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 **PI.20050411**, passo **A.1**

versione del 16 marzo 2007

Il Comune di Roma, d'intesa con il consorzio Metrebus, ha avviato nel 2005 l'erogazione del servizio di *car sharing* in città[1] (chiamato RomaCarSharing). Il servizio di car sharing permette ai soci di noleggiare, presso uno dei garage convenzionati, disseminati nel territorio, un'autovettura anche solo per poche ore, pagando l'uso effettivo del veicolo (ovvero un costo fisso, più un costo orario, mentre il carburante è compreso). I vantaggi sono molteplici, soprattutto per coloro che non usano frequentemente l'auto, potendo evitare i costi (assicurazione, bollo, manutenzione) derivanti dal possesso di un'auto privata.

Si vuole progettare un'applicazione che permetta di gestire alcune informazioni sul servizio di car sharing, relativamente ai soci e alle auto noleggiate.

Si richiede di effettuare la fase di Analisi, modellando la seguente specifica dei requisiti.

Requisiti

Dei soci interessano il codice fiscale, il nome, e il cognome. Di un socio è importante inoltre sapere se è iscritto al servizio come privato oppure come dipendente di una società. Dei soci privati interessa conoscere anche l'indirizzo di residenza, e le coordinate bancarie (ABI, CAB, e numero di conto) dove addebitare gli estratti conto, mentre dei soci dipendenti di società interessa la società presso cui lavorano. Di quest'ultima sono di interesse la ragione sociale, l'indirizzo della sede legale, e le coordinate bancarie dove addebitare gli estratti conto relativi ai noleggi dei suoi dipendenti.

I soci possono prenotare il noleggio di un'auto. Dei noleggi interessa, oltre il socio e l'auto prenotata, la data, l'ora e il garage in cui l'auto viene presa in consegna e quelli del rilascio (si noti che l'auto può essere rilasciata presso un garage diverso). Dei garage interessa l'identificativo (un intero), l'indirizzo, ed i numeri di telefono (almeno uno).

Delle auto è di interesse conoscere la targa, il modello, il numero di posti, la data dell'ultima manutenzione, e la categoria a cui appartiene. Le categorie delle auto (che hanno un nome,

[1] Per ora, in via sperimentale, solo nel III Municipio, dal 2006 sarà gradualmente esteso a tutto il territorio cittadino. Per info: `www.comune.roma.it`.

ad es. "city car", "berlina", ecc.) sono di interesse per l'applicazione, dato che da queste dipendono le tariffe applicate ai noleggi delle relative vetture. Si assuma che tali tariffe siano composte da un prezzo base più una tariffa oraria.

Alcuni soci vengono considerati *clienti abituali* e, in quanto tali, hanno diritto a sconti sulle tariffe dei noleggi. Un socio è considerato abituale se ha effettuato almeno 20 noleggi nel corso degli ultimi 12 mesi in caso sia un privato, o almeno 40 se usa il servizio per conto di una società.

Le auto si dividono in due tipologie, in base al tipo della loro alimentazione: in particolare, queste si classificano in auto ad alimentazione tradizionale (benzina o diesel) ed auto a carburanti ecocompatibili (metano, GPL, elettrica, ecc.). Delle auto ad alimentazione tradizionale interessa conoscere il numero di chilometri che percorrono al litro, mentre di quelle ecocompatibili il tipo di alimentazione (una stringa), e la loro autonomia (in Km).

Il sistema deve offrire le seguenti funzionalità:

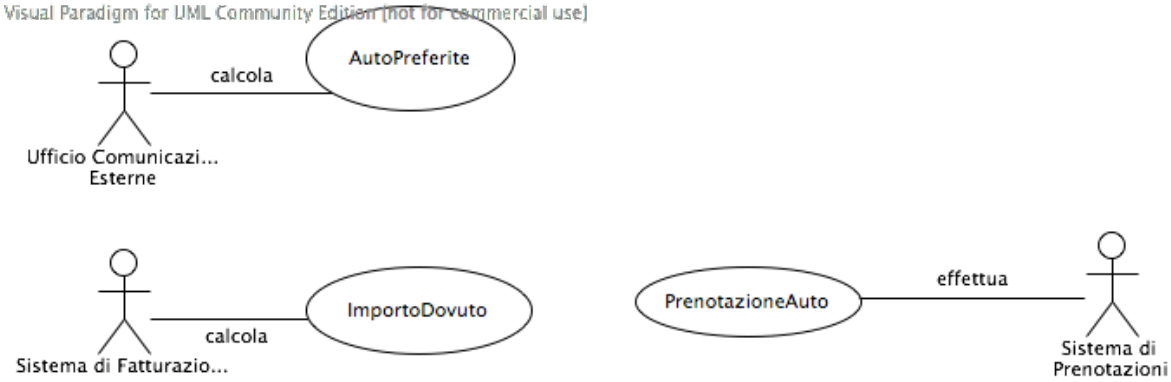
1. L'ufficio comunicazioni esterne del servizio RomaCarSharing, dato un socio, vuole conoscere le sue auto preferite, ovvero quelle che ha noleggiato il maggior numero di volte.

