**Università degli Studi di Parma**

**Facoltà di Scienze MM. FF. NN.**

**Corso di Laurea in Informatica**

**CORSO DI INGEGNERIA DEL SOFTWARE**

**Prof. Giulio Destri**




# GUIDA ALLA REALIZZAZIONE DEL PROGETTO DI ESAME

A.A. 2006-2007

# INTRODUZIONE ALLA REALIZZAZIONE DEL PROGETTO

La realizzazione del progetto del corso di Ingegneria del Software è premessa indispensabile per sostenere l'esame. La metodologia dell'Unified Process, suggerita per la realizzazione, è sicuramente piuttosto complessa, ma dà indicazioni preziose sui passi da seguire per la progettazione del software usando al meglio i diagrammi UML.

Occorre ricordare sempre che i passi della analisi e della progettazione non devono essere vissuti come imposizioni, bensì come un ausilio al mantenimento della precisione e, conseguentemente, di un migliore controllo sull'andamento dei lavori e del progetto in toto.

Con queste premesse, i documenti di accompagnamento al progetto devono essere organizzati sulla base della successione di fasi che compongono il progetto stesso:

1. Raccolta informale dei requisiti, comprensiva di tutti i requisiti funzionali e non
2. Stesura del Glossario
3. Stesura dell'analisi funzionale con i casi d'uso
4. Definizione della cronologia delle interazioni (activity diagram), design delle interfacce utente e diagramma di navigazione
5. Definizione delle entità, delle relazioni che tra esse intercorrono e formalizzazione con il class diagram di analisi e la sua descrizione
6. Design della base di dati e sua costruzione (e generazione degli script di creazione tabelle)
7. Design architetturale del sistema
8. Progettazione software con la definizione delle classi e l'interazione
9. Pianificazione delle attività ed assegnamento degli incarichi (solo per chi lavora in gruppo)
10. Sviluppo del software, seguendo le regole viste a lezione
11. Debug del software stendendo le note sui problemi incontrati
12. Definizione delle procedure per l'installazione del software stesso.

Pertanto il contenuto della relazione del progetto dovrà comprendere:

a. i documenti delle sezioni corrispondenti alle fasi da 1 a 9;
b. il software sviluppato, pacchettizzato e adeguatamente strutturato e commentato;
c. gli script di creazione della base di dati e/o di popolamento della base dati stessa con adeguati dati;
d. le note sui problemi incontrati
e. le istruzioni di installazione del software stesso.

I documenti dovranno essere strutturati come file PDF o sorgenti MS Word o OpenOffice Writer (si consiglia sempre di includere comunque anche il PDF). I diagrammi UML dovranno essere inclusi anche in forma di sorgenti nel formato dello strumento CASE utilizzato. E' anche possibile costruire un unico documento suddiviso in capitoli.

I sorgenti software dovranno essere organizzati come progetti o insiemi di progetti dell'ambiente integrato di sviluppo utilizzato (es. NetBeans, Eclipse o VisualStudio).

Gli script del database utilizzato dovranno essere inclusi, comprensivi dei dati necessari ad assicurare il funzionamento normale dell'applicativo. Si raccomanda, per quanto possibile, l'uso di Postgres 8.1.x come DBMS.

Le istruzioni di installazione dovranno essere sufficienti a realizzare l'installazione completa e funzionante, assumendo che nel sistema target siano già presenti i software infrastrutturali necessari al funzionamento dell'applicativo (es. macchina virtuale Java o .NET, DBMS).

Il progetto dovrà essere consegnato in formato elettronico entro e non oltre la data di chiusura delle iscrizioni all'appello, rilevabile dal sito del corso.

L’esame orale sarà diviso in due parti: la prima relativa alla teoria ed all’applicazione della teoria a casi reali, e la seconda alla discussione del progetto, entro la quale potranno essere chieste le giustificazioni teoriche delle scelte compiute nella realizzazione del progetto stesso.
Le piattaforme di realizzazione accettate per il progetto sono Java 1.5.x e .NET 2.0.
In entrambi gli ambienti dovranno essere seguite le direttive di modularità e stratificazione del software basate sull’uso dei pattern come l’MVC studiati nella teoria e viste negli esempi presentati a lezione.
In quanto segue sono definiti i dettagli di ciascuna sezione e dell’attività ad essa associata, comprensivi anche di alcuni esempi, tratti dai migliori esami degli anni scorsi.

## *1. Prima raccolta “informale” dei requisiti*

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

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

**ESEMPIO DI DOCUMENTO DI ANALISI (Autore: Fabio Bettinazzi)**
**Cliente :** Cooperativa GruppoDue SNC con sede in via Gavardina di Sotto 130, 25100 Ponte S.Marco(BS).

