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.3**

versione del 28 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 una terza (ed ultima) iterazione della fase di Analisi, modellando i seguenti requisiti aggiuntivi.

Requisiti

Durante il noleggio di un'auto, è possibile che accadano dei sinistri. Di ogni sinistro, interessa conoscere, oltre alla sua descrizione, e alle informazioni sulla data, l'ora e il luogo dove è avvenuto, anche se ha provocato feriti, e qual è l'ammontare economico dei danni provocati. I sinistri sono di due tipi: quelli per cui esiste una controparte (ovvero incidenti con altri veicoli), e quelli senza controparte (ad es., incidenti dovuti ad altre cause). Dei primi interessa conoscere la targa dell'altro veicolo coinvolto, e sapere se il socio del servizio è responsabile (ovvero ha colpa).

Il sistema deve offrire le seguenti funzionalità aggiuntive:

1. L'Ufficio Sinistri, dato un socio, vuole calcolare la sua *classe di rischio*, che dipende dal numero n di sinistri senza controparte più quelli con controparte di cui è responsabile

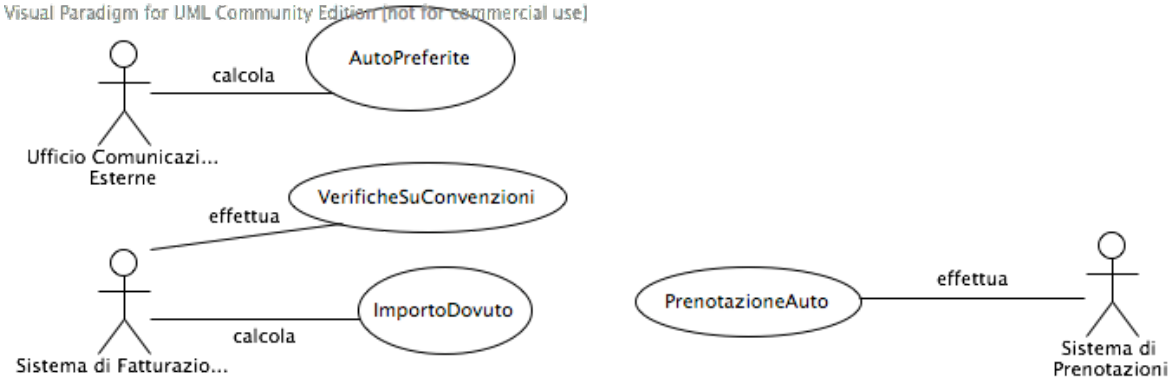
[1] Per ora, in via sperimentale, solo nel III Municipio, dal 2006 sarà gradualmente esteso a tutto il territorio cittadino. Per info: `www.comune.roma.it`.

avvenuti negli ultimi 3 anni. In particolare, la classe di rischio è 0 se $n = 0$, è pari ad n se $0 < n < 10$, ed è pari a 10 se $n >= 10$.

