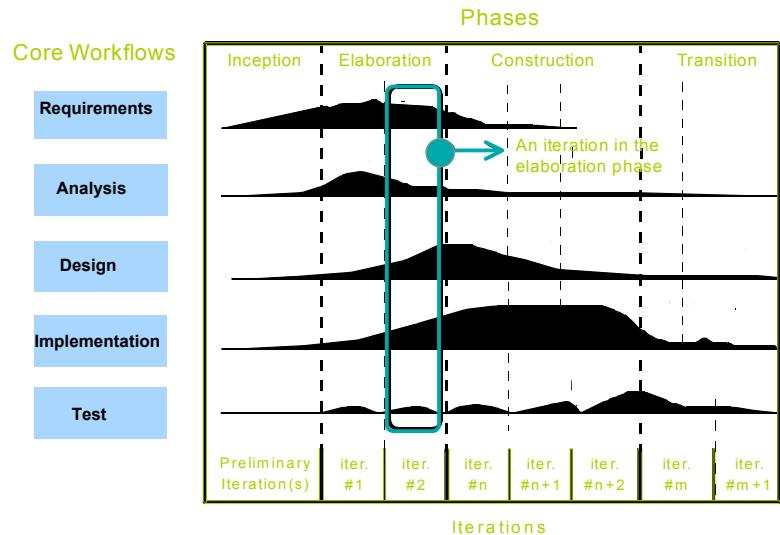


Fasi e iterazioni

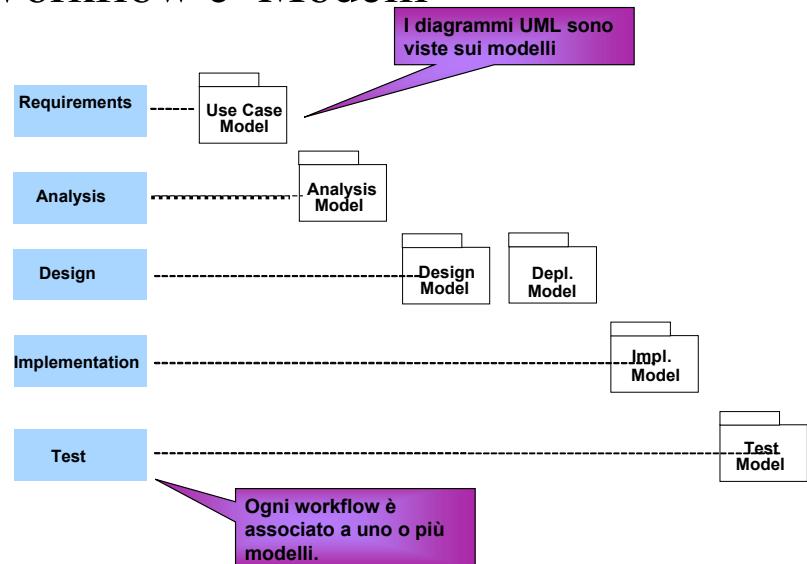


Un'**iterazione** è una sequenza di *attività* con un *piano* prestabilito e dei *criteri* di valutazione, che termina con un *rilascio* eseguibile

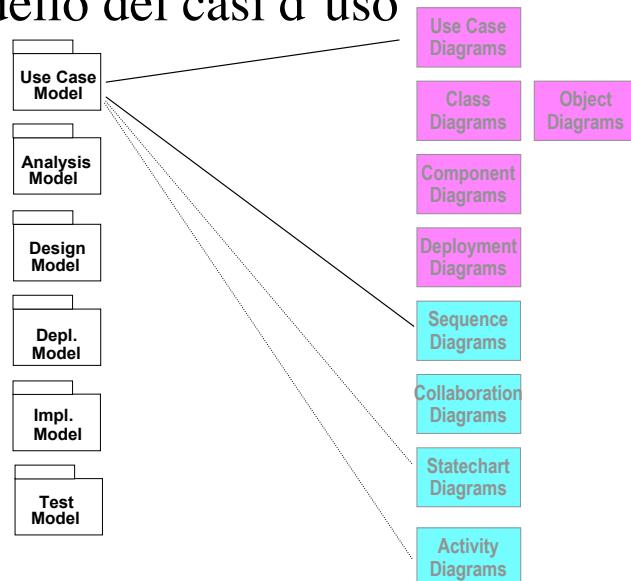
Iterazioni e workflow



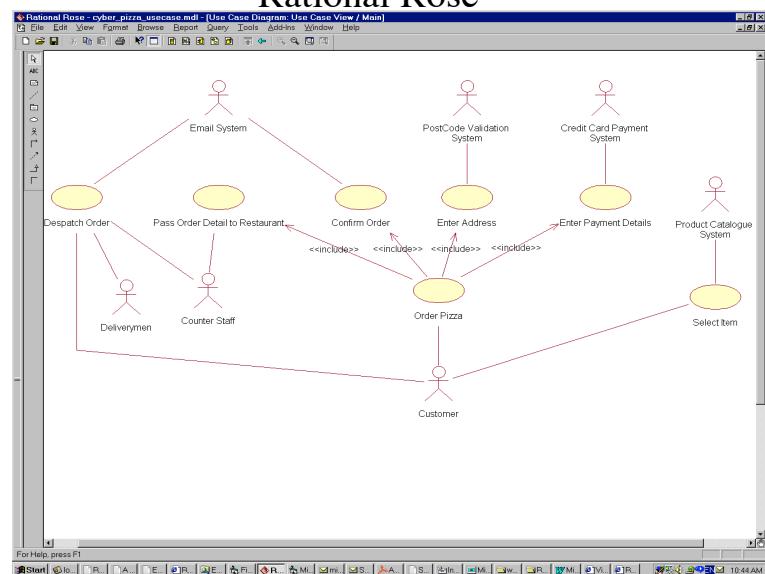
Workflow e Modelli



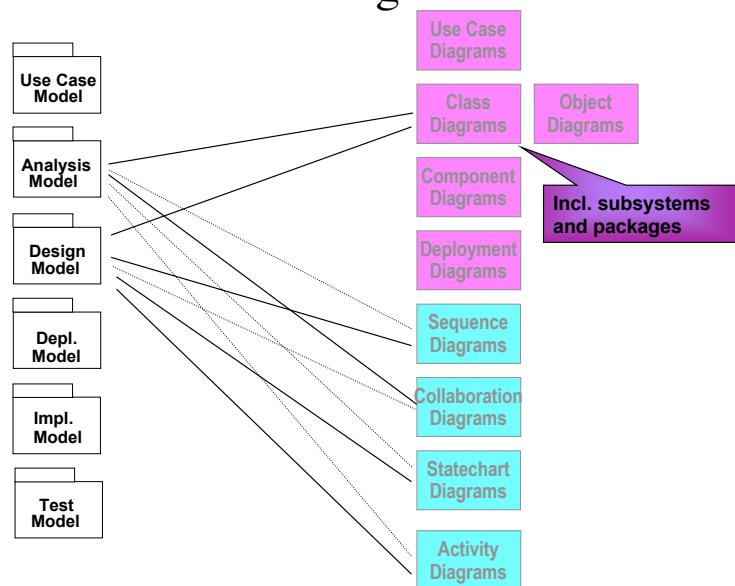
Modello dei casi d'uso



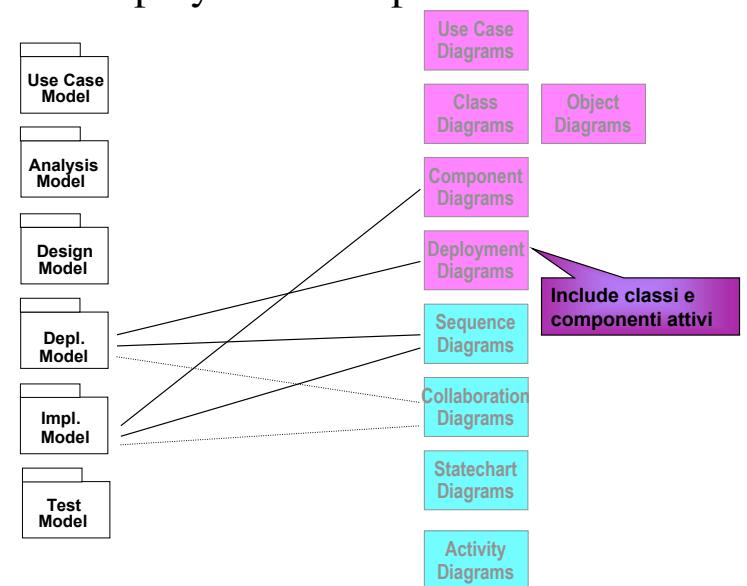
Esempio: un modello dei casi d'uso in Rational Rose



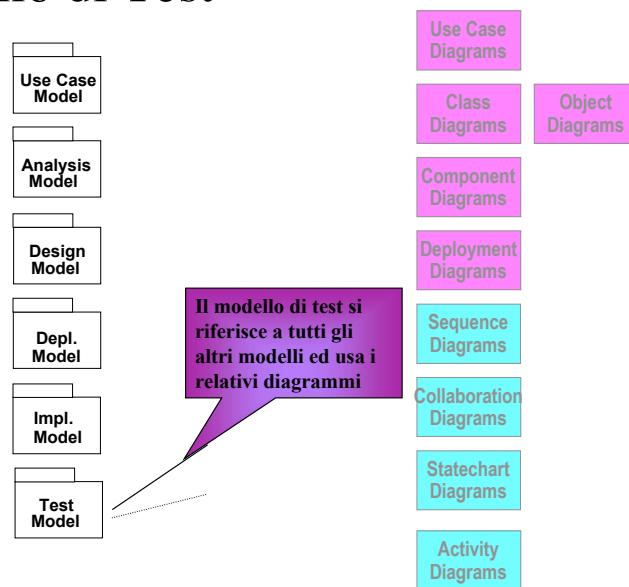
Modello di Analisi e Design



Modello di Deployment e Implementazione



Modello di Test



Concezione (Inception)

- **Inception**--The good idea: specifying the end-product vision and its business case, defining the scope of the project.
- La concezione è la fase in cui si forma l'idea di creare o adattare un sistema software

RUP: concezione (*inception*)

- Scopo
 - Stabilire il business case per il nuovo sistema o per l'aggiornamento di un sistema esistente.
- Prodotti
 - I **requisiti** chiave per il progetto
 - Una valutazione iniziale del **rischio**
- Prodotti opzionali:
 - Un **prototipo** concettuale
 - Un primo **modello del dominio** (completo al 10, 20%)

Elaborazione (Elaboration)

- **Elaboration**--Planning the necessary activities and required resources; specifying the features and designing the architecture
- L'Elaborazione è la fase in cui si inizia la progettazione

RUP: elaborazione

- Scopo
 - **Analizzare** il dominio del problema
 - Stabilire un'adeguata **base architetturale**
 - Evidenziare gli elementi ad alto **rischio** del progetto
 - Sviluppare un **piano** per la realizzazione del progetto
- Prodotti
 - Un modello del sistema con il contesto, gli scenari ed il modello del dominio
 - L'architettura dell'eseguibile
 - Un piano rivisto dei rischi
 - Un piano di sviluppo e di testing
 - Una descrizione della release
 - Una prima versione dello User Manual

Costruzione (Construction)

- **Construction**--Building the product and evolving the vision, the architecture, and the plans until the product--the completed vision--is ready for transfer to its users' community
- La costruzione è la fase in cui si realizza una versione del sistema