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

***Obbiettivo:*** Il sistema che la cooperativa ha richiesto e' un sistema di gestione informatico WEB per l'invio e la ricezione degli ordini.
Che in particolare permetta da ogni utente del servizio(i soci) di accedere al sistema, previa autenticazione, e gli permetta di controllare i prodotti che il magazzino ha a disposizione o che, in caso di momentanea indisponibilità, è in grado di avere nell'arco di pochi giorni. Inoltre al sistema è richiesta anche una funzione di amministrazione più generale del magazzino, cioè è richiesta la possibilità di gestire le anagrafiche dei clienti, dei fornitori e di ogni prodotto presente in

magazzino. In particolare l'amministratore del sistema deve poter controllare automaticamente quali prodotti presenti in magazzino sono esauriti o quali presentano una quantità minore di quella minima che deve essere sempre presente in magazzino, per evitare lunghe attese al socio che l'ha richiesta.
La gestione dell'ordine del cliente, da parte dell'amministratore deve essere eseguita nella seguente procedura: l'amministratore quando decide di gestire l'ordine deve poter marcare l'ordine in modo speciale e trasparente, poiché' anche il cliente deve avere la possibilità di controllare se un suo ordine è già stato gestito, e se quindi entro pochi giorni gli arriveranno a domicilio i prodotti richiesti.
Inoltre la segnatura dell'ordine deve portare ad un automatico aggiornamento delle quantità di magazzino, dei prodotti presenti nell'ordine del cliente, in modo da poter controllare in tempo reale le quantità di ogni prodotto presenti in magazzino, e quindi avere un risparmio di tempo e lavoro.
Inoltre il cliente(GruppoDue) richiede che sia implementata la parte delle anagrafiche anche come applicazione a maschere tradizionale e non solo come applicativo WEB, poiché le segretarie sono più abituate ad interagire tramite un'interfaccia tradizionale rispetto all'ambiente WEB.

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

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

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

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

della cooperativa GruppoDue.

## *2. Stesura del Glossario*

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

**ESEMPI DI GLOSSARIO**

MAGAZZINO (Autore: Fabio Bettinazzi)

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

RICETTARIO WEB (Autore: Davide Masi)

***Ricetta***

Insieme di informazioni necessarie per la preparazione di un piatto.
Una ricetta e' caratterizzata da:

1. **Nome**;
2. Numero di **porzioni**;
3. **Attributi di classificazione**;
4. Lista di **ingredienti**;
5. Lista di **passi per la preparazione**;
6. **Tempo di preparazione**;
7. **Voto** medio degli utenti: solo se sono presenti commenti per la ricetta;
8. Lista dei **commenti** degli utenti;

***Porzioni***

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

***Ingrediente***

Componente atomico di una ricetta; caratterizzato da:

1. **Nome**;
2. **Unità di misura**: relativa alla specifica ricetta in cui l'ingrediente compare;

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

***Unità di misura***

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

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

***Preparazione***

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

***Tempo necessario alla preparazione***

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

***Commento***

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

***Utente***

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

**<u>Attributi di classificazione</u>**

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

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

Ogni ricetta ha una e una sola classificazione.

***Portata***

Assume i valori:

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

***Costo Stimato***

1. Indicazione approssimata del costo di una ricetta.

Assume i valori:

- basso
- medio
- alto
- molto alto

***Alimento Base***

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

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

## *3. Analisi funzionale con i casi d'uso*

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

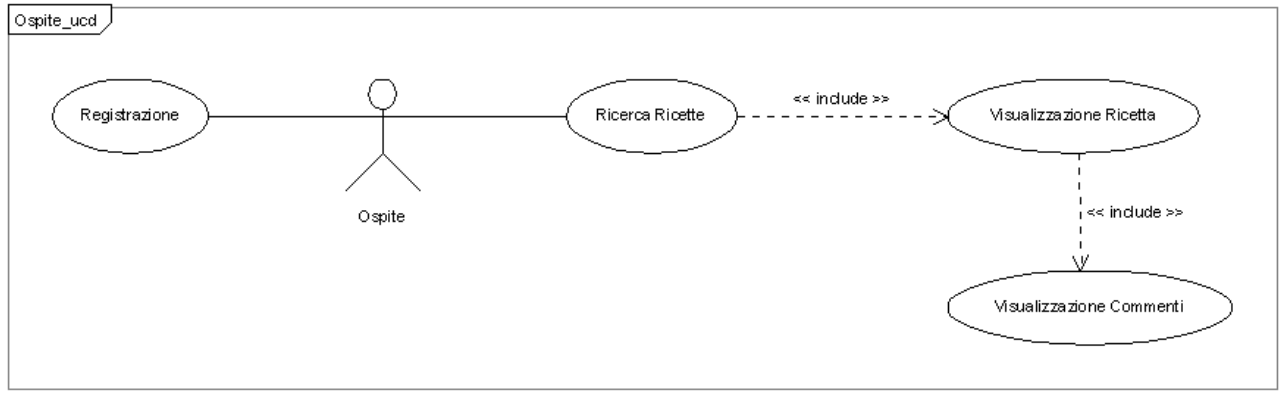
1. Definizione esatta del boundary o confine del sistema (entro sistemi particolarmente complessi questa fase può anche essere applicata a sotto-sistemi);
2. Identificazione e definizione degli attori, ossia delle entità esterne con cui il sistema (o i sotto-sistemi) oggetto dell'analisi interagiscono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

