Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Gestionale
Corso di Progettazione del Software
Proff. Toni Mancini e Monica Scannapieco

Progetto **PC.20080418**

versione del 22 aprile 2008

Si vuole progettare e realizzare *OptTraining*, un sistema informatico che gestisce corsi di formazione interni ad una certa azienda. Il sistema deve permettere l'organizzazione di corsi con docenti interni all'azienda, impartiti a personale anch'esso interno, e la valutazione sia dell'efficacia del processo di "knowledge transfer", sia della partecipazione effettiva ai corsi.

Si richiede di effettuare le fasi di Analisi, Progetto, e Realizzazione del sistema in JAVA, utilizzando la metodologia illustrata nel corso.

Requisiti

Il sistema *OptTraining* deve memorizzare nome, cognome e matricola dei partecipanti ai diversi corsi. Ogni corso (di cui deve essere memorizzato il nome) prevede tipicamente più di una edizione. È necessario tenere traccia sia delle iscrizioni alle edizioni che della partecipazione effettiva alle stesse. Un'edizione (di cui interessa il nome e le date di inizio e fine) è organizzata in più giornate. Di queste ultime interessa il numero progressivo e la data.

Un partecipante è identificato come "partecipante attivo" se ha preso parte ad almeno i 2/3 delle giornate complessive di un corso.

I docenti, di cui interessa nome, cognome e matricola, si distinguono in docenti di prima e di seconda fascia. *OptTraining* deve memorizzare le ore di docenza richieste da ciascun docente per le diverse edizioni dei corsi. Inoltre, il sistema deve permettere di calcolare il compenso che spetta ai docenti per ciascuna edizione. Tale funzionalità verrà utilizzata dal sistema esterno deputato ai pagamenti. La retribuzione di un docente è data dal prodotto tra il numero di ore di lezione effettuate e la sua tariffa oraria. Quest'ultima dipende dalla sua fascia di appartenenza: in particolare, si assuma di avere a disposizione uno use-case `Tariffe` con le operazioni `tariffaOrariaPrimaFascia():reale>0` e `tariffaOrariaSecondaFascia():reale>0`.[1]

I partecipanti ai corsi sono soggetti ad una valutazione del loro rendimento per ciascuna giornata formativa alla quale hanno partecipato. In particolare, la valutazione del rendimento,

[1] Si ignori la specifica di tale use-case, ma lo si utilizzi opportunamente nei diversi diagrammi e specifiche.

effettuata dal docente, è in forma booleana (positivo/negativo). Nessun rendimento viene ovviamente espresso per i partecipanti assenti ad una giornata.

Il responsabile della divisione "Formazione interna" dell'azienda che ha commissionato l'applicazione deve valutare un'edizione di un corso da un punto di vista complessivo. Tale valutazione avviene secondo due fattori. In primo luogo, va considerato il rendimento dei partecipanti. Tale rendimento è "positivo" se la maggioranza dei "partecipanti attivi" ha avuto un rendimento positivo in almeno l'85% delle giornate alle quali hanno partecipato, negativo altrimenti. In secondo luogo, si vuole considerare l'assenteismo degli iscritti. La valutazione dell'assenteismo in una edizione di un corso è considerata bassa se i partecipanti attivi sono almeno l'80% degli iscritti, alto altrimenti. La valutazione complessiva dell'edizione è quindi espressa secondo 3 valori:

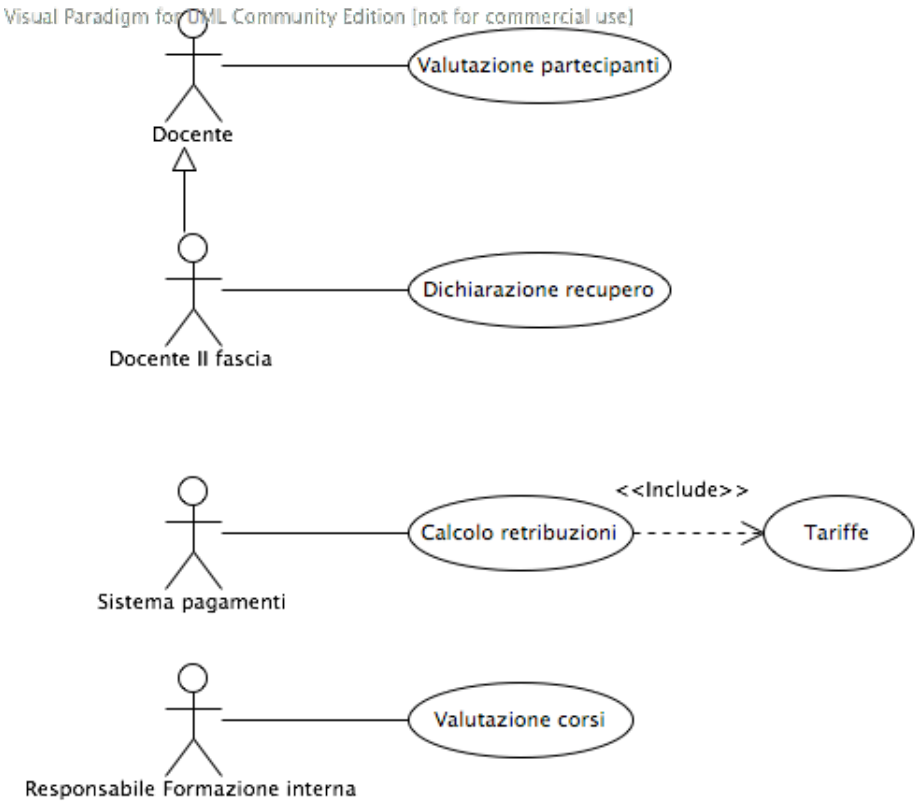
- "ottima" in caso di alto rendimento e basso assenteismo;
- "scarsa" in caso di basso rendimento e alto assenteismo;
- "media" altrimenti.

Per motivi contrattuali, i docenti di seconda fascia devono effettuare un recupero delle ore lavorative in cui hanno effettuato delle docenze: in altre parole, devono lavorare un numero di ore aggiuntive (svolgendo le mansioni per le quali sono normalmente pagati) per poter compensare il tempo impiegato nella docenza. Questa regolamentazione è relativa al fatto che non è espressamente parte del loro contratto effettuare docenze.