RUP: costruzione

- Scopo
 - Sviluppare **incrementalmente** un prodotto software completo pronto per essere inserito nella comunità degli utenti
- Prodotti
 - Una serie di rilasci degli eseguibili
 - Dei prototipi comportamentali
 - I risultati dell'assicurazione di qualità
 - La documentazione utente e del sistema
 - Il piano di rilascio
 - Criterio di valutazione per l'iterazione successiva

Transizione (Transition)

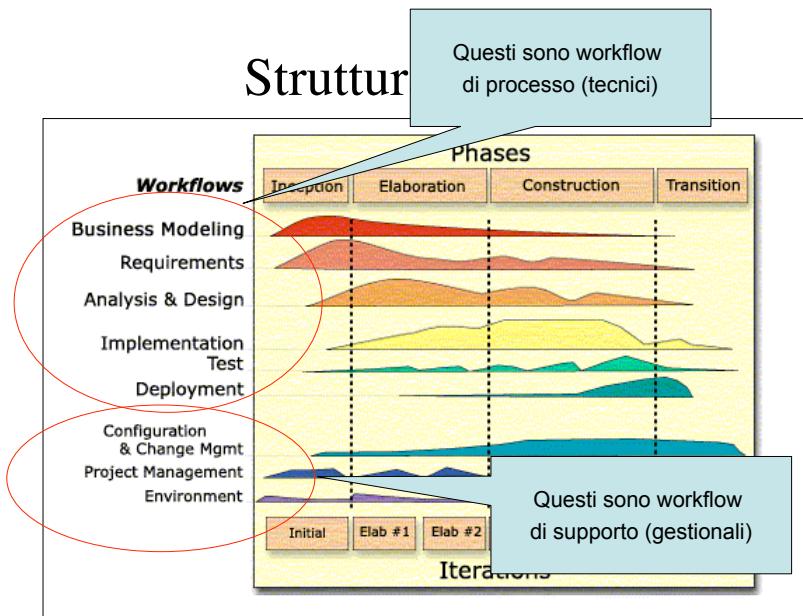
- **Transition**--Making the transition from the product to its user's community, which includes: manufacturing, delivering, training, supporting, maintaining the product until the users are satisfied
- La transizione è la fase in cui il sistema diventa operativo e validato dagli utenti

RUP: transizione

- Scopo
 - Inserire il prodotto software nella comunità degli utenti
- Prodotti
 - Una serie di rilasci degli eseguibili
 - I risultati dell'assicurazione di qualità
 - Documentazione utente e di sistema aggiornata
 - Analisi delle prestazioni del sistema dopo il rilascio

Il RUP ha due punti di vista

- Il processo RUP integra due diverse prospettive:
 - Una **prospettiva tecnica**, che tratta gli aspetti qualitativi, ingegneristici e di metodo di progettazione
 - Una **prospettiva gestionale**, che tratta gli aspetti finanziari, strategici, commerciali e umani
- Le due prospettive sono rispettivamente articolate su sei e tre “**core workflow**”



Dimensioni del RUP

- La dimensione **orizzontale** (temporale) rappresenta l'aspetto *dinamico* del processo
 - Cicli, fasi, iterazioni e milestone
- Un prodotto sw viene progettato e costruito durante una serie di iterazioni incrementali
- La dimensione **verticale** (strutturale) rappresenta la struttura *statica* del processo, descritta mediante i suoi componenti: *attività*, *workflow*, *artefatti* e *ruoli* (worker)

Workflow di processo

- **Business modeling:** attività che modellano l'ambito di risoluzione del problema, ovvero l'ambiente esterno al sistema da produrre
- **Requirements:** attività che modellano i requisiti del sistema da produrre
- **Analysis and design:** attività di decomposizione del problema e progetto architettonico del sistema
- **Implementation:** attività di progetto dettagliato e codifica del sistema
- **Test:** controllo di qualità, sia a livello di moduli che di integrazione
- **Deployment:** attività di consegna e messa in opera

Workflow di supporto

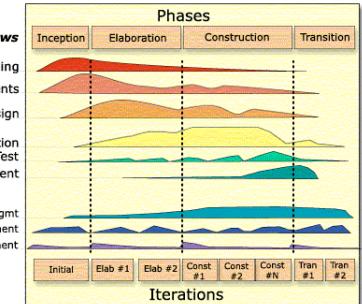
- **Configuration and change management:** attività di manutenzione durante il progetto, ovvero richieste di cambiamento e gestione della configurazione
- **Project management:** attività di pianificazione e governo del progetto
- **Environment:** attività che supportano il team di progetto, riguardo ai processi e strumenti utilizzati

Qualità del RUP

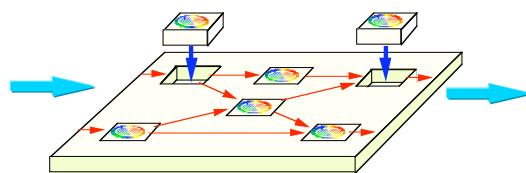
- Iterativo (ciclico)
- Incrementale
- Object-oriented
- Flessibile ed Estendibile
- Gestito e controllato
- Altamente automatizzato

RUP è un modello iterativo

- Una iterazione è un ciclo di sviluppo che porta al rilascio di una parte del prodotto finale
- Ogni iterazione tocca tutti gli aspetti dello sviluppo sw
- Ogni rilascio iterativo è una parte pienamente documentata del sistema finale



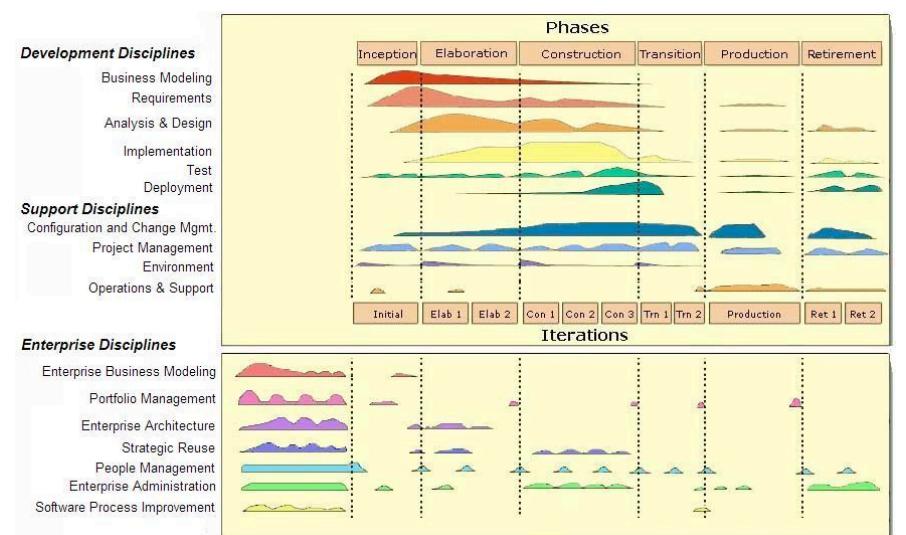
RUP è un Framework di Processo



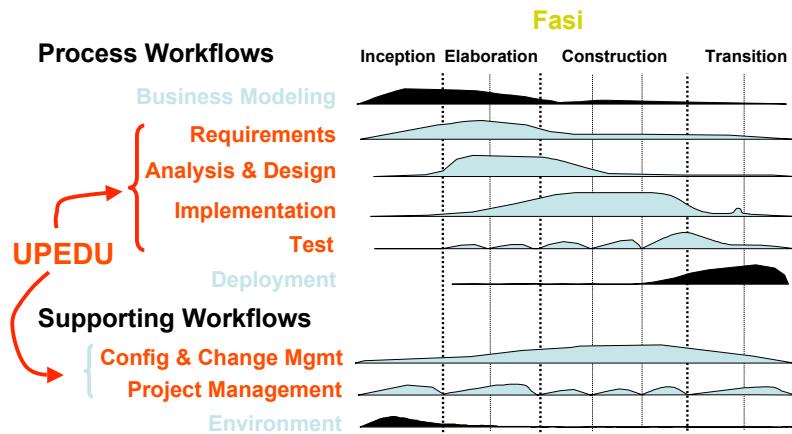
Siccome non esiste un processo universalmente valido, RUP è stato progettato per essere flessibile ed estendibile: permette

- Ridefinire la struttura del ciclo di vita
- Selezionare gli artefatti da produrre
- Definire attività e ruoli dei worker
- Modellare nuovi aspetti di processo

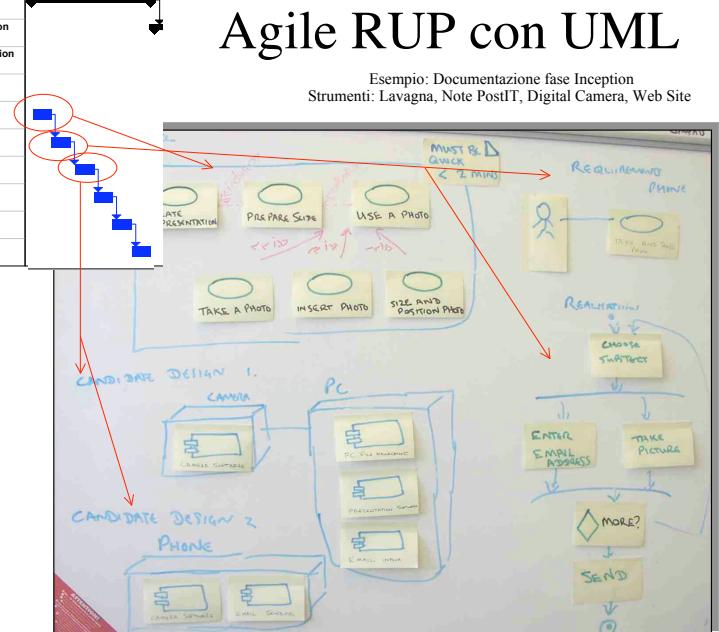
RUP si può estendere: es. EUP



RUP si può restringere: es UPEDU



ID	Task Name	November	December
1	1st Iteration: Inception		
2	2nd Iteration: Elaboration		
3	3rd Iteration: Construction		
4	4th Iteration: Transition		
5	Business Modelling		
6	Requirements		
7	Analysis and Design		
8	Build		
9	Test		
10	Deployment		



Gli strumenti di supporto

- Esistono innumerevoli strumenti di supporto alla progettazione con UML e RUP
- I più famosi:
 - Rational Rose
 - Eclipse (open source da IBM)
 - Microsoft Visual Studio
 - Together
 - Argo (open source) e Poseidon (versione commerciale)

Rational Rose: Artefatti

Artifact: Use-Case Model

Rational Rose: Linee guida

Guidelines: Use-Case Model

The use-case model is a model that describes a system's requirements in terms of use cases.

Topics

- Explanation
- How the use-case model evolves
- Avoiding functional decomposition
- Non-functional requirements
- The what versus how dilemma
- Concrete and abstract use cases
- Structuring the use-case model
- Are use cases always related to actors?
- The survey description

Explanation

A use-case model is a model of the system's intended functions and its surroundings, and serves as a contract between the customer and the developers. Use cases serve as a unifying thread throughout system development. The same use-case model is the result of the Requirements workflow and used as input to Analysis & Design and Test workflows.

The diagram below shows a part of a use-case model for the Recycling-Machine System.

Rational Rose: Attività

Activity: Structure the Use-Case Model

Purpose

- To extract behavior in use cases that need to be considered as abstract use cases. Examples of such behavior are common behavior, optional behavior, exceptional behavior, and behavior that is to be developed in later iterations.
- To find new abstract actors that define roles that are shared by several actors.