2. Il sistema di fatturazione, dato un socio ed un intervallo di tempo (nella forma di due date), ha bisogno di calcolare l'importo complessivo dei noleggi effettuati da quel socio nell'intervallo di tempo indicato (al fine di addebitare tale somma sul suo conto bancario). Si noti che, per i clienti abituali (ovvero tali al momento dell'invocazione dell'operazione), viene praticato uno sconto del 15%.

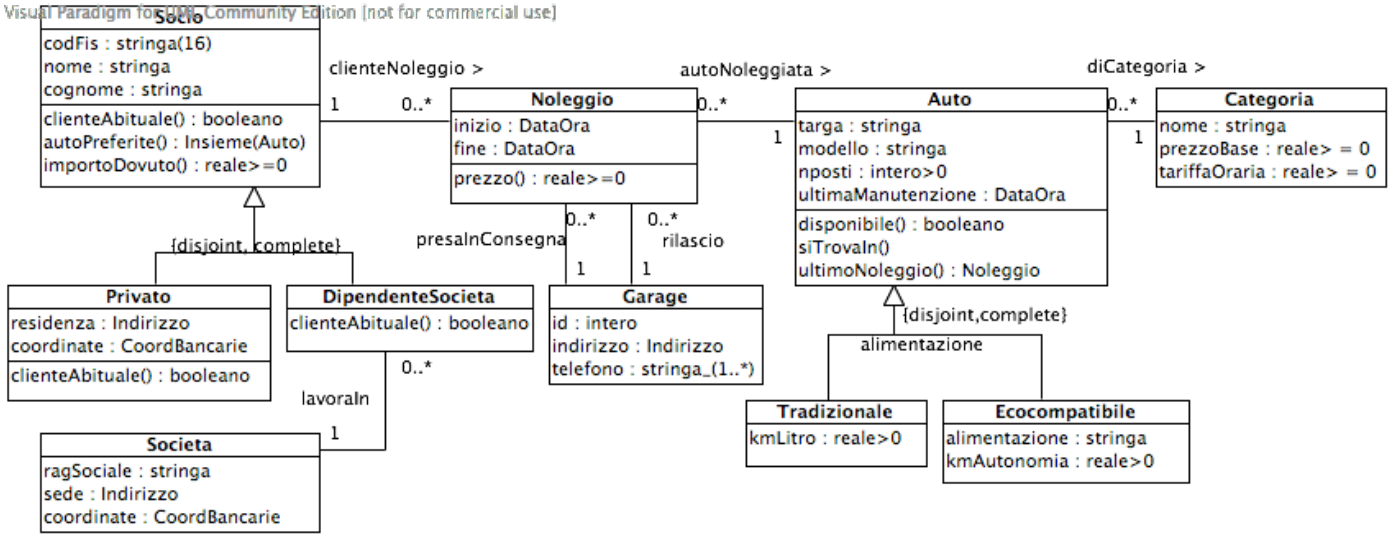
3. Il sistema di prenotazioni deve poter effettuare prenotazioni di auto per conto dei soci. Si noti che la prenotazione di un'auto può essere effettuata correttamente solo se, nel periodo richiesto, l'auto richiesta è *disponibile* (ovvero non è stata già noleggiata da altri), e si trova (ovvero è stata rilasciata dal suo ultimo utente) presso quel garage. Si assuma, per semplicità, che se l'auto non ha precedenti noleggi, la prenotazione ha successo, perché verrebbe spostata dal personale addetto presso il garage richiesto. Inoltre si ignori il fatto che l'auto debba essere rilasciata nel garage dove verrà presa dall'eventuale cliente successivo.