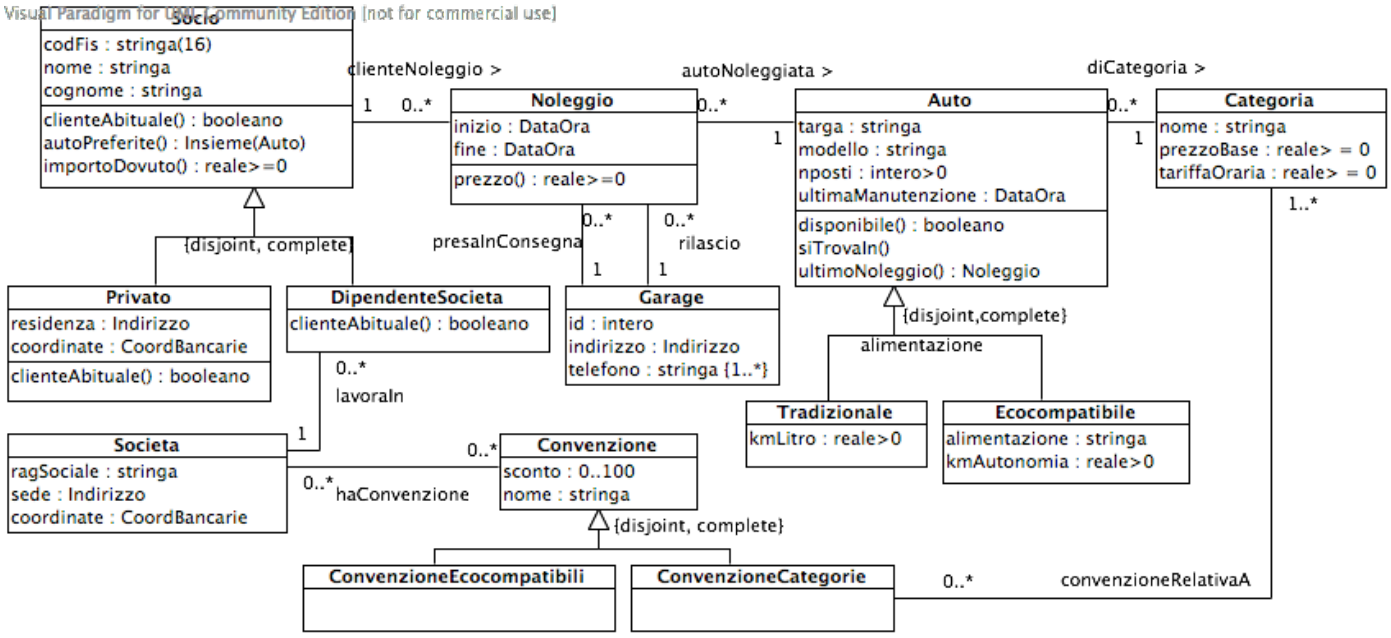
2. L'Ufficio Sinistri, dati un'auto (di cui si è scoperto un danno), ed un'istante di tempo (nella forma di una data ed un'ora), vuole conoscere l'ultimo utente che l'ha guidata (a quel momento).