Steps

- Establish Include-Relationships Between Use Cases
- Establish Extend-Relationships Between Use Cases
- Establish Generalizations Between Use Cases
- Establish Generalizations Between Actors
- Evaluate Your Results

Input Artifacts:

- Use-Case Modeling Guidelines
- Glossary
- Use-Case Model
- Use Cases
- Supplementary Specifications
- Use-Case Packages (optional)

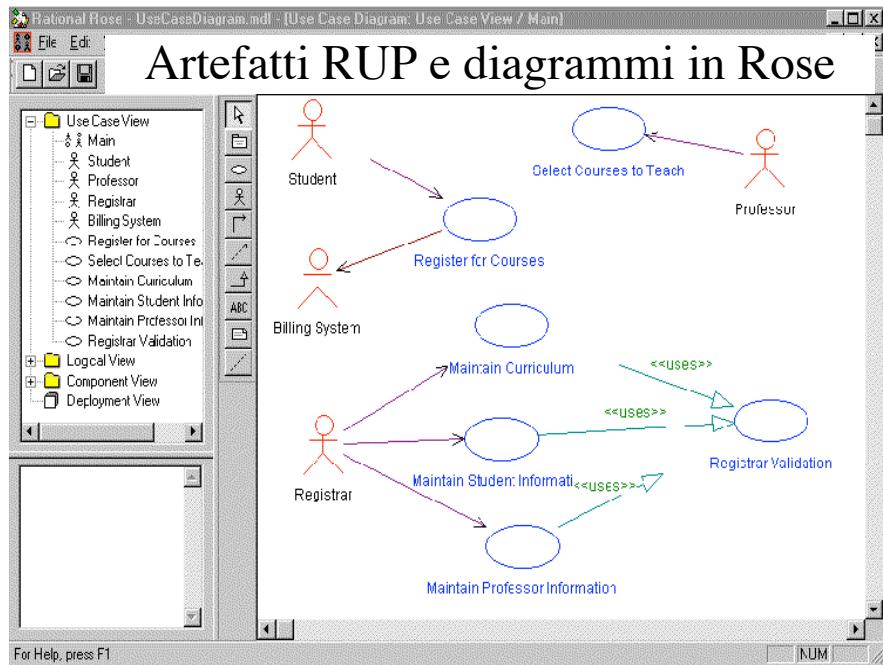
Resulting Artifacts:

- Use Case, existing ones are refined, and new abstract ones are found.
- Use-Case Model, refined.
- Use-Case Package (optional), refined.

Role: System Analyst

Tool Mentors:

- Structuring the Use-Case Model using Rational Rose



Il linguaggio di modellazione

- Un linguaggio di modellazione permette di specificare, *visualizzare* e documentare un processo di sviluppo OO
- I **modelli** sono strumenti di comunicazione tra cliente e sviluppatori
- I **linguaggi di modellazione** più usati sono anche standardizzati (UML è standard OMG)

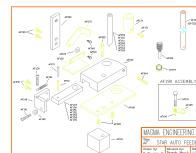
La documentazione del sw

Quanta documentazione è "giusta"?

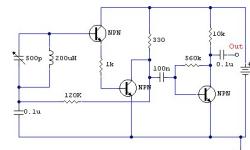
- Prodotto interno IBM (50 KLOC)
 - 28 pagine di documentazione per KLOC
- Prodotto commerciale IBM (50 KLOC)
 - 66 pagine di documentazione per KLOC
- IMS/360 Version 2.3 (166 KLOC)
 - 157 pagine di documentazione per KLOC
- (TRW) Per ogni 100 ore di codifica, 150-200 ore di attività di documentazione

Documentazione visuale

- I modelli del software sono simili a...



Disegni
Meccanici



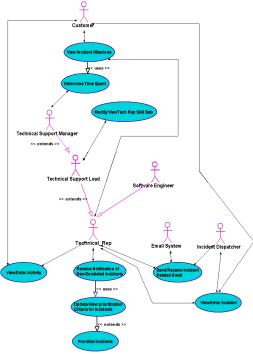
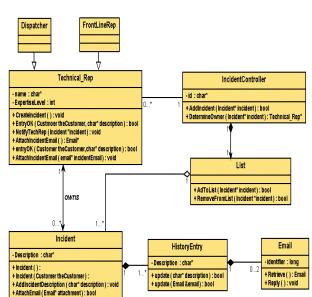
Schemi
Elettrici



Progetti
Edili

Descrizioni visuali del software

Le descrizioni grafiche del software sono più facili da capire (rispetto al codice sorgente), se ci si è educati al loro uso e significato



Unified Modeling Language

Una notazione - standard industriale - per:

- Modellare un ambito aziendale
- Esprimere i requisiti del software
- Esprimere l'architettura software
- Esprimere la struttura ed il comportamento del software
- Documentare l'operatività del software

Riferimento industriale internazionale

Versione corrente delle Specifiche OMG: 1.5

UML e il processo di sviluppo

- UML è solo una **notazione** standard
- Non specifica affatto il processo, cioè il modo in cui dev'essere usata la notazione
- UML è più efficace se viene esplicitamente combinato con un processo di sviluppo
- Gli inventori di UML raccomandano la sua combinazione con RUP

Unified Modelling Language

- UML è un sistema di notazioni principalmente grafiche (con sintassi, semantica e pragmatica predefinite) per la modellazione OO
- UML non è un processo, né è una notazione proprietaria
- È uno standard OMG (Object Management Group), definito mediante un **metamodello**
- UML include:
 - **Viste** (mostrano diverse facce del sistema: utente, strutturale, operazionale, ecc., anche in relazione al processo di sviluppo)
 - **Diagrammi** (grafi che descrivono i contenuti di una vista)
 - **Elementi di modellazione** (costrutti usati nei diagrammi)

Storia di UML

All'inizio degli anni '90 tre metodi convergono:

- **Metodo Booch** (Grady Booch)
 - **OMT** (Jim Rumbaugh)
 - **Fusion/OOSE** (Ivar Jacobson)
- '94 – creano **Rational Software Corporation**
'95 – arriva a Rational

1995 → “Unified” Modeling Language
(versione 0.8)

Unified Modeling Language

- La prima versione pubblicata di UML fu numerata 0.8 perché la specifica sarebbe cambiata...
- in base alle reazioni dei clienti:
 - Versione 0.9 rilasciata in giugno 1996
 - Versione 0.91 rilasciata in ottobre 1996
- La notazione ebbe successo immediato

Unified Modeling Language

Aziende che usano UML:

UML 1.0 viene sottomessa per la standardizzazione a OMG-Gennaio 1997
La versione corrente è 1.5 (2003)

**UML è uno standard internazionale,
e non è proprietà di una singola azienda**

UML – a che serve?

- UML descrive software (e servizi) per:
 - L'utente
 - Lo sviluppatore
 - Il dirigente
 - Il cliente

UML – a che serve?

- Può descrivere
 - L'*uso* del software
 - Come *funziona*
 - Come *va costruito*
 - L'*accordo* (contratto) tra cliente e sviluppatore

Componenti di UML

I principali diagrammi componenti di una specifica UML, in tutte le versioni, sono:

- Diagrammi dei **Casi d'Uso**
- Diagrammi delle **Classi**
- Diagrammi di **Sequenza**
- Diagrammi di **Collaborazione**

Ce ne sono altri:

- Componenti, Deployment, ecc.

I diagrammi canonici (UML 1.5)

- Casi d'uso
- Classe
- Comportamento
 - Statecharts
 - Attività
 - Interazione
 - Sequenza
 - Collaborazione
- Implementazione
 - Componenti
 - Deployment

Diagrammi UML e ciclo di vita

... Requisiti ... Design ... Implementazione

Casi d'Uso

Diagrammi delle classi

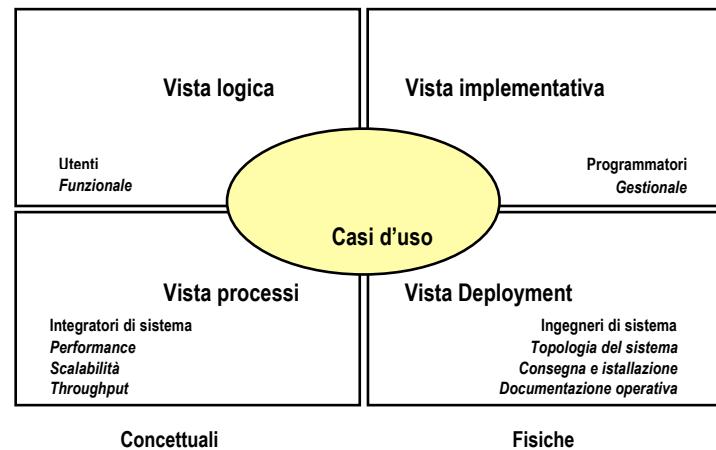
Diagrammi di sequenza

Diagrammi delle attività e State Chart

Analisi = Processo + Modelli

Processo	Modello prodotto
1. Elicitazione dei requisiti d'utente e identificazione dei casi d'uso	Diagrammi e scenari dei casi d'uso
2. Estrazione delle classi candidate, identificazione degli attributi e dei metodi, definizione della gerarchia delle classi	Schede Classe- Responsabilità Collaboratore (CRC)
3. Costruzione di un modello a oggetti e relazioni	Diagramma delle classi
4. Costruzione di un modello operazionale degli oggetti	Diagramma delle interazioni

L'architettura di sistema in UML



Casi d'uso

- Un Caso d'Uso è una **vista** sul sistema che mostra il suo comportamento come apparirà agli utenti esterni
- Partiziona le funzionalità in *transazioni* ('casi d'uso') significative per gli utenti ('attori').
- Consiste di scenari - interazioni tipiche tra un ruolo d'utente ed un sistema informatico
- Proprietà:
 - Mostra funzioni visibili all'utente
 - Raggiunge obiettivi specifici per l'utente
 - Non rappresenta l'ordine o il numero di volte che una funzione viene eseguita
- Si costruisce dialogando con l'utente (*elicitazione*)
- Si usa per costruire i modelli sia strutturali che operazionali, e inoltre per predisporre casi di test

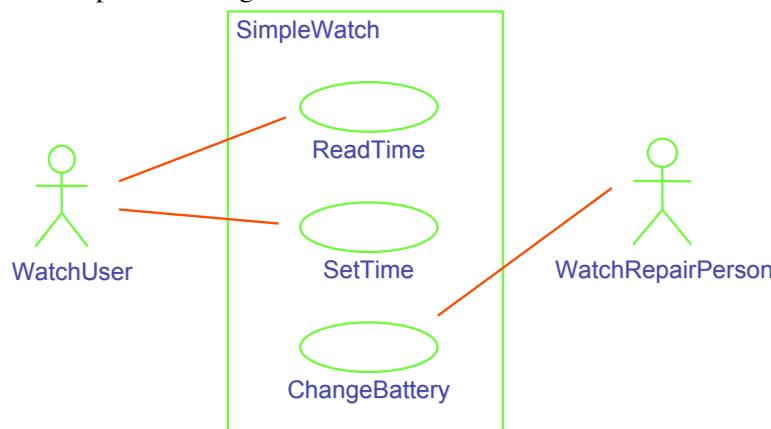
Il Caso d'Uso

- **Tutto** viene costruito su un “**Caso d'Uso**”

“Il Caso d'Uso è una descrizione del **comportamento** di un sistema dal **punto di vista** di un utente”

Esempio

Caso d'uso per un orologio a batteria



Caso d'uso

Ogni caso d'uso ha (almeno):

- Un “**Attore**”
- Un compito (o **comportamento**)

Nota: non c'e' bisogno di usare la grafica:
Andrebbe bene anche la frase
“Impiegato Trasferisce Soldi”

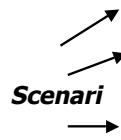
OMG UML 1.5

"A use case diagram is a graph of actors, a set of use cases, possibly some interfaces, and the relationships between these elements. The relationships are associations between the actors and the use cases, generalizations between the actors, and generalizations, extends, and includes among the use cases. The use cases may optionally be enclosed by a rectangle that represents the boundary of the containing system or classifier."