1 Fase di Analisi

1.1 Diagramma degli Use Case



1.2 Diagramma delle classi UML



1.3 Specifica degli use case

Use case AutoPreferite

```
SpecificaUseCase AutoPreferite
  autoPreferite(s : Socio) : Insieme(Auto)
    pre: nessuna
    post: result e' pari ad s.autoPreferite().
FineSpecifica
```

Use case ImportoDovuto

```
SpecificaUseCase ImportoDovuto
  importoDovuto(s : Socio, da : DataOra, a : DataOra) : reale >= 0
    pre: Quelle di s.importoDovuto().
    post: result e' pari a s.importoDovuto(da, a).
FineSpecifica
```

Use case PrenotazioneAuto

```
SpecificaUseCase PrenotazioneAuto
  prenotaAuto(s:Socio, auto:Auto, inizio:DataOra, da:Garage,
                         fine:DataOra, a:Garage) : Noleggio

    pre: 'fine' >= 'inizio',
            'inizio' > 'adesso', con 'adesso' l'istanza del tipo DataOra che rappresenta
                    l'istante corrente,
            auto.disponibile(inizio, fine) = true, e
            auto.siTrovaIn(inizio, da) = true.

    post: result e' pari ad un nuovo oggetto di classe Noleggio con:
                result.inizio = inizio,
                result.fine = fine.
             Inoltre, sono creati i seguenti nuovi link:
                <result, s> di associazione clienteNoleggio,
                <result,auto> di associazione autoNoleggiata,
                <result, da> di associazione presaInConsegna,
                <result,a> di associazione rilascio.
FineSpecifica
```

1.4 Specifica delle classi

La classe Socio

SpecificaClasse Socio
  clienteAbituale() : booleano
    pre: nessuna
    post:
        Sia N l'insieme dei noleggi effettuati da this nel corso degli ultimi 12 mesi,
        ovvero:

$$\mathtt{N} = \left\{ \begin{array}{l|ll} \mathtt{n} \in \mathtt{Noleggio} & \langle \mathtt{this}, \mathtt{n} \rangle \in \mathtt{clienteNoleggio} \wedge & \\ & \mathtt{adesso.differenza(n.inizio,\ ANNI)} \leq 1. & \end{array} \right\}$$

        result e' pari a true se e solo se |N| >= x, dove 'x' dipende dalla classe piu'
        specifica di this.

  autoPreferite() : Insieme(Auto)
     pre: nessuna
     post: Sia N l'insieme dei noleggi che coinvolgono this, ovvero:

$$\mathtt{N} = \{\mathtt{n} \in \mathtt{Noleggio} \mid \langle \mathtt{this}, \mathtt{n} \rangle \in \mathtt{clienteNoleggio}\}$$

    Sia inoltre A l'insieme delle auto noleggiate da this, ovvero:

$$\mathtt{A} = \{\mathtt{a} \in \mathtt{Auto} \mid \langle \mathtt{n}, \mathtt{a} \rangle \in \mathtt{autoNoleggiata\ e\ n} \in \mathtt{N}\}.$$

    Sia infine 'Amax' il sottoinsieme delle auto in 'A' oggetto del maggior
    numero di noleggi da parte di this, ovvero:

$$\mathtt{Amax} = \left\{ \begin{array}{l|l} \mathtt{a} \in \mathtt{A} & \mathtt{per\ ogni\ altro\ oggetto\ a'} \in \mathtt{A\ t.c.\ \ a'} \neq \mathtt{a\ si\ ha\ che} \\ & |\{\mathtt{n} \in \mathtt{N} \ :\ \langle \mathtt{n}, \mathtt{a} \rangle \in \mathtt{autoNoleggiata}\}| \geq \\ & \quad |\{\mathtt{n'} \in \mathtt{N} \ :\ \langle \mathtt{n'}, \mathtt{a'} \rangle \in \mathtt{autoNoleggiata}\}| \end{array} \right\}.$$

     result e' pari ad 'Amax'.

  importoDovuto(da : DataOra, a : DataOra) : reale >= 0
     pre: 'a' >= 'da',
             'a' <= 'adesso', con 'adesso' l'istanza del tipo DataOra rappresentante
                l'istante corrente.

