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.20061201**

versione del 16 marzo 2007

Si vuole progettare e realizzare un sistema, per conto del centro di supporto tecnico on-line di una certa azienda di software, che consenta di gestire le richieste dei clienti (per lo più quesiti tecnici e di assistenza).

In particolare, si desidera che il sistema permetta ai clienti di inviare quesiti agli operatori del centro (via posta elettronica), e agli operatori di rispondere sulla base delle loro competenze.

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

Requisiti

Le entità di interesse per l'applicazione sono i clienti (con nome, cognome ed indirizzo email), gli operatori del centro di supporto, e i messaggi scambiati. I messaggi che il sistema deve mantenere sono sia quelli spediti dai clienti, sia le relative risposte degli operatori del centro (di cui interessa l'operatore che effettua la risposta). Di tutti i messaggi interessa inoltre data e ora di spedizione.

Le interazioni tra centro assistenza e clienti partono sempre da questi ultimi. In particolare, quando il sistema riceve una nuova richiesta di assistenza, assegna ad essa un codice identificativo univoco (una stringa). Tale richiesta viene successivamente presa in carico da un operatore del centro (secondo regole che sono descritte in seguito), che invia un messaggio di risposta al cliente.

Tuttavia, dato che la risposta può essere ritenuta non soddisfacente, è possibile che lo scambio di messaggi relativo ad una richiesta di assistenza continui, con il cliente che invia un ulteriore messaggio al centro, rispondendo a quello ricevuto, con il centro che si occuperà di inviare una nuova risposta, e così via. Una funzionalità importante che il sistema deve offrire è quindi il *tracciamento della sequenza dei messaggi* scambiati tra i clienti e il centro, per risolvere le singole richieste di assistenza. In particolare, si può assumere che il modulo del sistema deputato a ricevere messaggi dai clienti (di cui non è richiesta la progettazione) riesca a determinare correttamente (tramite ispezione dell'oggetto e del corpo del messaggio alla ricerca

di opportuni codici identificativi) se questo è relativo ad una nuova richiesta di assistenza, oppure è da considerarsi un passo di ulteriore interazione relativo ad una richiesta a cui si è data già risposta. In quest'ultimo caso, il sistema deve permettere di risalire a tale risposta e, via via, all'indietro, all'intera sequenza di messaggi scambiati tra cliente e centro per quella richiesta di assistenza.

Il modulo di ricezione dei messaggi si occupa anche di classificare le nuove richieste di assistenza in base all'applicativo software (prodotto dall'azienda) sul quale il cliente chiede aiuto. Tale informazione deve essere mantenuta dal sistema. Degli applicativi software interessa conoscere il nome, la descrizione, e le versioni prodotte (di cui interessa codice e data di rilascio).

La conoscenza dell'applicativo a cui una richiesta di assistenza fa riferimento è importante perché:

- I clienti possono richiedere assistenza esclusivamente per quei prodotti sw di cui hanno acquistato almeno una versione (questa informazione deve essere mantenuta dal sistema). L'applicazione ignorerà i messaggi dei clienti relativi a prodotti software che non hanno acquistato.
- Gli operatori del centro di supporto (di cui interessa nome, cognome e matricola), sono competenti solo su alcuni degli applicativi prodotti dall'azienda, per cui non possono rispondere a richieste di assistenza relative ad altri applicativi.

Infine, allo scopo di migliorare la qualità del servizio offerto, l'azienda consente ad alcuni clienti, dietro pagamento di una quota aggiuntiva per l'assistenza, di avere un qualche diritto di precedenza nell'evasione delle loro richieste (cf. seguito). Tali clienti vengono denominati *clienti gold*.

Il sistema deve consentire ai singoli operatori di scegliere, dall'elenco dei messaggi *pendenti* (ovvero a cui non si è ancora data risposta), il prossimo messaggio a cui rispondere. In particolare, dato un operatore e l'insieme dei messaggi pendenti, il sistema deve restituirne uno, scegliendolo secondo le regole seguenti:

- È relativo ad un prodotto sw acquistato dal cliente;
- È relativo ad un prodotto sw di cui l'operatore è esperto.