Un “attore”

- Un “**utente**” del sistema
- Descrive un **ruolo**
- Può essere una **persona** o **un altro sistema**
- L’attore ha un “**obiettivo**” da raggiungere usando il sistema
- L’attore è **esterno** al sistema

Uno “scenario”

- Una particolare istanza di un Caso d’Uso
 - Esempio:
 - Una persona riesce a ricevere denaro (*successo*)
 - Una persona non riesce a ricevere denaro *perché il PIN è sbagliato*
 - Una persona non riesce a ricevere denaro *perché il suo conto è a zero*
- Scenari**
- 

Il flusso degli eventi

- Ogni scenario ha un flusso di eventi: è un’*agenda* di cose che debbono accadere
 - **Interazione tra attore e sistema**
 - **Non ci si preoccupa** del *perché* o del *come*
 - Cattura **che cosa** il sistema dovrebbe fare

La “Precondizione”

- Ciò che deve essere vero **prima** che il caso d’uso possa **iniziare**
- Esempi
 - L’utente deve connettersi
 - Il sistema dev’essere attivo
 - Deve esistere una connessione di rete
 - Ecc.

La “Post condizione”

- **Ciò che** sarà *vero al termine* del caso d’uso
- ...o “*come sappiamo*” che il caso d’uso è terminato.
 - Il denaro è stato erogato
 - Lo scontrino è stato stampato
 - L’ordine è stato inviato

La “descrizione”

- A che serve il Caso d’Uso?
 - **Cosa si aspetta l’attore?**
- Siate **brevi!** Se occorrono più di due righe per spiegare il caso d’uso, questo è spesso segnale che va spezzato

Modellazione dei casi d’uso

- Uno scenario è una sequenza di passi che descrivono l’interazione tra un utente ed un sistema
- Un caso d’uso è un insieme di scenari legati da un obiettivo comune
- Un caso d’uso consiste di:
 - Un unico nome
 - Attori partecipanti
 - Condizioni d’ingresso (precondizioni)
 - Flusso degli eventi
 - Condizioni d’uscita (postcondizioni)

Esempio di scenario: negozio on line

- **Attori:** cliente, sistema
- **Precondizioni:**
 - il cliente accede alla pagina Web
- **Flusso normale:**
 1. Il cliente naviga nel catalogo e seleziona degli articoli
 2. Il cliente si avvia alla "cassa"
 3. Il cliente indica le informazioni relative alla spedizione
 4. Il sistema presenta il prospetto
 5. Il cliente inserisce i dati della carta di credito
 6. Il sistema autorizza l'acquisto
 7. Il sistema conferma l'acquisto
- **Flusso alternativo:**
 - carta di credito non valida....
 - cliente abituale → pagina personalizzata...

Diagramma delle classi

- Serve allo **sviluppatore**
- Inizia la descrizione di *come* verrà **costruito** il sistema

Diagramma delle classi

Mostra *quali componenti* sw verranno *usati* e come si *relazionano* gli uni agli altri

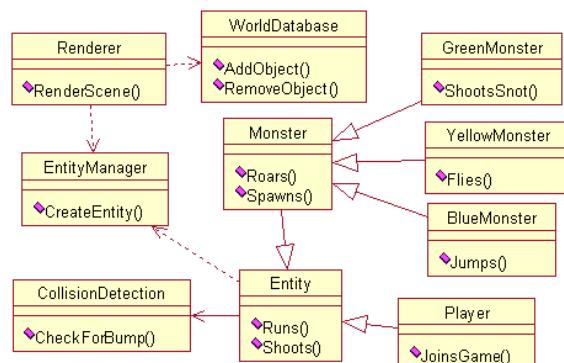


Diagramma di sequenza

Mostra la sequenza **temporale** delle interazioni tra oggetti

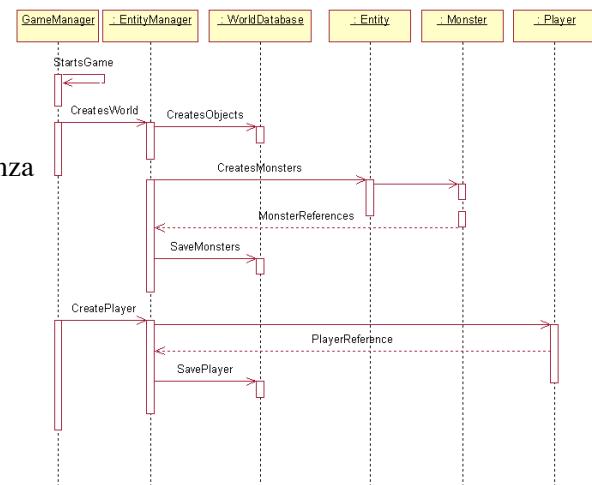
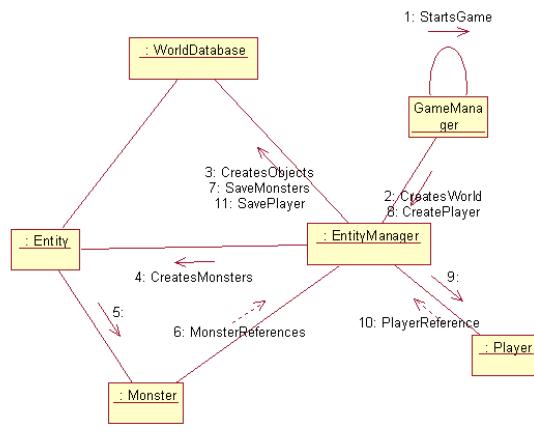


Diagramma delle collaborazioni

Mostra le
interazioni
tra gli *oggetti*



Esempio

- L'ateneo deve informatizzare il sistema di immatricolazione
 - La Segreteria definisce il curriculum di un semestre
 - Ci sono molteplici offerte di corsi
 - Gli studenti scelgono 4 corsi principali e 2 secondari
 - Quando uno studente si registra per un semestre, il sistema di contabilità viene avvertito in modo che possa emettere fattura allo studente
 - Gli studenti possono aggiungere o togliere corsi per un certo periodo dopo la registrazione
 - I professori usano il sistema per avere la lista degli iscritti ai loro corsi
 - Gli utenti del sistema sono autenticati mediante login e password

Attori



Segreteria



Professore



Studente



Contabilità

Casi d'uso

- Un caso d'uso è uno schema di comportamento del sistema
 - Ogni caso d'uso è una sequenza di transazioni correlate eseguite in un dialogo tra un attore ed il sistema
- Quali sono i bisogni dei singoli attori?
 - Segreteria -- gestisce i curricula degli studenti
 - Professore -- richiede lista iscritti
 - Studente -- gestisce il suo piano di studi
 - Contabilità -- riceve informazioni dalla segreteria



Gestisci Curriculum



Richiedi iscritti

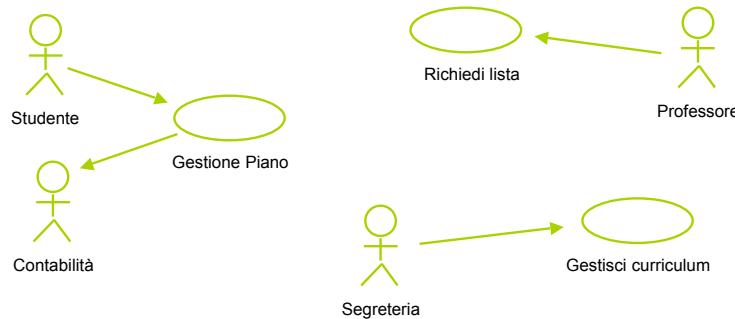


Gestisci piano

Documentare i Casi d'uso

- Per ciascun Caso d'Uso si scrive un documento “flusso di eventi”
 - Scritto dalla prospettiva di un singolo attore
- Definisce i dettagli di cosa il sistema deve fornire all'attore quando i casi d'uso vengono eseguiti
- Contenuti tipici
 - Come inizia e finisce il caso d'uso
 - Flusso normale degli eventi
 - Flusso alternativo degli eventi
 - Flusso eccezionale degli eventi

Diagrammi Casi d'uso

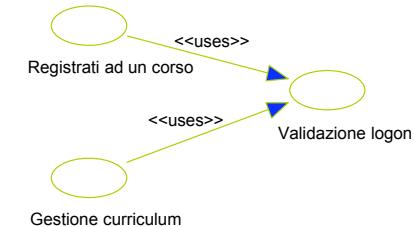


Flusso di eventi Gestione curriculum

- Questo Caso d'Uso inizia quando la Segreteria si collega al Sistema ed inserisce la password. Il sistema verifica che la password sia valida (E-1) e chiede di selezionare il semestre corrente o uno futuro (E-2). La Segreteria inserisce il semestre desiderato. Il sistema chiede di inserire un comando: ADD, DELETE, REVIEW, o QUIT.
- Se il comando scelto è ADD, viene eseguito S-1: Aggiungi un corso.
- Se il comando scelto è DELETE, viene eseguito S-2: Cancella un Corso.
- Se il comando scelto è REVIEW, viene eseguito S-3: Rivedi Curriculum.
- Se il comando scelto è QUIT, il caso d'uso termina.
- ...

Casi d'Uso: relazioni “uses” e “extends”

- Mentre si costruisce un Caso d'Uso, possono scoprirsene degli altri
 - Una relazione “uses” mostra un comportamento comune a più Casi d'Uso
 - Una relazione “extends” mostra comportamento opzionale



Supporto ai Casi d'Uso

- Il diagramma dei Casi d'Uso presenta una vista esterna del sistema
- I diagrammi di interazione descrivono come i casi d'uso vengono realizzati come interazione tra società di oggetti
- Si possono usare due tipi di diagrammi di interazione
 - Diagrammi di sequenza
 - Diagrammi di collaborazione

Diagramma di sequenza

- Un diagramma di sequenza mostra le interazioni degli oggetti in sequenza temporale

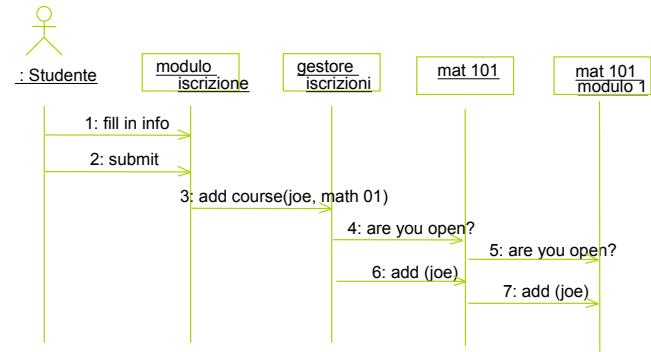
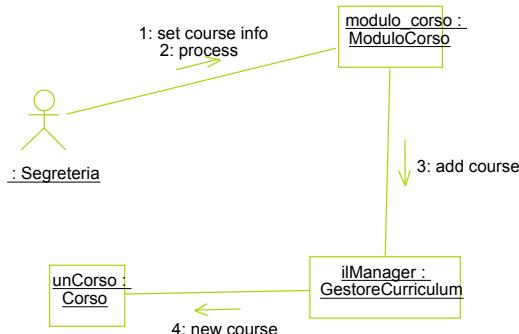