     post: Sia N l'insieme dei noleggi che coinvolgono this, che iniziano nel
             periodo indicato, ovvero:

$$\mathtt{N} = \left\{ \begin{array}{l|ll} \mathtt{n} \in \mathtt{Noleggio} & \langle \mathtt{this}, \mathtt{n} \rangle \in \mathtt{clienteNoleggio} \wedge \\ & \mathtt{n.inizio} \geq \mathtt{da} \ \wedge \ \mathtt{n.inizio} < \mathtt{a} \end{array} \right\}$$

Sia 'importoBase' pari a:

$$\Sigma_{\mathtt{n} \in \mathtt{N}} \left(\mathtt{n.prezzo()} \right).$$

Se this.clienteAbituale() = true, result e' pari a importoBase * 0.85, altrimenti result e' pari a importoBase.

FineSpecifica

La classe Privato

```
SpecificaClasse Privato is-a Socio
  clienteAbituale() : booleano
    pre: nessuna
    post: Quelle di Socio.clienteAbituale() con x=20.
FineSpecifica
```

La classe DipendenteDiSocieta

```
SpecificaClasse DipendenteDiSocieta is-a Socio
  clienteAbituale() : booleano
    pre: nessuna
    post: Quelle di Socio.clienteAbituale() con x=40.
FineSpecifica
```

La classe Noleggio

```
SpecificaClasse Noleggio
  prezzo() : reale >= 0
    pre: nessuna
    post: Sia 'c' la categoria dell'auto noleggiata, ovvero
         c = this.autoNoleggiata.Auto.diCategoria.Categoria.

       result e' pari a c.prezzoBase + c.prezzoOrario*numore, dove

    numore e' la durata del noleggio this:
```

```
      numore = parteInteraSup(this.fine.differenza(this.inizio, ORE))

FineSpecifica
```

La classe Auto

```
SpecificaClasse Auto
  disponibile(inizio : DataOra, fine : DataOra) : booleano
    pre: fine >= inizio
    post:
       Se le precondizioni di this.ultimoNoleggio(fine) non sono rispettate (non ci
       sono noleggi che iniziano prima di fine), result e' pari a true.

      Altrimenti, sia n = this.ultimoNoleggio(fine) l'ultimo noleggio
      dell'auto this che inizia prima di fine.

      result e' true se e solo se n.fine <= inizio.

  ultimoNoleggio(momento : DataOra) : Noleggio
    (operazione ausiliaria introdotta per semplificare la specifica delle
     altre operazioni)

    pre:
       Detto 'N' l'insieme dei noleggi relativi all'auto this che iniziano fino a
       'momento', ovvero:
```

$$\mathtt{N} = \left\{ \begin{array}{l|ll} \mathtt{n} \in \mathtt{Noleggio} & \langle \mathtt{n}, \mathtt{this} \rangle \in \mathtt{autoNoleggiata} \text{ e} \\ & \mathtt{n.inizio} \leq \mathtt{momento} \end{array} \right\}.$$

```
    N non e' vuoto.

    post:
       Sia n_max l'ultimo noleggio in N in ordine cronologico, ovvero tale che,
       per ogni n in N, si ha n_max.inizio >= n.inizio.

       result e' pari a n_max.

  siTrovaIn(momento : DataOra, garage : Garage) : booleano
    pre: nessuna
```

```
    post:
       Se le precondizioni di this.ultimoNoleggio(momento) non sono
       rispettate (non ci sono noleggi fino a 'momento'), result = true.

       Altrimenti, sia n = this.ultimoNoleggio(momento)
       l'ultimo noleggio dell'auto this fino a 'momento'.

       result e' true se e solo se n.fine <= momento e n.rilascio.Garage = garage.

FineSpecifica
```

1.5 Specifica dei tipi di dato

Il tipo Indirizzo

```
SpecificaTipoDiDato Indirizzo
  attributi
    via: Stringa
    civico: Stringa
    citta: Stringa
    cap: intero >= 0
FineSpecifica
```

Il tipo Stringa(n))

```
SpecificaTipoDiDato Stringa(n)
  attributi
    s: Stringa
  operazioni
    Stringa(n)(ss: Stringa) : Stringa(n)
      pre: |ss| = n;
      post: result e' l'istanza del tipo con result.s = ss;
FineSpecifica
```

Il tipo CoordBancarie

```
SpecificaTipoDiDato CoordBancarie
  attributi
```

```
    abi: intero > 0
    cab: intero > 0
    cc: intero > 0
FineSpecifica
```