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

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

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

## ESEMPIO DI USE CASE CON DESCRIZIONE




**Visualizzazione ricetta (Autore: Davide Masi)**

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

**Descrizione del procedimento**

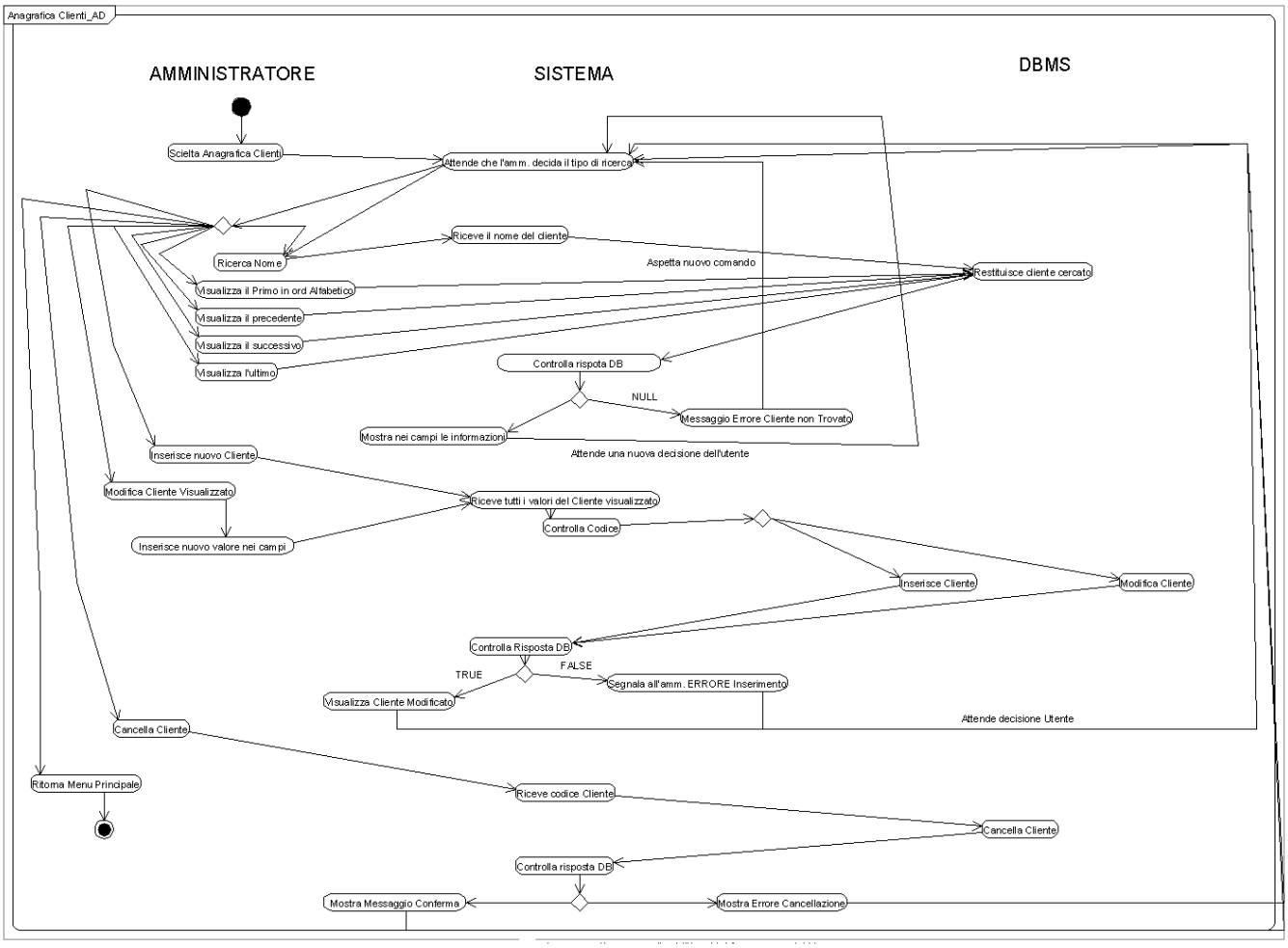
1. Il **Sistema** mostra gli attributi della ricetta da visualizzare. Gli attributi sono elencati secondo il seguente ordine:
    1. Nome.
    2. Classificazione della ricetta: tipo di portata e alimento caratterizzante, costo.
    3. Tempo totale di preparazione.
    4. Ingredienti: sono presentati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