In caso esistano più messaggi che rispettano i suddetti vincoli, la scelta avverrà nel modo seguente (precedenza cronologica, con una certa priorità per i clienti gold):

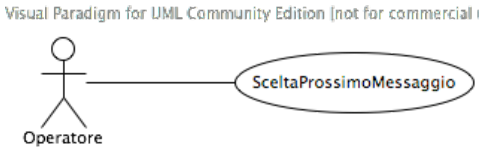
- Siano m ed mg i messaggi di clienti rispettivamente non-gold e gold ricevuti da più tempo;

- Se la data e l'ora di mg è precedente quella di m, la funzione restituirà mg (ordine cronologico);
- Se la data e l'ora di m precede quella di mg di almeno 36 ore, la funzione restituirà m (ordine cronologico);
- Altrimenti (m precede mg di meno di 36 ore) la funzione restituirà mg (priorità dei clienti gold).

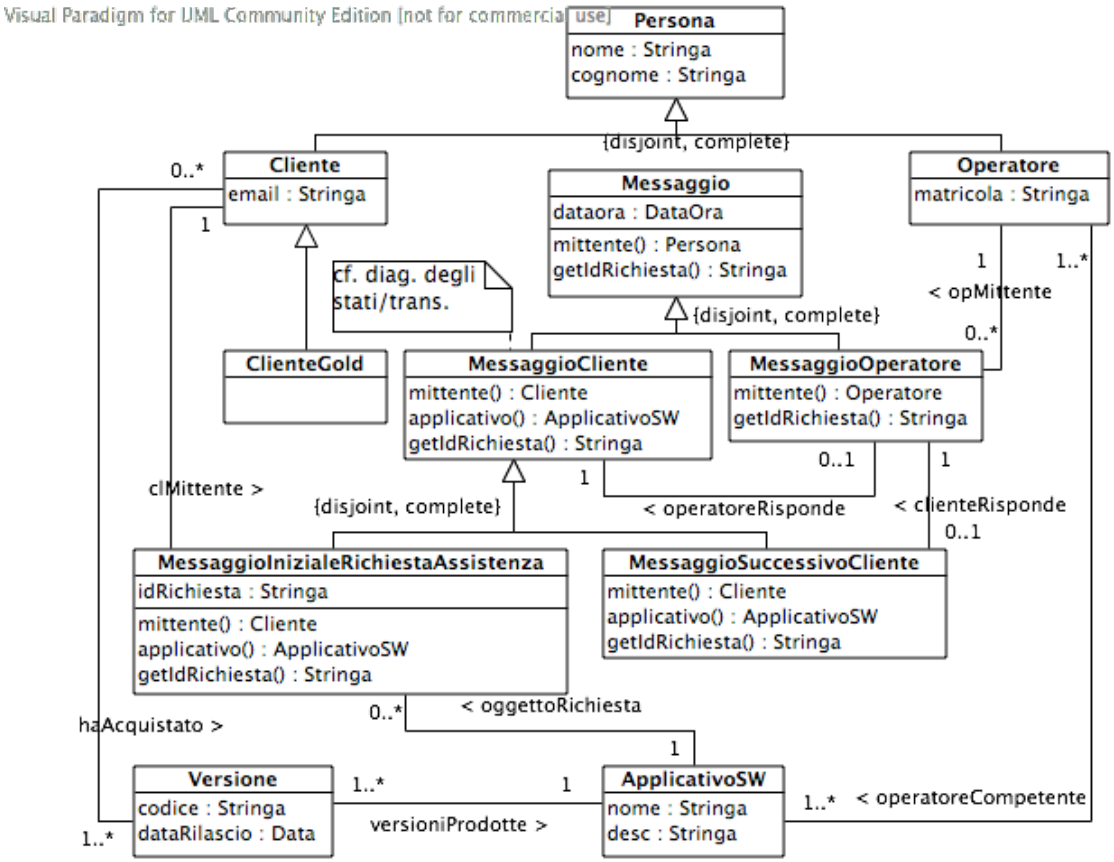
L'idea è quindi quella di favorire i clienti gold, a patto che non esistano messaggi precedenti di clienti non-gold in attesa da più di 36 ore.

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 presente.

1.4 Specifica degli use case

```
SpecificaUseCase SceltaProssimoMessaggio
  getMessaggio(o:Operatore, I:Insieme(MessaggioCliente)):MessaggioCliente
    pre: I contiene almeno un messaggio 'm' nello stato 'pendente'.
```

Nota: Non c'e' bisogno di richiedere che tutti i messaggi in I siano pendenti,
dato che l'operazione puo' semplicemente ignorare quelli ai quali si e' data
gia' risposta (cf. definizione di I' nelle postcondizioni).
E' pero' importante che esista in I almeno un messaggio pendente, altrimenti
il valore di ritorno dell'operazione non sarebbe definito.