Diagramma di collaborazione

- Un diagramma di collaborazione mostra le interazioni tra gli oggetti mediante collegamenti



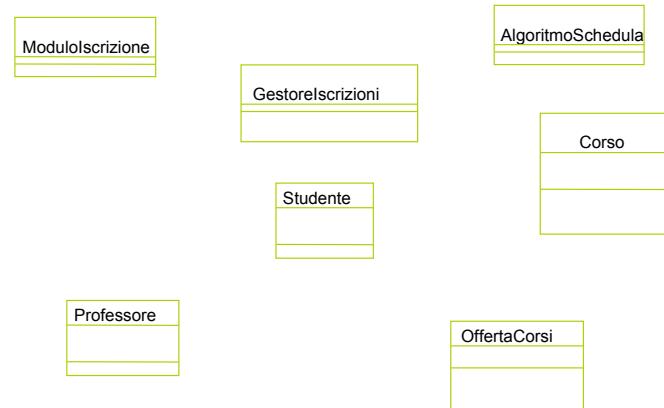
Diagrammi delle classi

- Un diagramma delle classi mostra le classi e le loro relazioni nella vista logica di un sistema
- Elementi di modellazione in questi diagrammi:
 - **Classi**, con struttura e comportamento
 - **Legami** di Associazione, aggregazione, dipendenza, ereditarietà
 - **Indicatori** di molteplicità e navigazione
 - **Nomi** di ruoli

Classi

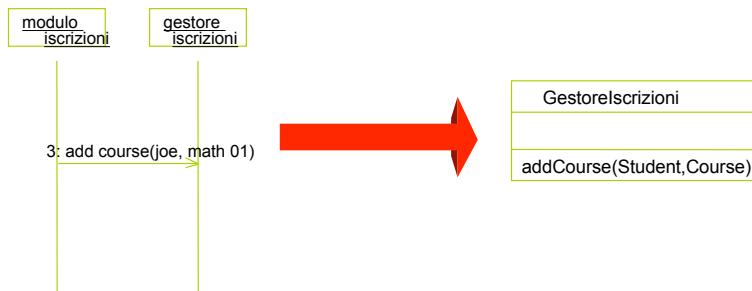
- Una classe è una collezione di oggetti con struttura, comportamento, relazioni e semantica condivisi
- Le classi si ottengono esaminando gli oggetti presenti nei diagrammi di sequenza e collaborazione
- Una classe si rappresenta con un rettangolo a tre compartimenti
- I nomi delle classi derivano dal vocabolario del dominio
 - Occorrono standard di denominazione
 - Esempio: le classi sono nomi al singolare che iniziano con lettera maiuscola

Classi



Operazioni

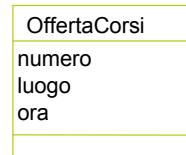
- Le operazioni definiscono il comportamento di una classe
- Le operazioni derivano dai diagrammi di interazione



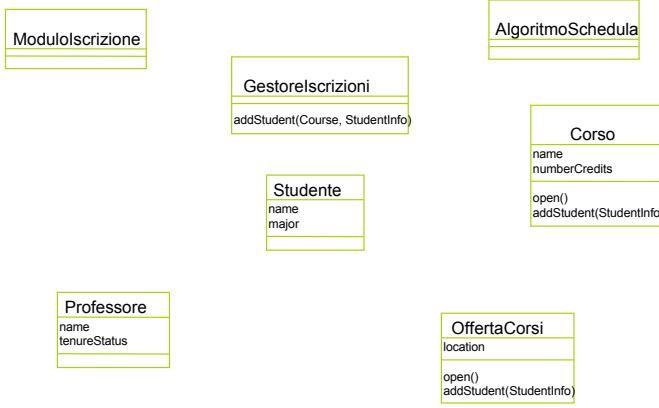
Attributi

- Gli attributi definiscono la struttura di una classe
- Gli attributi derivano dalla definizione dell'aclasses, dai requisiti del problema, e dalla conoscenza del dominio

Ogni corso ha un id numerico, un'aula e un'ora della settimana



Classi

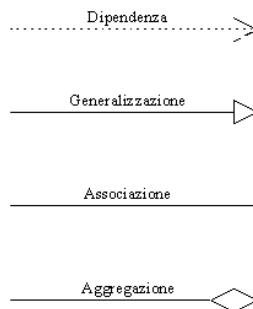


Relazioni

- Le relazioni permettono agli oggetti di comunicare o interagire
- Si esaminano i diagrammi di sequenza e di collaborazione per determinare quali legami relazionali tra oggetti dovrebbero esistere per realizzare il comportamento voluto
- Tre tipi di relazione:
 - Associazione
 - Aggregazione
 - Dipendenza

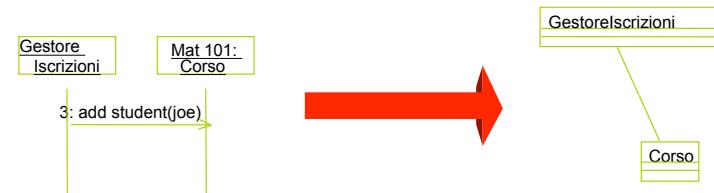
Relazioni

- Un' **associazione** è una connessione bidirezionale tra classi
 - Un' associazione si rappresenta con una linea che collega due classi
- Un' **aggregazione** è una forma più forte di relazione: tra un intero e le sue parti
 - Anche le aggregazioni sono rappresentate da linee: in più si mette un ◆ (“diamond”) vicino alla classe che rappresenta l'intero
- Una **dipendenza** è una forma più debole di relazione: tra un cliente ed un fornitore di servizio, in cui il cliente non ha conoscenza semantica del fornitore
 - Le dipendenze sono rappresentate da frecce tratteggiate che vanno dal cliente al fornitore

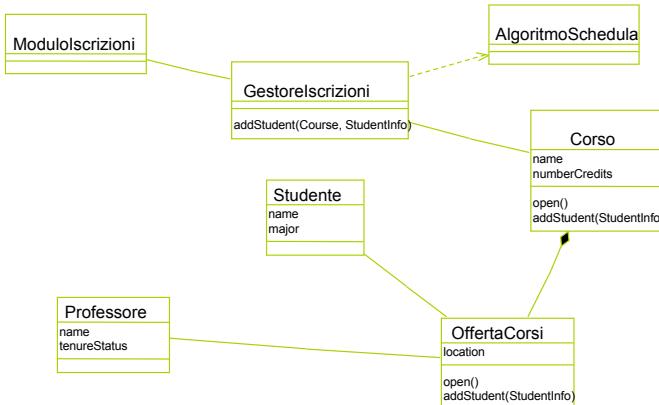


Trovare le relazioni

- Le relazioni si scoprono esaminando i diagrammi di interazione
 - Se due oggetti debbono “comunicare” deve esistere un canale di comunicazione



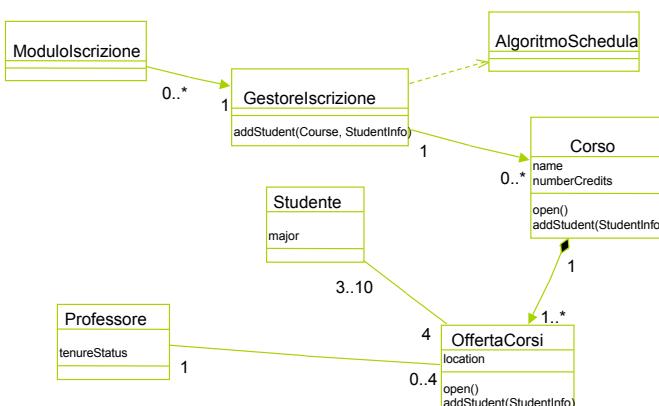
Relazioni



Molteplicità e Navigazione

- La *molteplicità* definisce quanti oggetti partecipano ad una relazione
 - La molteplicità è il numero di istanze di una classe legate ad *una* istanza di un'altra classe
 - Per ogni associazione o aggregazione occorre decidere due molteplicità, una per ciascun capo del legame
- Sebbene aggregazioni e associazioni siano bidirezionali, spesso è desiderabile restringere la navigazione ad una sola direzione
- Se la *navigazione* è vincolata, si aggiunge una freccia per denotare la direzione di navigazione

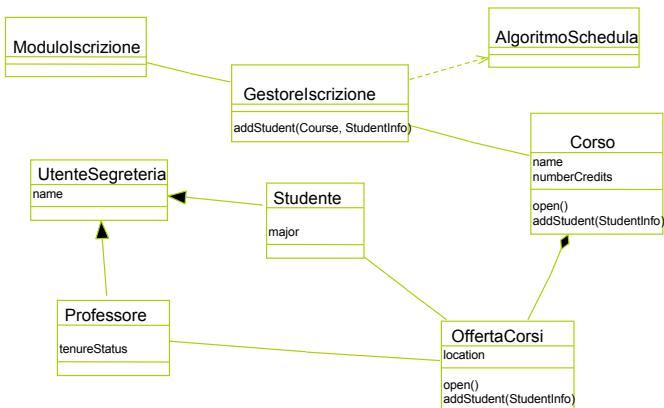
Molteplicità e Navigazione



Ereditarietà

- L'ereditarietà è una relazione tra una superclasse e le sue sottoclassi; forma una gerarchia
- Ci sono due modi per trovare ereditarietà:
 - Generalizzazione (bottom up)
 - Specializzazione (top down)
- Attributi, operazioni e relazioni a comune vengono mostrati al livello più alto possibile della gerarchia

Ereditarietà

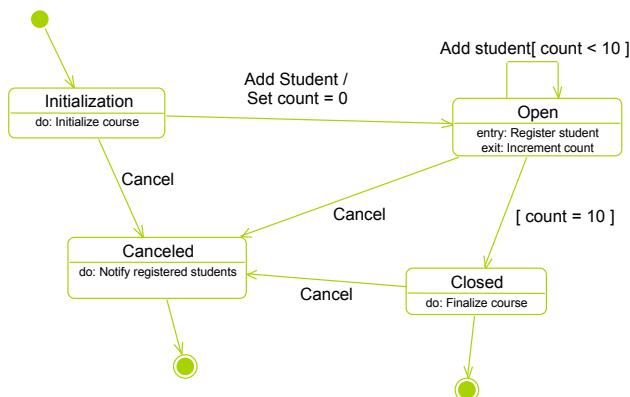


Lo Stato di un Oggetto

- Un diagramma di transizione di stato mostra
 - Il ciclo di vita di un oggetto di una classe
 - Gli eventi che causano una transizione da uno stato all’altro a
 - Le azioni risultanti da un cambiamento di stato
- Questi diagrammi si creano solo per gli oggetti con comportamento dinamico significativo