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

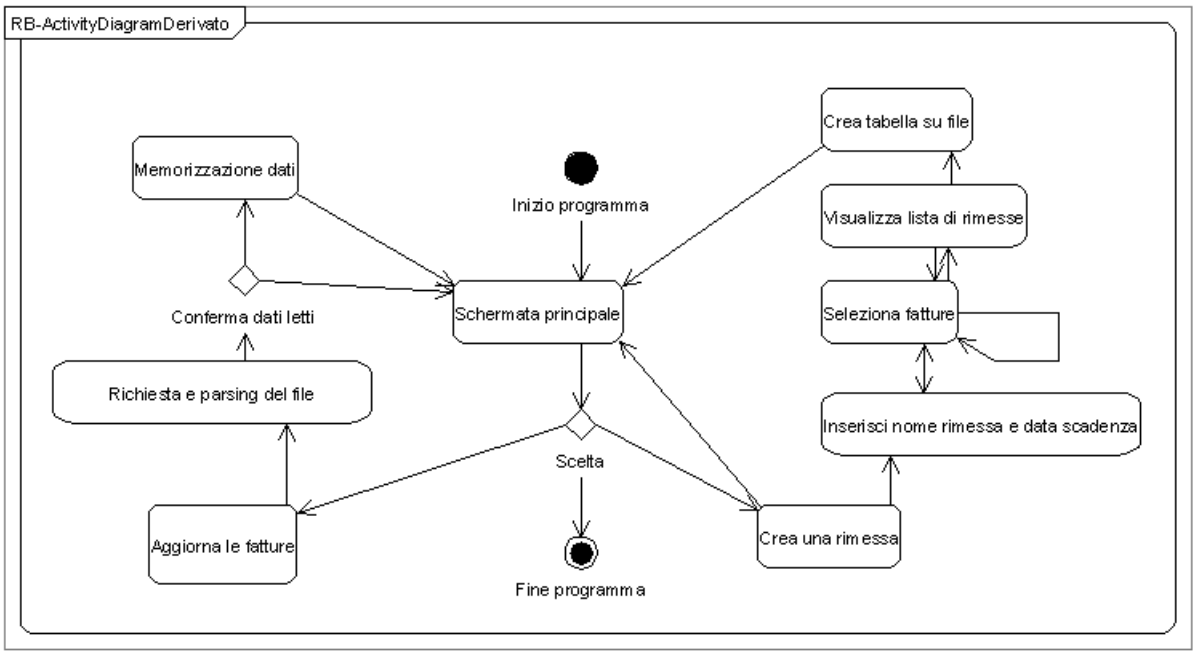
## *4. Stesura del Diagramma di navigazione*

L'insieme dei casi d'uso, pur esprimendo perfettamente l'insieme delle interazioni tra utente e sistema ed i legami logici che fra esse intercorrono, manca dell'aspetto cronologico della navigazione fra le finestre od interfacce, per cui, attraverso l'uso di un activity diagram, occorre dare un legame anche cronologico alla successione delle interazioni.
Non è detto che le singole attività abbiano sempre una corrispondenza uno a uno con i singoli casi d'uso: in alcuni casi è opportuno suddividere un singolo use case in più di un'attività o, viceversa, accorpare in una sola attività più di un caso d'uso.
In ogni caso il risultato di questa fase viene essere espresso attraverso un activity diagram che definisca la successione cronologica di attività, ciascuna delle quali corrisponde ad un tipo particolare di interazione tra gli attori ed il sistema.
Da questo deriva il diagramma di navigazione, entro il quale da ciascuna attività è derivata una interfaccia utente (maschera Form o Web) e che definisce con precisione la navigazione fra le maschere utente.
I diagrammi vanno ovviamente corredati di descrizione di ciascuna delle interfacce utente.

**ESEMPIO DI ACTIVITY DIAGRAM (Autore: Fabio Bettinazzi)**



**ESEMPIO DI ACTIVITY DIAGRAM E CONNESSO DIAGRAMMA DI NAVIGAZIONE (Autore: Alessandro Vincenzi)**




Spiegazione relativa ai diagrammi

**Schermata Principale**

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

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

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

**Aggiornamento delle fatture**

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

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

In seguito alla selezione del file si deve eseguire il parsing dello stesso, modo da leggere tutti i dati presenti.

Il parsing si esegue seguendo il modello di fattura e leggendo una fattura alla volta (riga per riga). Man mano che i dati si leggono, vanno salvati in oggetti del tipo corrispondente (fattura, cliente ...) e al termine della singola fattura si caricano tutti gli oggetti in un vettore. Nel caso si verifichino errori che compromettano la lettura del file, il processo di parsing viene arrestato e si comunica l'errore all'utente. Nel caso sulla fattura non si trovino i dati aspettati, tale informazione deve essere salvata per una comunicazione successiva.