1 Fase di Analisi

1.1 Diagramma degli Use Case prodotto nel passo A.2

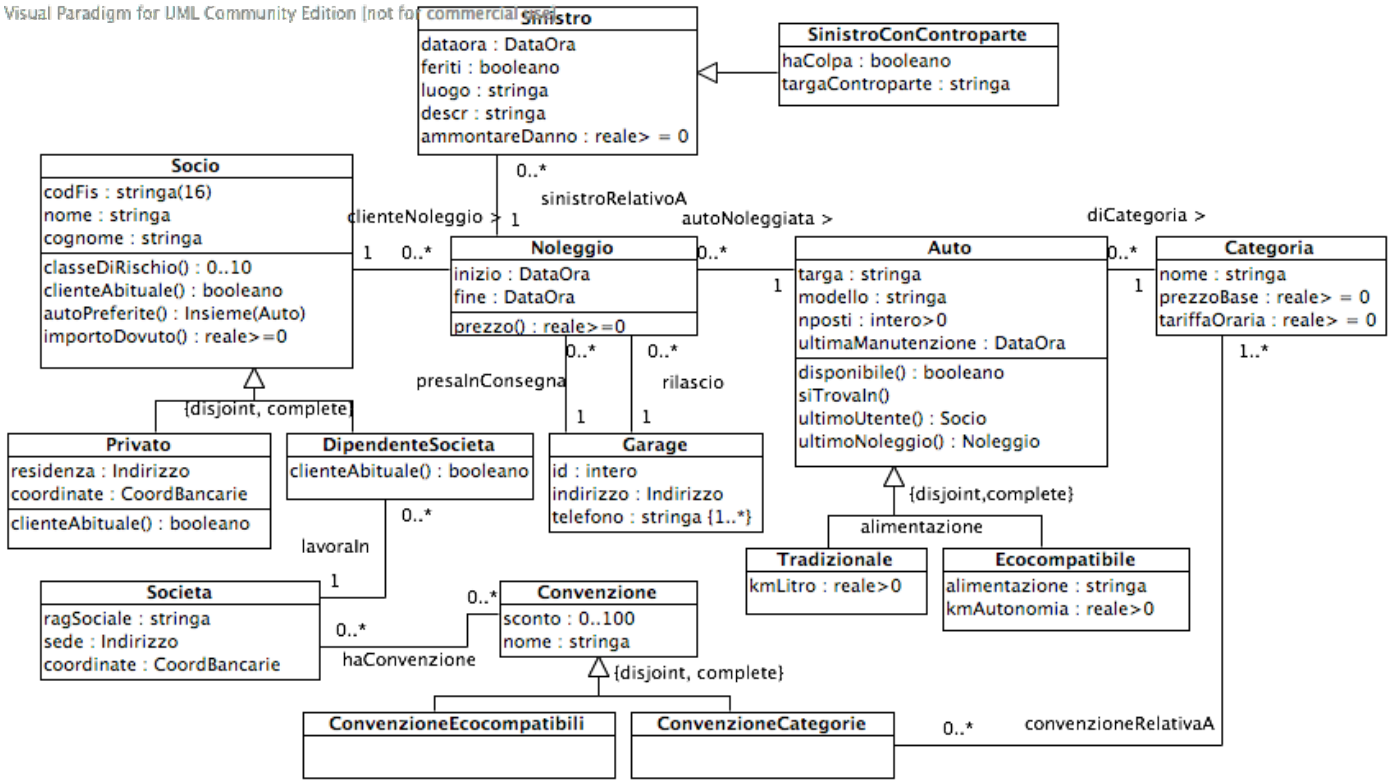


1.2 Diagramma delle classi UML prodotto nel passo A.2



1.3 Diagramma degli Use Case

1.4 Diagramma delle classi UML



1.5 Specifica degli use case

Use case ClasseDiRischio

```
SpecificaUseCase ClasseDiRischio
  classeDiRischio(s:Socio) : 0..10
    pre: nessuna
    post: result e' pari a s.classeDiRischio().
FineSpecifica
```

Use case UltimoUtente

```
SpecificaUseCase UltimoUtente
  ultimoUtente(a:Auto, momento:DataOra) : Socio
    pre: Quelle di a.ultimoUtente(momento).
    post: result e' pari a a.ultimoUtente(momento).
```

```
FineSpecifica
```

1.6 Specifica delle classi

La classe Socio

```
SpecificaClasse Socio

  classeDiRischio() : 0..10
    pre: nessuna
    post: Detta 'adesso' l'istanza del tipo 'DataOra' rappresentante
             l'istante corrente, sia N l'insieme dei noleggi che coinvolgono this
             effettuati negli ultimi 3 anni, ovvero:
```

$$\texttt{N} = \left\{ \texttt{n} \in \texttt{Noleggio} \mid \begin{array}{l} \langle \texttt{this}, \texttt{n} \rangle \in \texttt{clienteNoleggio} \wedge \\ \texttt{adesso.differenza(n.inizio, ANNI)} \leq 3 \end{array} \right\}$$

```
    Sia 'S' l'insieme di sinistri (di ogni tipo) avvenuti durante i noleggi in 'N',
    ovvero:
```

$$\texttt{S} = \{\texttt{s} \in \texttt{Sinistro} \mid \texttt{esiste n} \in \texttt{N t.c.} \quad \langle \texttt{n}, \texttt{s} \rangle \in \texttt{sinistroRelativoA}\}.$$

```
    Sia infine S' l'insieme dei sinistri in S senza controparte oppure con
    controparte di cui il socio ha colpa, ovvero:
```

$$\texttt{S'} = \texttt{S} - \{\texttt{s} \in \texttt{S} \mid \texttt{s e' di classe SinistroConControparte e s.haColpa = false}\}.$$

```
    Se |S'| <= 10, result e' pari a |S'|, altrimenti result e' pari a 10.

  clienteAbituale() : booleano
    cf. passo A.1

  autoPreferite() : Insieme(Auto)
    cf. passo A.1

  importoDovuto(da : DataOra, a : DataOra) : reale >= 0
    cf. passo A.1
FineSpecifica
```

La classe Auto

```
SpecificaClasse Auto
  disponibile(inizio : DataOra, fine : DataOra) : booleano
    cf. passo A.1

  ultimoNoleggio(momento : DataOra) : Noleggio
    cf. passo A.1

  siTrovaIn(momento : DataOra, garage : Garage) : booleano
    cf. passo A.1

  ultimoUtente(momento : DataOra) : Socio
     pre: Le precondizioni di this.ultimoNoleggio(momento) sono rispettate.
     post: result e' pari a this.ultimoNoleggio(momento).clienteNoleggio.Socio.

FineSpecifica
```