post: Sia I' il sottoinsieme di I definito come segue:

$$\mathtt{I'} = \left\{ \mathtt{m} \in \mathtt{I} \;\middle|\; \begin{array}{l} \exists \mathtt{v} \in \mathtt{Versione\ t.c.}\ \ \langle \mathtt{m.mittente()}, \mathtt{v} \rangle \in \mathtt{haAcquistato\ e} \\ \qquad\qquad\qquad\qquad\ \ \langle \mathtt{v}, \mathtt{m.applicativo()} \rangle \in \mathtt{versioniProdotte\ e} \\ \langle \mathtt{o}, \mathtt{m.applicativo()} \rangle \in \mathtt{operatoreCompetente\ e} \\ \mathtt{m\ e'\ nello\ stato\ `pendente'} \end{array} \right\}$$

ovvero l'insieme dei messaggi pendenti in 'I' relativi ad applicativi di cui
i mittenti hanno acquistato almeno una versione
e di cui l'operatore 'o' e' esperto.

Siano ora M e MG i sottoinsiemi di I' dei messaggi i cui mittenti sono
clienti non-gold e gold, rispettivamente:

$$\mathtt{M} = \{\mathtt{m} \in \mathtt{I'} \mid \mathtt{m.mittente()} \notin \mathtt{ClienteGold}\};$$

$$\mathtt{MG} = \{\mathtt{m} \in \mathtt{I'} \mid \mathtt{m.mittente()} \in \mathtt{ClienteGold}\}.$$

Nel caso in cui sia M che MG sono non-vuoti, detti m e mg i messaggi meno recenti
in --rispettivamente-- M e MG, ovvero tali che:

$$\forall \mathtt{m'} \in \mathtt{M} \quad \mathtt{m.dataora} \leq \mathtt{m'.dataora}$$

$$\forall \mathtt{mg'} \in \mathtt{MG} \quad \mathtt{mg.dataora} \leq \mathtt{mg'.dataora}$$

result sara' pari a:
- mg, se mg.dataora < m.dataora + 36 ore;
- m, altrimenti.

In caso invece M (rispettivamente MG) sia vuoto, result sara' invece pari
a mg (risp. m), come definiti nel caso precedente.
FineSpecifica

1.5 Specifica delle classi

La classe Messaggio e le sue derivate

```
SpecificaClasse Messaggio
  mittente(): Persona
    pre: nessuna
    post: result dipende dalla classe piu' specifica a cui this appartiene.
  getIdRichiesta(): Stringa
    pre: nessuna
    post: result dipende dalla classe piu' specifica a cui this appartiene.
FineSpecifica

SpecificaClasse MessaggioCliente is-a Messaggio
  mittente(): Cliente
    pre: nessuna
    post: result dipende dalla classe piu' specifica a cui this appartiene.
  getIdRichiesta(): Stringa
    pre: nessuna
    post: result dipende dalla classe piu' specifica a cui this appartiene.
  applicativo(): ApplicativoSW
    pre: nessuna
    post: result dipende dalla classe piu' specifica a cui this appartiene.
FineSpecifica

SpecificaClasse MessaggioOperatore is-a Messaggio
  mittente(): Operatore
    pre: nessuna
    post: result e' pari a this.opMittente.Operatore;
  getIdRichiesta(): Stringa
    pre: nessuna
    post: result e' pari a this.operatoreRisponde.MessaggioCliente.getIdRichiesta().
FineSpecifica

SpecificaClasse MessaggioInizialeRichiestaAssistenza is-a MessaggioCliente
  mittente(): Cliente
    pre: nessuna
    post: result e' pari a this.clMittente.Cliente.
  getIdRichiesta(): Stringa
    pre: nessuna
```

```
    post: result e' pari a this.idRichiesta.
  applicativo(): ApplicativoSW
    pre: nessuna
    post: result e' pari a this.oggettoRichiesta.ApplicativoSW.
FineSpecifica

SpecificaClasse MessaggioSuccessivoCliente is-a MessaggioCliente
  mittente(): Cliente
    pre: nessuna
    post: result e' pari a
        this.clRisponde.MessaggioOperatore.operatoreRisponde.MessaggioCliente.mittente().
  getIdRichiesta(): Stringa
    pre: nessuna
    post: result e' pari a this.clRisponde.MessaggioOperatore.getIdRichiesta().
  applicativo(): ApplicativoSW
    pre: nessuna
    post: result e' pari a
       this.clRisponde.MessaggioOperatore.operatoreRisponde.MessaggioCliente.applicativo().
FineSpecifica
```

1.6 Diagramma degli stati/transizioni per gli oggetti della classe MessaggioCliente