Al termine della lettura del file, si deve visualizzare una finestra di riassunto dei dati letti (solo il numero di fattura) e dei dati mancanti. Sempre con questa finestra si chiude all'utente se procedere

con la memorizzazione dei dati o con l'annullamento dell'operazione. In quest'ultimo caso i dati vanno persi e si rientra alla schermata principale. Se l'utente sceglie di procedere con la

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

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

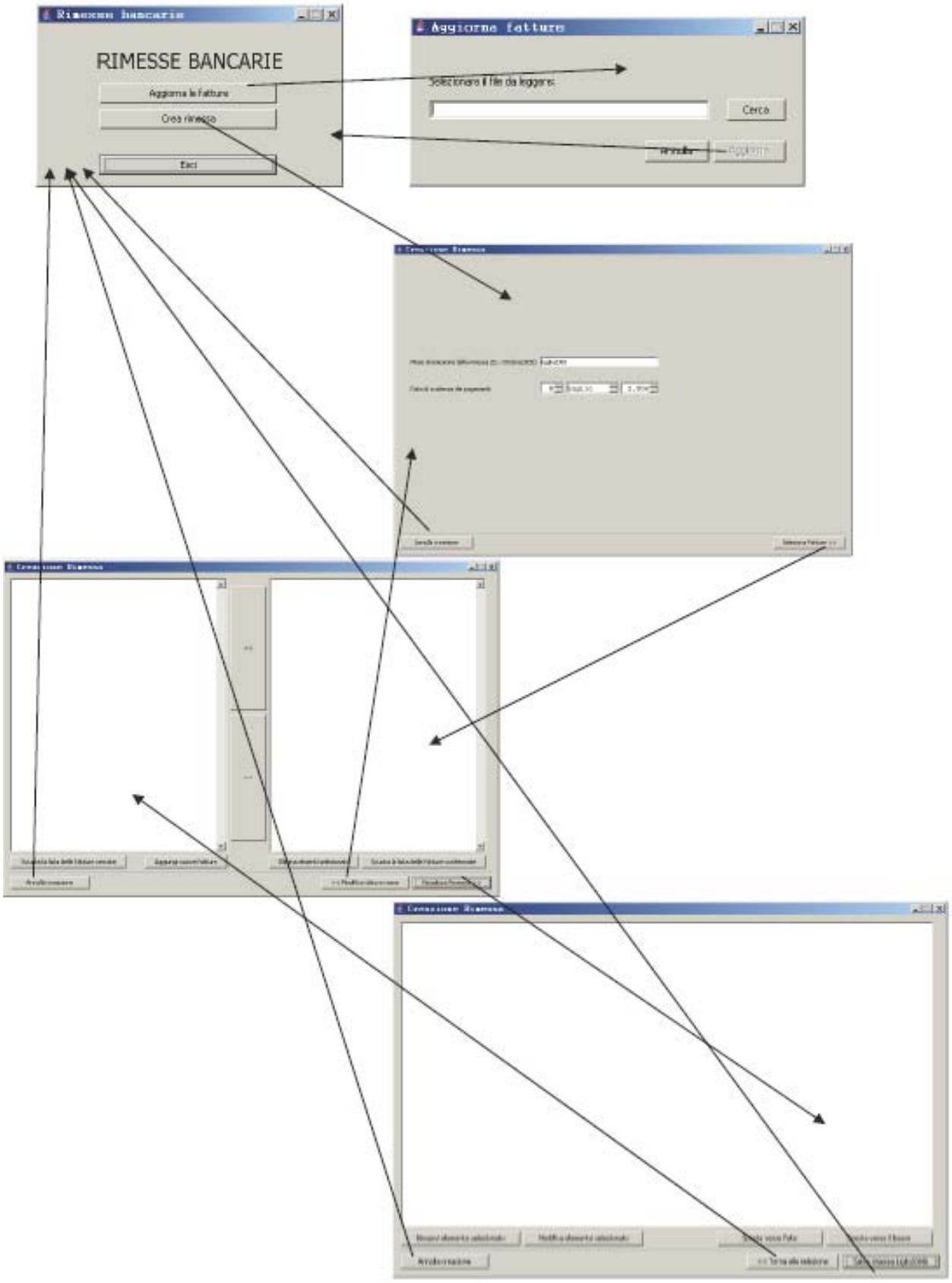
in modo da garantire i vincoli di chiave esterna tra le fatture. Se durante tale operazione non si verificano errori da parte del database, allora si esegue il commit delle operazioni, altrimenti si esegue il rollback e si comunica l'insuccesso all'utente.
In ogni caso si rientra nella schermata principale.