Diagramma di Transizione di Stato

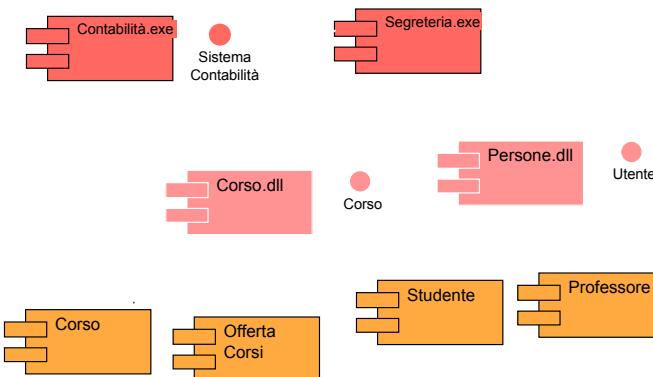
Oggetto: Corso



Il mondo fisico

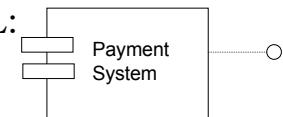
- I diagrammi dei componenti illustrano l’organizzazione e le dipendenze tra i componenti software
- Un componente può essere
 - Codice sorgente
 - RunTime (libreria)
 - Eseguibile

Diagramma dei componenti



Componenti

- Icona di componente in UML:



- Un componente software è “*un’entità software non banale: un modulo, un package o un sottosistema che abbia una funzione precisa ed un contorno ben delineato, e che possa essere integrato in un’architettura ben definita*”
- Concetti chiave in questa definizione:
 - non banale
 - funzione precisa
 - contorno ben delineato

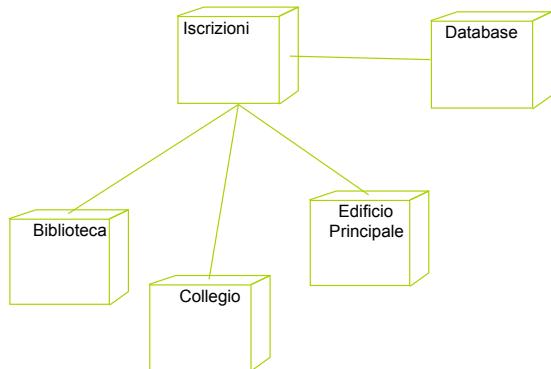
Componenti e architetture

- Le architetture modulari sono fatte di componenti ben formati
 - Occorre identificare, progettare, sviluppare e testare i componenti
 - Prima vengono testati da soli, poi integrati e testati nel sistema
- I componenti dovrebbero essere **riusabili**
 - Specialmente se risolvono problemi ricorrenti
 - Sono qualcosa di più che utilità di libreria
- Ogni organizzazione costruisce il **proprio** insieme di componenti riusabili
- I componenti si possono comprare già fatti (COTS: Components Off The Shelf): lo sviluppo di un sistema sw allora si riduce a trovare, integrare e testare componenti

Configurare il sistema

- Il diagramma di deployment mostra la configurazione a run time degli elementi di calcolo e dei processi software che vivono in essi
- Il diagramma di deployment visualizza la distribuzione dei componenti all’interno dell’azienda

Diagramma di Deployment



Il RUP

- **Inception:** Concentrarsi su obiettivo e requisiti
- **Elaboration:** Concentrarsi sull'analisi dei requisiti, pensando al design ed alla modellazione
- **Construction:** Concentrarsi su progetto e implementazione, costruendo prime versioni preliminari e poi versioni migliorate del prodotto
- **Transition:** Concentrarsi sulla qualità del prodotto, sul suo deployment, sulla formazione e supporto

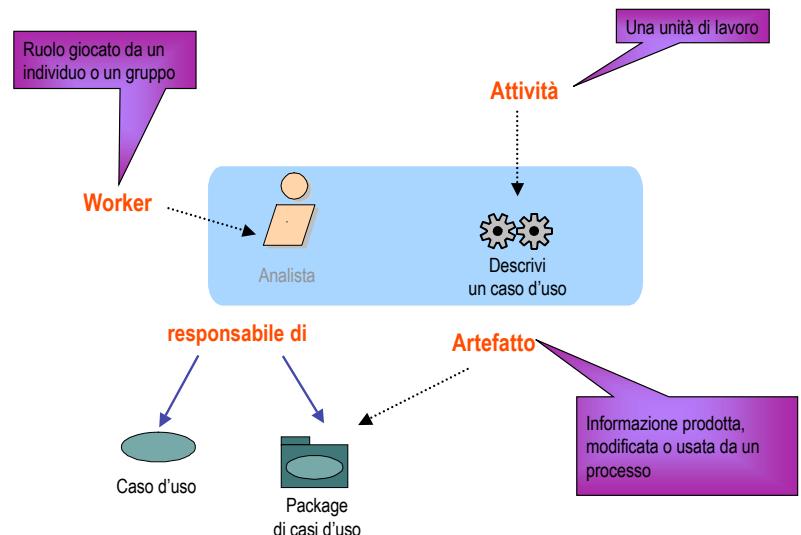
Elementi dei processi basati su RUP

Un processo descrive chi fa cosa, come e quando.

Il RUP si basa su 4 elementi principali:

- **Chi :** Workers
- **Come :** Attività
- **Cosa :** Artefatti
- **Quando :** Workflows

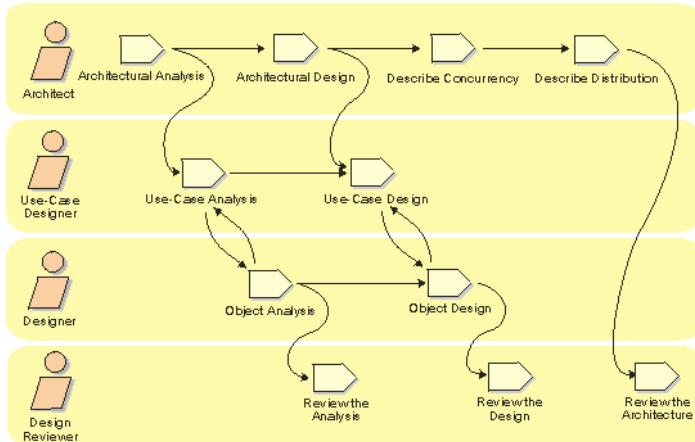
Elementi base di RUP



Workflow

I workflow descrivono

- Quando i workers eseguono attività per produrre artefatti
- Le attività e gli artefatti che dovrebbero produrre



Worker

Un worker definisce comportamenti e responsabilità di un individuo o di un gruppo di individui che lavorano in squadra: duqne un worker è un **ruolo**

Ad ogni worker viene associato un insieme coeso di attività

- “coeso” significa che tali attività vanno logicamente eseguite dalla stessa persona

Il worker è responsabile di creare, modificare e controllare certi artefatti

Esempio di Worker

- Analista di sistema: **individuo responsabile** di dirigere e coordinare l’elicitazione dei requisiti e la modellazione dei casi d’uso, schematizzando le funzionalità del sistema e delimitandone i contorni

Attività

- I worker eseguono attività; queste definiscono il lavoro eseguito dai worker. Infatti ogni attività è assegnata ad un worker specifico, per esempio un individuo responsabile della sua esecuzione
- Un’attività è “una unità di lavoro che un individuo in quel ruolo può dover eseguire e che produce un risultato significativo per il progetto”
- Le attività hanno uno scopo preciso; di solito creano un artefatto
- La granularità temporale di un’attività va da poche ore ad alcuni giorni
- Le attività sono soggette a pianificazione e tracciamento

Passi di una attività

Le attività sono articolate su tre tipi di passi

- Passi di **riflessione** (*Thinking*): Comprendere la natura del compito, raccogliere ed esaminare gli artefatti iniziali. Riflettere sul risultato
- Passi di **esecutivi** (*Performing*): Creare o aggiornare artefatti
- Passi di **revisione** (*Reviewing*): Ispezionare e valutare gli artefatti secondo qualche criterio

Esempio di attività

Attività: Trovare Casi d'uso e attori

Nel RUP si esprime così:

1. Trova attori (Thinking)
2. Trova Casi d'Uso (Thinking)
3. Descrivere interazione di attori e casi d'uso (Performing)
4. Impacchettare attori e casi d'uso (Performing)
5. Presentare modello dei casi d'uso in diagramma (Performing)
6. Riassumere il modello dei casi d'uso (Performing)
7. Valutare il risultato (Reviewing)

Artefatti

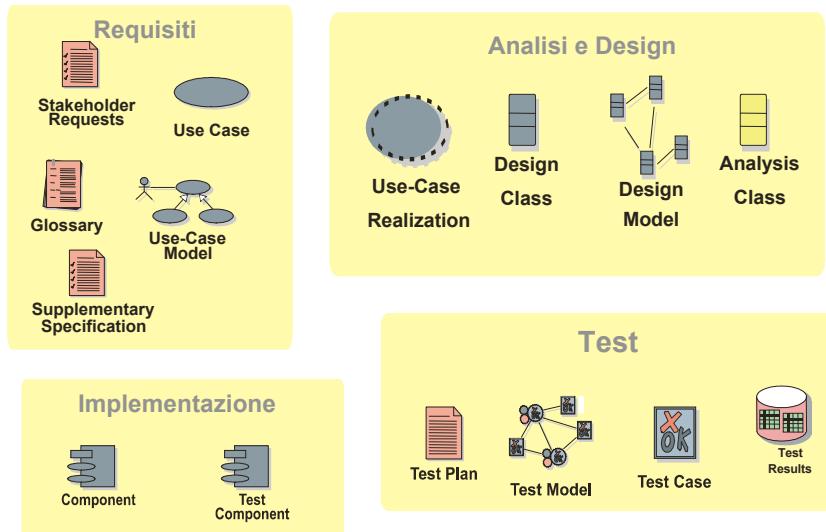
Le attività hanno **artefatti** in input e output

In terminologia RUP, un artefatto è un documento prodotto, modificato o usato da una attività, quindi è un prodotto "tangibile" del processo

Esempi

- Un **modello**, ad esempio dei casi d'uso
- Un **elemento** di un modello, ad esempio una classe n caso d'uso o un diagramma
- Un **documento**, ad esempio lo studio di fattibilità
- Il **sorgente** di un programma
- Un file contenente codice **eseguibile**

Esempio: artefatti di processo



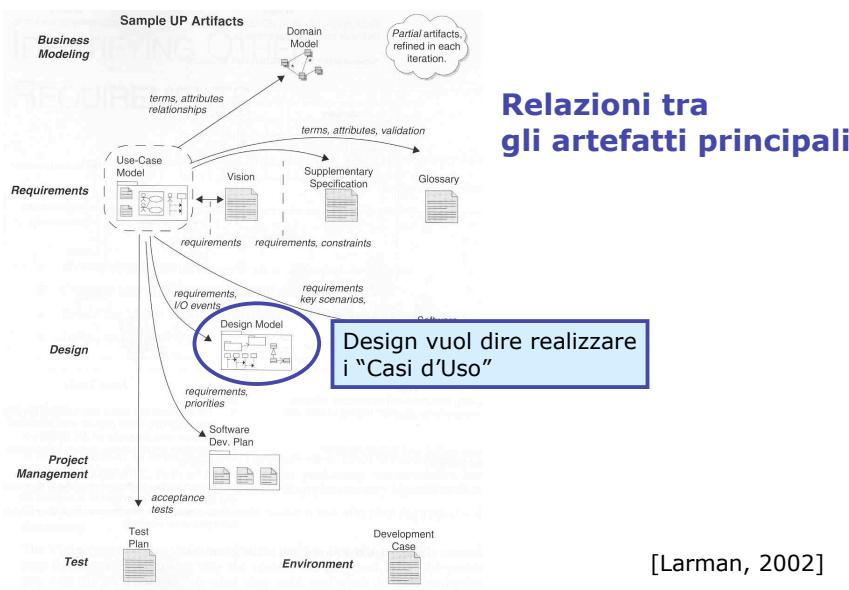
Esempio: artefatti di supporto

Artefatti

- Alcuni artefatti sono “*deliverables*”, cioè vengono consegnati e “congelati”
- Un artefatto può includere altri artefatti.
- Gli artefatti spesso hanno bisogno di controllo delle versioni.
- Gli artefatti RUP tipicamente NON sono documenti cartacei; di solito sono documenti digitali on line, facili da modificare e dunque almeno in teoria sempre aggiornati