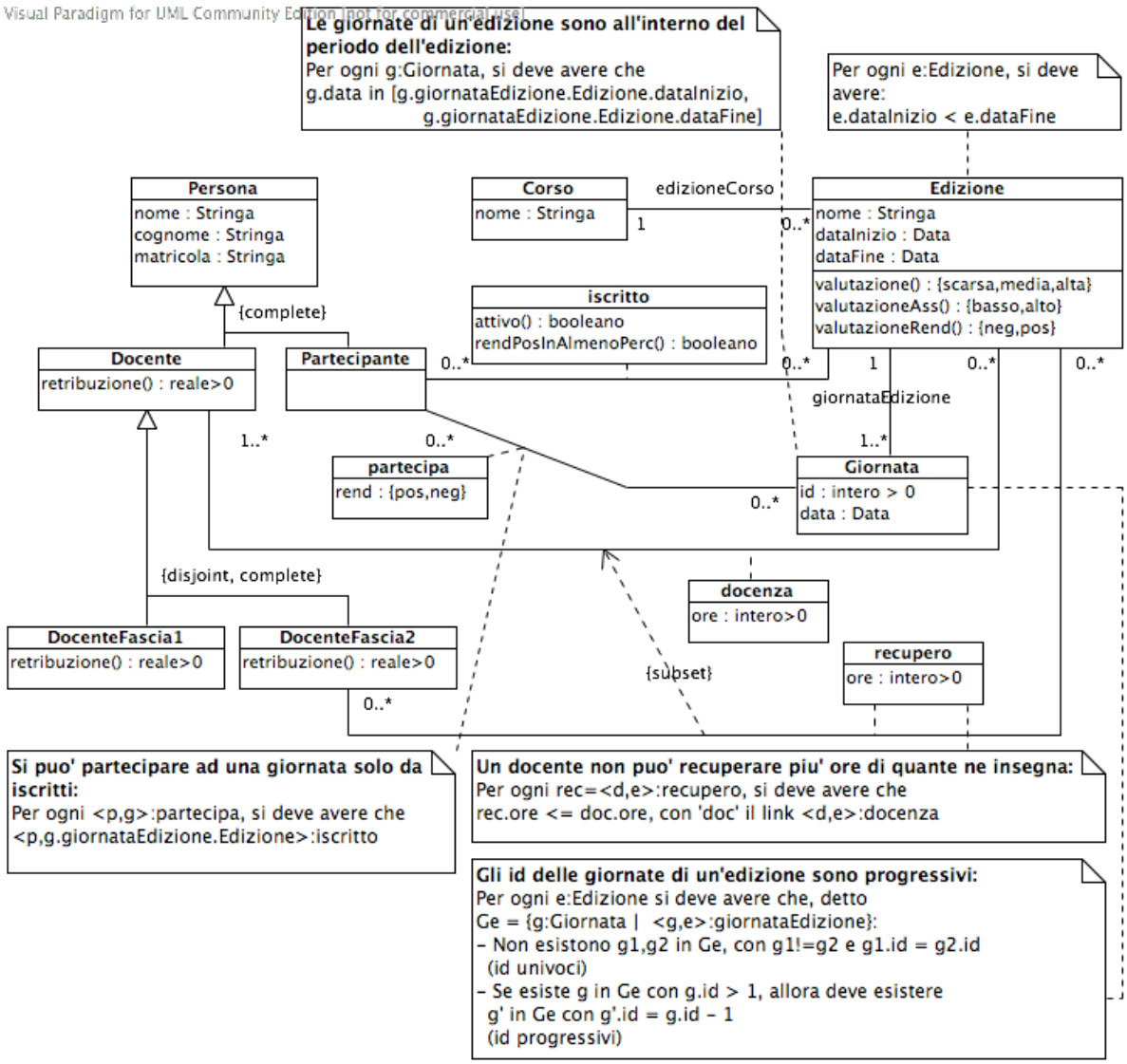
La procedura prevede che un docente di seconda fascia possa dichiarare al sistema il numero di ore in eccesso che intende lavorare per recuperare le sue ore di docenza in una data edizione di un corso. Il costo orario delle ore di docenza recuperate resta invariato, mentre quello delle ore non recuperate viene decurtato del 75%. Il meccanismo di calcolo del compenso di un docente deve tener conto delle ore recuperate.

1 Fase di Analisi

1.1 Diagramma degli Use Case



1.2 Diagramma delle classi UML



1.3 Specifica dei tipi di dato

Nessun tipo di dato definito

1.4 Specifica degli use case

```
SpecificaUseCase ValutazionePartecipanti
  valutaPartecipante(d:Docente, p:Partecipante, g:Giornata, rend:{pos,neg})
    pre: <d,g.giornataEdizione.Edizione>:docenza,
         <p,g.giornataEdizione.Edizione>:iscritto
    post:
       Se non esiste il link l=<p,g>:partecipa, questo viene creato.
       l.rend = rend
FineSpecifica

SpecificaUseCase DichiarazioneRecupero
  dichiaraRecupero(d:DocenteFascia2, e:Edizione, ore:intero>0)
    pre: doc=<d,e>:docenza e doc.ore >= ore
    post:
        Se non esiste il link rec=<d,e>:recupero, questo viene creato.
        rec.ore = ore.
FineSpecifica

SpecificaUseCase CalcoloRetribuzioni
  calcolaRetribuzione(d:Docente, e:Edizione):reale>0
    pre: Le precondizioni di d.retribuzione(e) sono rispettate
    post: result = d.retribuzione(e)
FineSpecifica

SpecificaUseCase ValutazioneCorsi
  valutaEdizione(e:Edizione): {scarsa,media,alta}
    pre: nessuna
    post: result = e.valutazione()
```