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

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

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

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


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

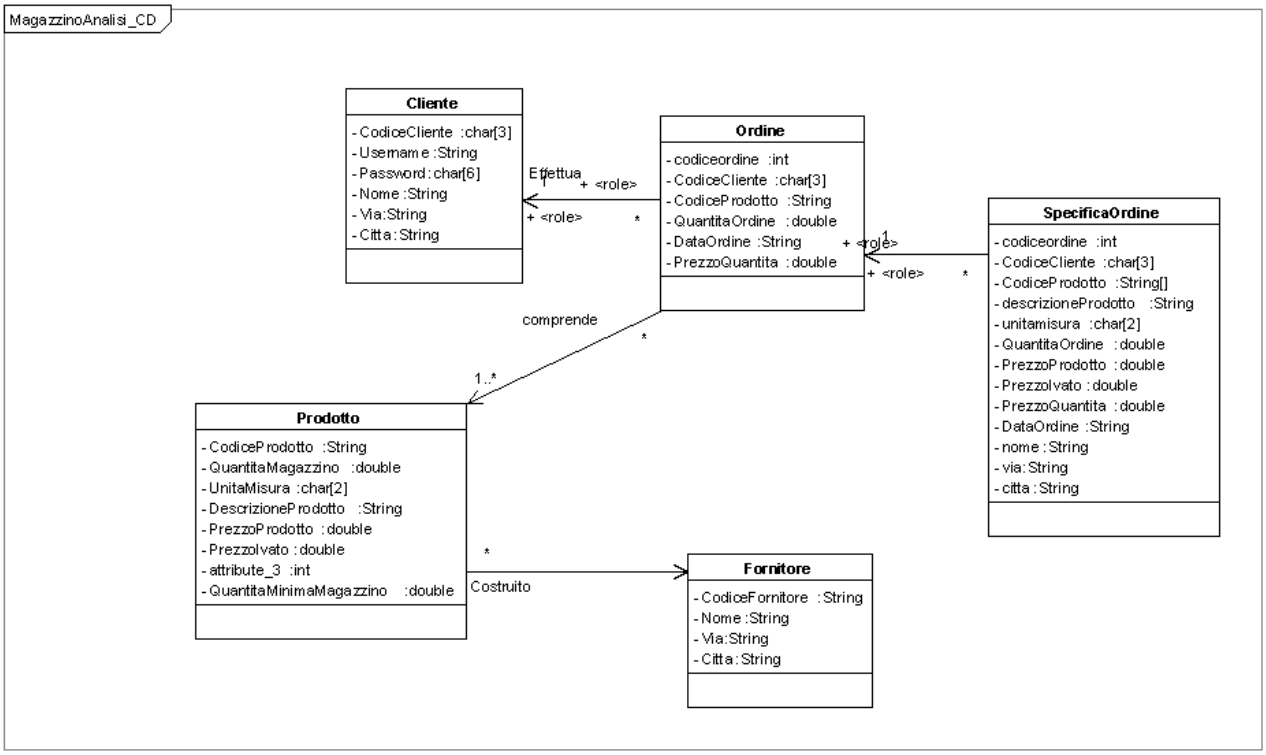
## *5. Definizione delle entità attraverso il Class Diagram di Analisi*

In questa fase, che parte dal Glossario realizzato in fase 2 e dallo/dagli Use Case Diagram (corredati anche degli Activity Diagram) realizzati in fase 3, deve essere realizzato il diagramma delle classi di analisi. Tale diagramma deve indicare chiaramente tutte le classi entità, ossia le classi definibili come "proiezioni" nel dominio della applicazione software delle entità del dominio del problema dove la applicazione software andrà ad operare, più eventuali altre classi individuate nel corso dell'analisi che siano di importanza per i concetti funzionali che definiscono i requisiti del progetto. In pratica nel diagramma, che è l'equivalente da un punto di vista del ruolo (e l'evoluzione da un punto di vista storico e metodologico) del diagramma Entità-Relazioni (ER) usato nelle metodologie di sviluppo più tradizionali, devono essere chiaramente indicate:

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

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

## ESEMPIO DI CLASS DIAGRAM DI ANALISI (Autore: Fabio Bettinazzi)



## *6. Design della base di dati*

In questa fase si applicano le regole dell'esame di Basi di Dati per giungere alla costruzione di una base dati completa per il sistema software. L'obiettivo deve essere la costruzione degli script per la generazione della base dati, che dovranno essere salvati a parte.

## *7. Scelta architetturale e definizioni conseguenti*

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

## *8. Progettazione software e Class Diagram di Progetto*

In questa fase occorre definire chiaramente tutte le classi che fanno parte dell'applicazione software da implementare. Il Class Diagram di Progetto è l'elenco completo delle classi, con tutte le loro relazioni e su di esso si basa anche il dimensionamento della fase di sviluppo (ovvero scrittura vera e propria del software).

Il processo che permette di giungere al diagramma delle classi di progetto è necessariamente iterativo. Si parte dal diagramma delle classi di analisi e devono essere inserite tutte le classi di servizio, ossia le classi infrastrutturali, non necessariamente derivate dalla fase di analisi, che permettono al programma nel suo insieme di operare correttamente ed in modo efficiente. Le classi di servizio sono ovviamente fortemente dipendenti nella loro struttura dall'architettura scelta e da eventuali framework utilizzati nel progetto. Se un diagramma di analisi ben fatto può essere spesso utilizzato con diverse tecnologie ad oggetti, ovvero essere punto di partenza per progetti analoghi realizzati su piattaforme diverse, un diagramma di progetto è chiaramente molto più influenzato dalla tecnologia usata. Il processo usa anche altri diagrammi UML.