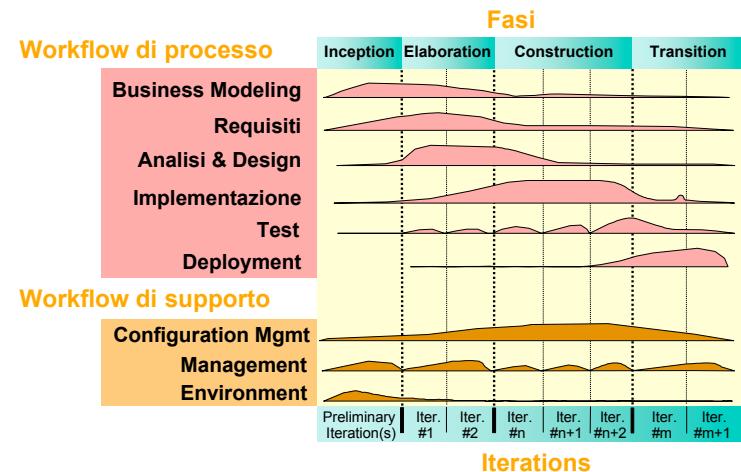
Pianificazione degli artefatti RUP

Discipline	Artifact Iteration→	Incep.	Elab.	Const.	Trans.
		I1	E1..En	C1..Cn	T1..T2
Business Modeling	Domain Model			s	
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model			s	r
	SW Architecture Document			s	
	Data Model			s	r
Implementation	Implementation Model			s	r
Project Management	SW Development Plan	s	r	r	r
Testing	Test Model			s	r
Environment	Development Case	s	r		



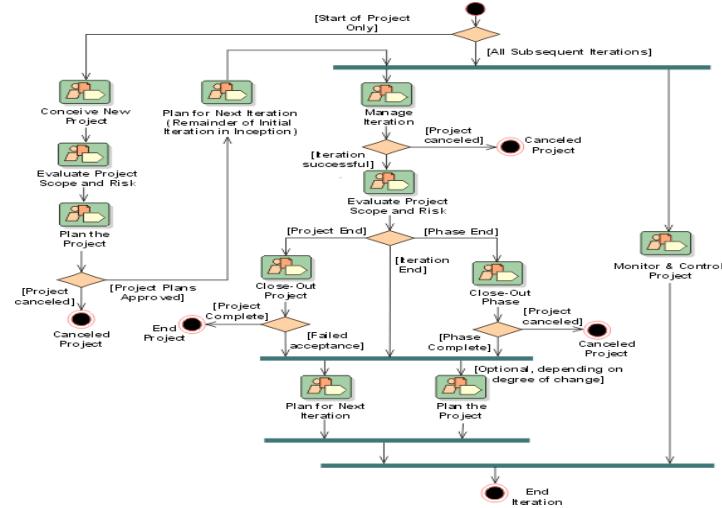
Il RUP stesso è un workflow

RUP stesso: 9 core workflows, 31 workers, 103 artefatti, 136 attività



Artefatti di Inception

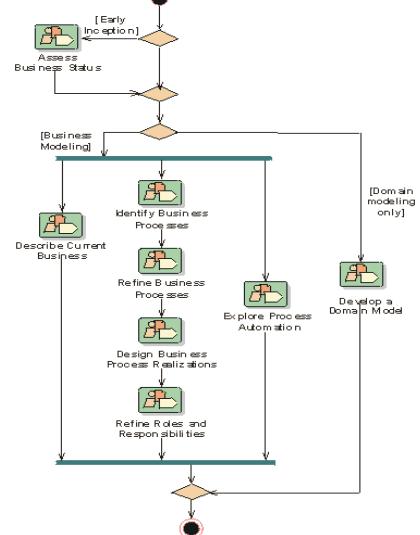
Workflow di Project Management



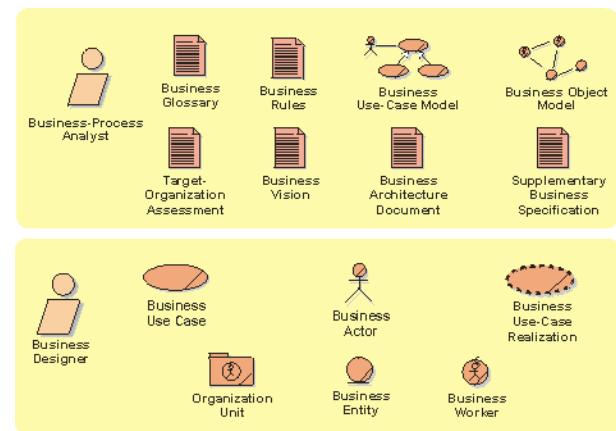
Artifact [†]	Comment
Vision and Business Case	Describes the high-level goals and constraints, the business
Use-Case Model	• Gli artefatti sono opzionali
Supplementary Specific	• Possono essere abbozzati
Glossary	• Scrivere quanto necessario per le decisioni
Risk List & Risk Manag	che si debbono prendere
Plan	• Può essere necessaria qualche prototipazione...
Prototypes and proof-of-concepts	and ideas for their mitigation or response.
Iteration Plan	To clarify the vision, and validate technical ideas.
Phase Plan & Software Develop	Describes what to do in the first elaboration iteration.
ment Plan	Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

[Larman, 2002]

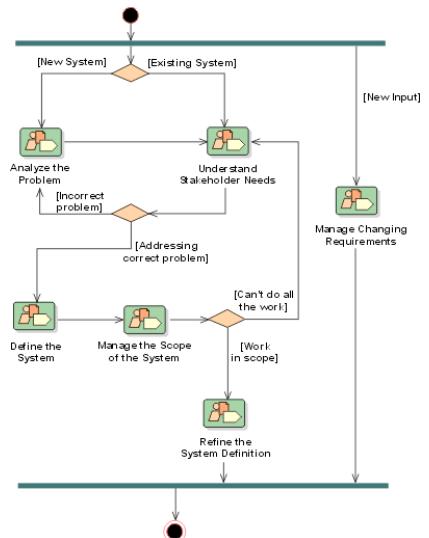
Workflow Business Modelling



Artefatti di Business Modelling



Workflow dei requisiti



Il Workflow dei Requisiti

Lo scopo del workflow dei Requisiti:

- Costruire e conservare il consenso degli stakeholder sull'obiettivo del sistema
- Migliorare la comprensione del sistema da parte degli sviluppatori
- Delimitare i confini del sistema.
- Fornire un punto di partenza per pianificare i contenuti tecnici delle iterazioni
- Base di stima di costi e durata dello sviluppo del sistema
- Definire l'interfaccia utente del sistema esplicitando gli obiettivi ed i bisogni degli utenti

I Requisiti diventano Casi d'Uso

- Il workflow dei requisiti è ispirato dalle idee di Ivar Jacobson sui Casi d'Uso (*metodo Objectory*)
- Il workflow dei requisiti produce i requisiti di sistema creando un Modello dei Casi d'Uso
- Il Modello dei Casi d'Uso è un insieme di:
 - Casi d'Uso che rappresentano specifiche interazioni
 - Attori interagenti col sistema
 - Requisiti non funzionali
 - Prototipi di GUI

Punto di partenza

- Occorre cominciare dalla definizione del problema che si cerca di risolvere col sistema da progettare
- Un input è il Workflow Business Modelling
- Questo dovrebbe contenere
 - Le regole di business rules
 - Il modello dei casi d'uso di Business
 - Il modello degli oggetti di Business
 - Le richieste degli stakeholder

Funzione del Modello “Casi d’Uso”

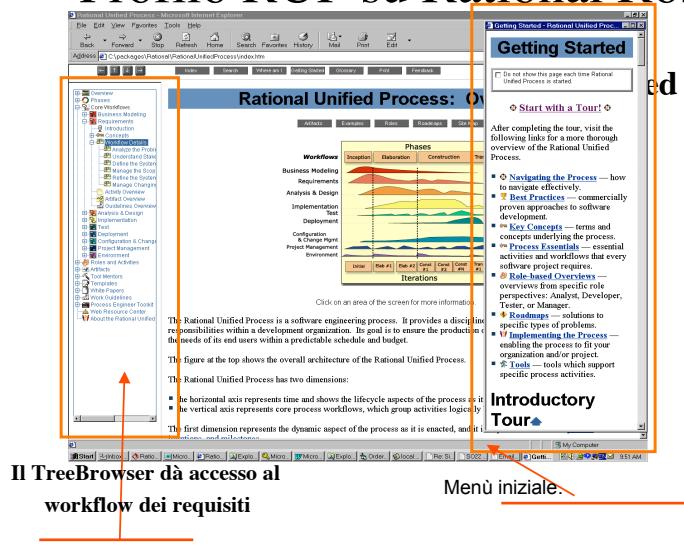
- **“Contratto”**
 - Il modello dei casi d’uso dovrebbe servire come documento di riferimento, quindi come contratto (formale o informale) tra cliente, utenti e sviluppatori
- Permette:
 - A clienti e utenti di convalidare il sistema, nel senso che il prodotto è quel che si aspettavano
 - Agli sviluppatori di costruire ciò che è richiesto

Funzioni del Modello “Casi d’Uso”

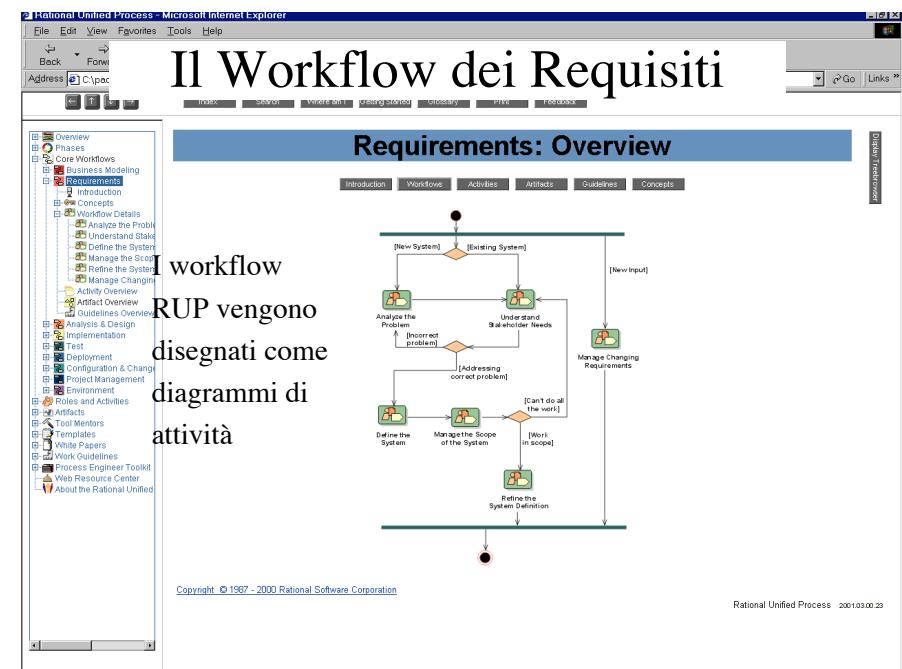
Come “controllo”