1.5 Specifica delle classi e diagrammi degli stati e transizioni

La classe Edizione

```
SpecificaClasse Edizione
  valutazione(): {scarsa,media,alta}
    pre: nessuna
    post:
      Se this.valutazioneRend() = pos e this.valutazioneAss() = basso
          allora result = alta
```

```
      altrimenti se this.valutazioneRend() = neg e this.valutazioneAss() = alto
          allora result = bassa
      altrimenti result = media.

  valutazioneRend(): {pos, neg}
    pre: nessuna
    post:
       Detto P = { p:Partecipante  |  <p,this>:iscritto e <p,this>.attivo() = true }
       l'insieme dei partecipanti attivi dell'edizione this,

       sia Q = { p in P  |   <p,this>.rendPosInAlmenoPerc(0.85) = true }
       l'insieme di quelli che hanno avuto rendimento positivo in almeno l'85%
       delle relative giornate alle quali hanno partecipato.

       result = pos se |Q|/|P| > 0.50,  result = neg altrimenti.

  valutazioneAss(): {basso, alto}
    pre: nessuna
    post:
       Detti
           P = { p:Partecipante  |  <p,this>:iscritto }
       e
           A = { p:P  |  <p,this>.attivo() = true }
       result = basso se |A|/|P| >= 0.80,  result = alto altrimenti.
FineSpecifica
```

La classe Docente e le sue derivate

```
SpecificaClasse Docente
  retribuzione(e:Edizione): reale > 0
    pre: <this,e>:docenza
    post: result dipende dalla classe piu' specifica a cui this appartiene
FineSpecifica

SpecificaClasse DocenteFascia1
  retribuzione(e:Edizione): reale > 0
    pre: <this,e>:docenza
    post: result = <this,e>.ore * Tariffe.tariffaOrariaPrimaFascia()
FineSpecifica
```

```
SpecificaClasse DocenteFascia2
  retribuzione(): reale > 0
    pre: <this,e>:docenza
    post:
       Sia oreRecuperate = rec.ore, se esiste il link rec=<this,e>:recupero,
                           0 altrimenti.
       Detto doc=<this,e>:docenza

       result = oreRecuperate * Tariffe.tariffaOrariaSecondaFascia()   +
                   (doc.ore-oreRecuperate) * Tariffe.tariffaOrariaPrimaFascia()*0.25
FineSpecifica
```

Le operazioni dell'associazione Iscritto

Nota: Sebbene non esplicitamente previste nella metodologia di analisi illustrata nel corso, anche le associazioni possono avere operazioni. Di queste va quindi prevista una specifica concettuale in modo analogo a quanto avviene per le classi.

In questo esempio, collocare le operazioni `attivo()` e `rendPosInAlmenoPerc()` nell'associazione *iscritto* è particolarmente conveniente, visto che l'input richiesto è una coppia `p:Partecipante, e:Edizione` tale che `<p,e>:iscritto`. L'istanza di invocazione `this` di queste operazioni sarà quindi un link esistente dell'associazione *iscritto*.

Ovviamente collocazioni alternative di queste operazioni sarebbero valide, seppure con input e precondizioni diversi.

```
SpecificaAssociazione iscritto
  attivo(): booleano
    pre: nessuna
    post:
       Detti
              G = {g:Giornata  |  <this.Edizione, g>:giornataEdizione}
       l'insieme delle giornate dell'edizione del corso coinvolta nel link this, e
              GP = {g:G  |  <this.Partecipante, g>:partecipa}
       l'insieme delle giornate in G in cui il partecipante coinvolto nel link this ha
       partecipato,
       result = true se e solo se |GP|/|P| >= 2/3
  rendPosInAlmenoPerc(perc:reale in [0..1]): booleano
    pre: nessuna
    post:
```

```
      Detto  GP = {g:Giornata  |  <this.Edizione, g>:giornataEdizione e
                                          <this.Partecipante, g>:partecipa}
      l'insieme delle giornate dell'edizione this.Edizione in cui il partecipante
      this.Partecipante ha partecipato,

      Sia GPA = {g:GP   |   <this.Partecipante, g>.rend = pos}
      l'insieme di tali giornate in cui il partecipante this.Partecipante ha avuto un
      rendimento positivo.

      result = true se e solo se |GPA|/GP| >= perc
FineSpecifica
```