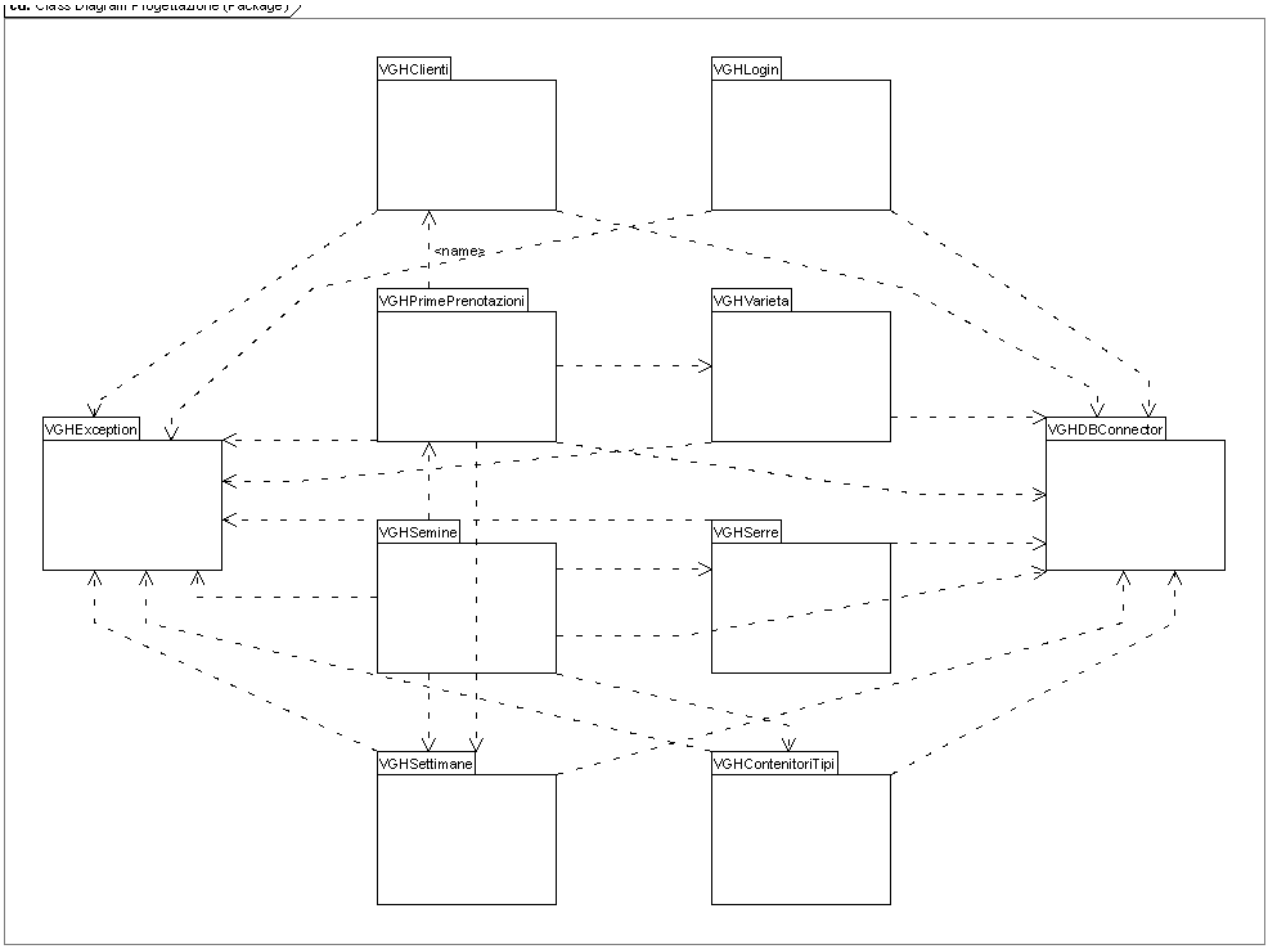
1. I diagrammi di interazione (sequence, che pone enfasi sulla sequenza temporale delle interazioni, e collaboration, che pone enfasi sulla dipendenza fra le classi) sono di importanza fondamentale sia per la definizione dei metodi che le classi offrono le une alle altre (e dei loro argomenti e valori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