- Lo stesso modello dei Casi d’Uso si usa per i workflow Analisi, Design, Implementazione e Testing
- Assicura che tutto il lavoro fatto si ricolleghi ai requisiti reali del sistema
- Il Modello dei Casi d’Uso assicura che il progetto sia controllato e coordinato

Profilo RUP su Rational Rose



Il Workflow dei Requisiti



I workflow
RUP vengono
disegnati come
diagrammi di
attività

Rational Unified Process - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Address C:\pc

Dettaglio di Workflow

Workflow Detail: Analyze the Problem

Topics

- Purpose
- How to Staff
- Work Guidelines

Purpose

The purpose of this workflow detail is to:

- Gain agreement on the problem being solved,
- Identify stakeholders,
- Define the system boundaries, and
- Identify constraints imposed on the system

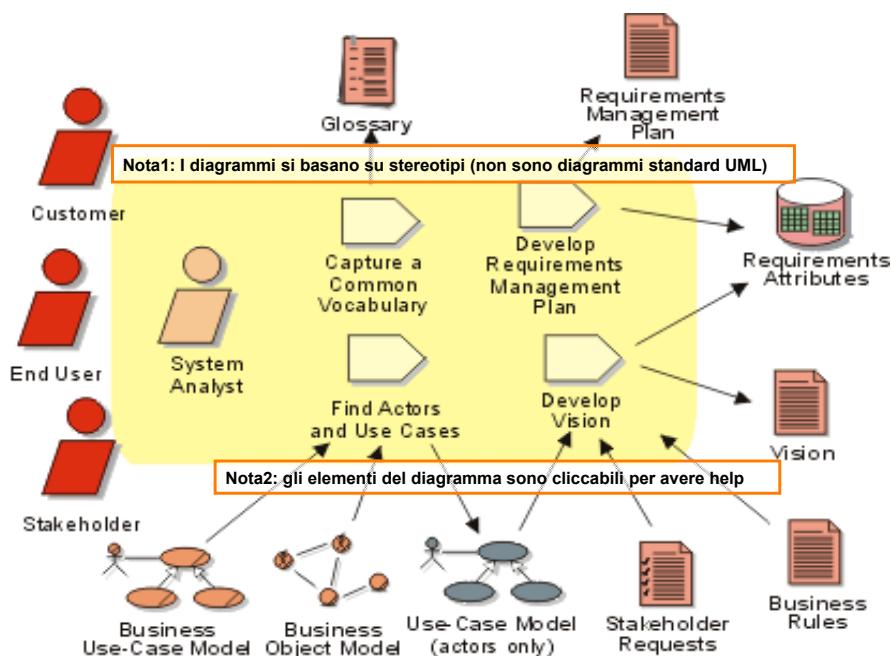
The first step in any problem analysis is to make sure that all parties involved agree on what is the problem that we are trying to solve with our system. In order to avoid misunderstandings, it is important to agree on common terminology which will be used throughout the project. Early on, we should begin defining our project terms in a [glossary](#) which will be maintained throughout the project lifecycle.

In order to fully understand the problem(s) we should be addressing, it is very important to know who are our [stakeholders](#). Note that some of these stakeholders -- the users of the system -- will be represented by [actors](#) in our [use-case model](#).

Esempio: analizza il problema

- Lo scopo di questo dettaglio di workflow è :
 - Stabilire il consenso sul problema da risolvere
 - Identificare gli stakeholder
 - Definire i contorni del sistema
 - Identificare i vincoli imposti sul sistema
- Il primo passo di ogni analisi è assicurarsi che tutte le parti interessate si accordino sul problema da risolvere col sistema da progettare. Per evitare incomprensioni è importante concordare una terminologia comune che verrà usata per tutto il progetto. Occorre iniziare costruendo un glossario dei termini del progetto, che verrà mantenuto per tutta la vita del progetto*

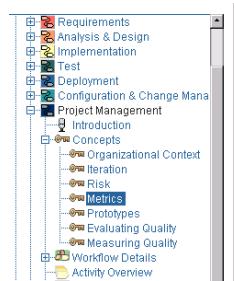
Krutch, 2000



Esempio di guide RUP: misure

- RUP enfatizza le misure
- Metodi di descrizione delle misure
- Metriche predefinite utilizzabili
- Strumenti di supporto alle misurazioni ed alla loro memorizzazione

Guideline RUP per le misure



Concepts: Metrics

Topics

- Why do we Measure?
 - Organizational Needs for Metrics
 - Project Needs for Metrics
 - Technical Needs for Metrics
- What is a Metric?
 - Templates
 - Metric Activities
 - How are the Metrics Used?

Why do we Measure?

Guidelines RUP per le misure

Concepts: Measuring Quality

Topics

- Measuring Quality
- Measuring Product Quality
- Measuring Process Quality

Measuring Quality

The measurement of Quality, whether Product or Process, requires the collection and analysis of information, usually stated in terms of measurements and metrics. Measurements are made primarily to gain control of a project, and therefore be able to manage it. They are also used to evaluate how close or far we are from the objectives set in the plan in terms of completion, quality, compliance to requirements, etc.

Metrics are used to attain two goals, knowledge and change (or achievement):

Knowledge goals: they are expressed by the use of verbs like evaluate, predict, monitor. You want to better understand your development process. For example, you may want to assess product quality, obtain data to predict testing effort, monitor test coverage, or track requirements changes.

Change or achievement goals: these are expressed by the use of verbs such as increase, reduce, improve, or achieve. You are usually interested in seeing how things change or improve over time, from an iteration to another, from a project to another.

Guidelines RUP per le misure

Guidelines: Metrics

Topics

- Principles
- A Taxonomy of Metrics
- A Small Set of Metrics
- A Complete Metrics Set
 - What Should be Measured?
 - The Process
 - The Product
 - The Project
 - The Resources

Principles

- Metrics must be simple, objective, easy to collect, easy to interpret, and hard to misinterpret.
- Metrics collection must be automated and non-intrusive, i.e., not interfere with the activities of the developers.
- Metrics must contribute to quality assessment early in the lifecycle, when efforts to improve software quality are effective.
- Metric absolute values and trends must be actively used by management personnel and engineering personnel for communicating progress and quality in a consistent format.
- The selection of a minimal or more extensive set of metrics will depend on the project's characteristics and context: if it is large or has stringent safety or reliability requirements and the development and assessment teams are knowledgeable about metrics, then it may be useful to collect and analyze the technical metrics.

Guidelines RUP per le misure

Metrics and Primitives metrics

Total SLOC	SLOCt = Total size of the code
SLOC under configuration control	SLOCc = Current baseline
Critical defects	SCO0 = number of type 0 SCO
Normal defects	SCO1 = number of type 1 SCO
Improvement requests	SCO2 = number of type 2 SCO
New features	SCO3 = number of type 3 SCO
Number of SCO	N = SCO0 + SCO1 + SCO2
Open Rework (breakage)	B = cumulative broken SLOC due to SCO1 and SCO2
Closed rework (fixes)	F = cumulative fixed SLOC
Rework effort	E = cumulative effort expended fixing type 0/1/2 SCO
Usage time	UT = hours that a given baseline has been operating under realistic usage scenarios

Guidelines RUP per le misure

Quality Metrics for the End-Product

From this small set of metrics, some more interesting metrics can be derived:

Scrap ratio	B/SLOC _t , percentage of product scrapped
Rework ratio	E/Total effort, percentage of rework effort
Modularity	B/N, average breakage per SCO
Adaptability	E/N, average effort per SCO
Maturity	UT/(SCO ₀ + SCO ₁), Mean time between defects
Maintainability	(scrap ratio)/(rework ratio), maintenance productivity

Guidelines RUP per le misure

In-progress Indicators

Rework stability	B - F, breakage versus fixes over time
Rework backlog	(B-F)/SLOC _c , currently open rework
Modularity trend	Modularity, over time
Adaptability trend	Adaptability, over time
Maturity trend	Maturity, over time

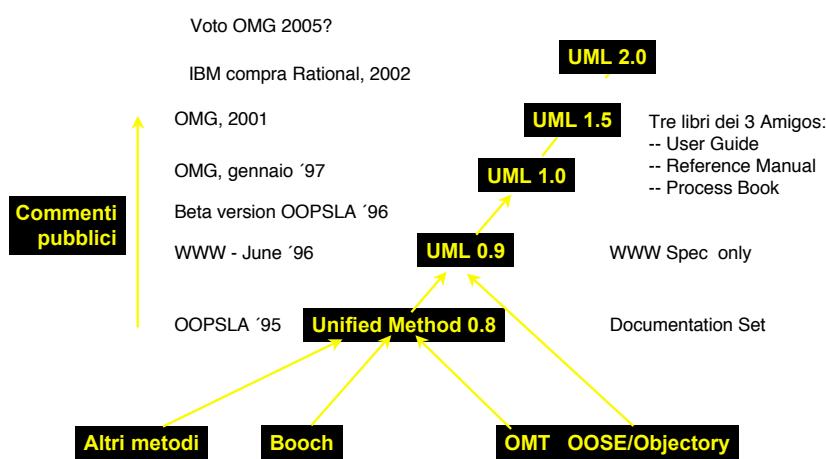
Guidelines RUP per le misure

- Strumenti di supporto
 - ClearQuest
 - ClearCase
 - RequisitePro
 - Project Console

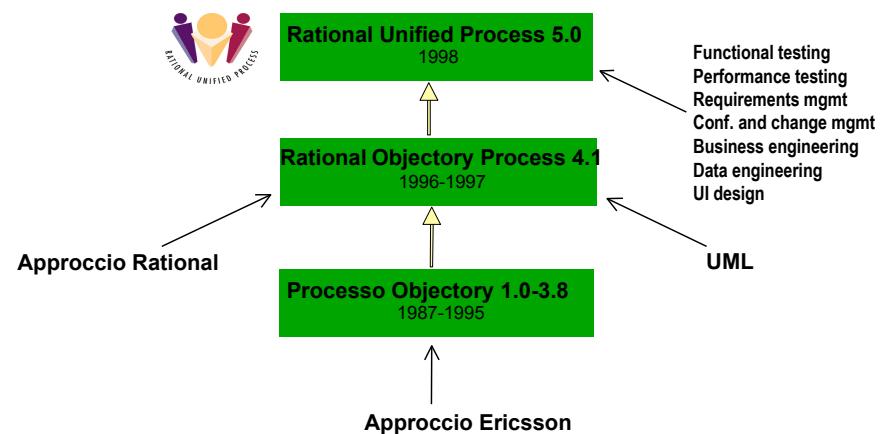
Evoluzione di UML

- UML pubblicato da Rational nel 1995
- UML 1.1: **standard OMG** dal 1997
- UML 1.3: Giugno 1999
- UML 1.4: Settembre 2001
- UML 1.5: Marzo 2003
- UML 2.0: in attesa di omologazione

Evoluzione di UML



Evoluzione di RUP



Riferimenti

- OMG, *UML Specification v. 1.5*. OMG, 2003
- Arlow e Neustadt, *UML e Unified Process*, McGrawHill, 2003
- Larman, *Applying UML and Patterns*, PrenticeHall, 2002
- Conallen, *Building Web Applications with UML*, Addison Wesley, 2000
- Fowler, *UML Distilled – 2nd Edition*, Addison Wesley, 1997
- Rumbaugh, Jacobson, Booch, *The UML Reference Manual*, AW, 1999
- www.rational.com (sito IBM)
- www.agilemodeling.com
- www.omg.org
- alistair.cockburn.us/usecases/usecases.html



Domande?