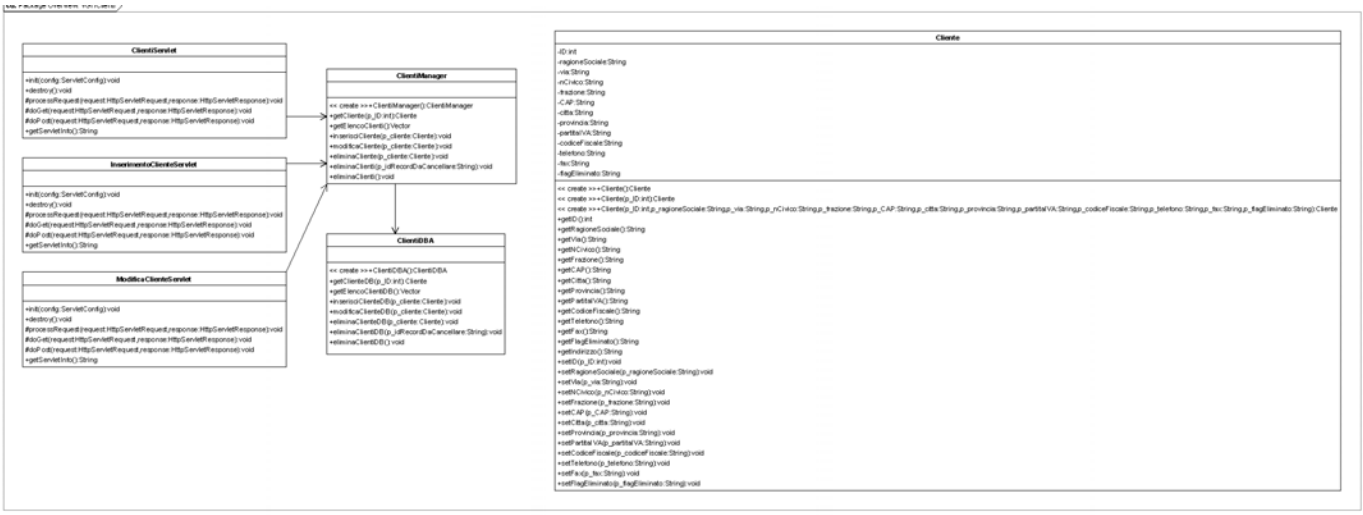
L'obiettivo finale è comunque la realizzazione del Class Diagram di Progetto, completo di tutte le classi. Spesso per motivi di chiarezza (specialmente in progetti grandi dove le classi sono molto numerose) il diagramma viene diviso in package, associazioni di classi corrispondenti ad unità funzionali,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

Dal punto di vista didattico, il prodotto di questa fase deve essere il class diagram di progetto, corredato dalla descrizione di tutte le classi coinvolte. Qualora le classi siano

molte conviene applicare una struttura a due livelli, suddividendo il progetto in package, come nell'esempio sotto riportato

## ESEMPIO DI CLASS DIAGRAM DI PROGETTO (Gestione Vivaio di Irene Bacchi)



## *9. Pianificazione delle attività*

Questa parte è riservata solo a chi lavora in gruppo e deve contenere le seguenti componenti:

1. Suddivisione del progetto in attività
2. Definizione degli incarichi comprendenti le attività
3. Assegnazione degli incarichi ai componenti del progetto
4. Ipotesi di successione delle attività e di computo delle tempistiche associate.

## *10. Altri documenti di contorno*

Usando i diagrammi realizzati in precedenza si arriva a definire le parti implementative di contorno del progetto, che devono essere opportunamente documentate come segue.

1. Note sui problemi riscontrati che indicano chiaramente indicati eventuali limiti e/o malfunzionamenti della/e piattaforma software ed hardware utilizzata e le soluzioni trovate per ovviare ad essi;
2. Istruzioni di installazione, che spieghino chiaramente cosa occorre fare per installare l'applicativo su una piattaforma scelta;
3. Eventuali altri documenti che si ritengano necessari.

**ESEMPIO DI NOTE DI INSTALLAZIONE (Autore: Andrea Cimino)**

Per installare ed eseguire questo software in modo corretto è necessario
avere:

1. Una Java Virtual Machine.
2. MySQL server, configurato in maniera appropriata.
3. TomCat Server per la gestione della parte Web.

In particolare il punto chiave dell'installazione è l'accesso al DB. E' presente all'interno del pacchetto il file (configurazioneMySQL.sql) da importare su MySQL server tramite l'utilità MySQL-admin, questo per creare automaticamente un database valido. Questo avrà nome: "infowebshop". Bisogna prestare particolare attenzione al fatto che, per il corretto funzionamento del programma, deve esistere un utente specifico sul DB che deve aver accesso completo alle tabelle di "infowebshop".
Le caratteristiche di questo utente sono:

- username: pippo
- password: pippo
- accesso completo (lettura/scrittura) alle tabelle

Per creare ulteriori account per l'accesso al sistema da "amministratore", basterà creare questi utenti (anche con accesso in sola lettura) sul DB "infowebshop" gestito da MySQL server.
N.B.: la scelta di username e password sono state fatte ad uso esclusivamente didattico.