COWS Specification of the On Road Assistance Scenario

Technical Report

Authors: Alessandro Lapadula, Rosario Pugliese and Francesco Tiezzi

Date: December 19, 2007
Institute: Dipartimento di Sistemi e Informatica Università degli Studi di Firenze
Revision: Draft

1 On road assistance scenario

The 'on road assistance scenario' is one of the scenarios in the area of automotive systems defined within the EU project SENSORIA which describes some functionalities that will be likely available in the near future. The scenario involves a number of services that are discovered and bound at run-time according to levels of service specified at design time, so as to deliver the best available functionalities at agreed levels of quality. A brief description follows.

> The in-vehicle *diagnostic* system reports a severe failure when the car is no longer drivable. The car's *discovery* system then identifies garages, car rentals and towing truck services in the car's vicinity. At this point, the car's *reasoner* system selects a set of adequate services taking into account personalised policies and preferences of the driver, e.g. balancing cost and delay, and tries to order them. Before being enable to order services, the owner of the car has to deposit a security payment, that will be given back if ordering the services fails. Other components of the in-vehicle service platform involved in this assistance activity are a *GPS* service, providing the car's current location, and an *orchestrator*, coordinating all the described services.

An UML-like activity diagram of the orchestration of services is shown in Figure 1. For simplicity, we assume that the orchestration is only triggered either by an 'engine failure' or by a 'low oil level' sensor signal. The process starts with a request from the orchestrator to the bank to charge the driver's credit card with the security deposit payment. This is modelled by the UML action requestCardCharge for charging the credit card whose number is provided as an output parameter of the action call. In parallel to the interaction with the bank, the orchestrator requests the current location of the car from the car's internal GPS service. The current location is modelled as an input to the requestLocation action and subsequently used by the findServices interaction which retrieves a list of services. If no service can be found, an action to compensate the credit card charge will be launched. For the selection of services, the orchestrator synchronises with the reasoner service to obtain the most appropriate (best) services.

Service ordering is modelled by the UML actions orderGarage, orderTowTruck and orderRentalCar. When the orchestrator makes an appointment with the garage, the diagnostic data are automatically transferred to the garage, which could then be able, e.g., to identify the spare parts needed to perform the repair. Then, the orchestrator makes an appointment with the towing service, providing the GPS data of the stranded vehicle and of the garage, to tow the vehicle to the garage. Concurrently, the orchestrator makes an appointment with the rental service, by indicating the location where the car will be handed over to the driver.

The workflow described in Figure 1 models the overall behaviour of the system. Besides interactions among services, it also includes activities using concepts developed for long running business transactions. These activities entail fault and compensation handling, kind of specific activities attempting to reverse the effects of previously committed activities, that are an important aspect of SOC applications. Specifically, in the considered scenario, the security deposit payment charged to the driver's credit card must be revoked if either the discovery phase does not succeed or ordering the services fails, i.e. both garage/tow truck and car rental services reject the requests. Moreover, if ordering a tow truck fails, the garage appointment has to be cancelled and the rental car delivery has to be redirected to the stranded car's actual location. Instead, if ordering the car rental fails, the overall process may not fail, as the activity is enclosed in a sub-transaction.

2 COWS specification

The COWS term representing the 'orchestration' of all services within the scenario is

$$[p_{car}]\,(\,\mathit{Orchestrator} \mid \mathit{GPS} \mid \mathit{Discovery} \mid \mathit{Reasoner} \mid \mathit{SensorsMonitor}\,) \mid \mathit{Bank} \mid \mathit{OnRoadRepairServices}$$

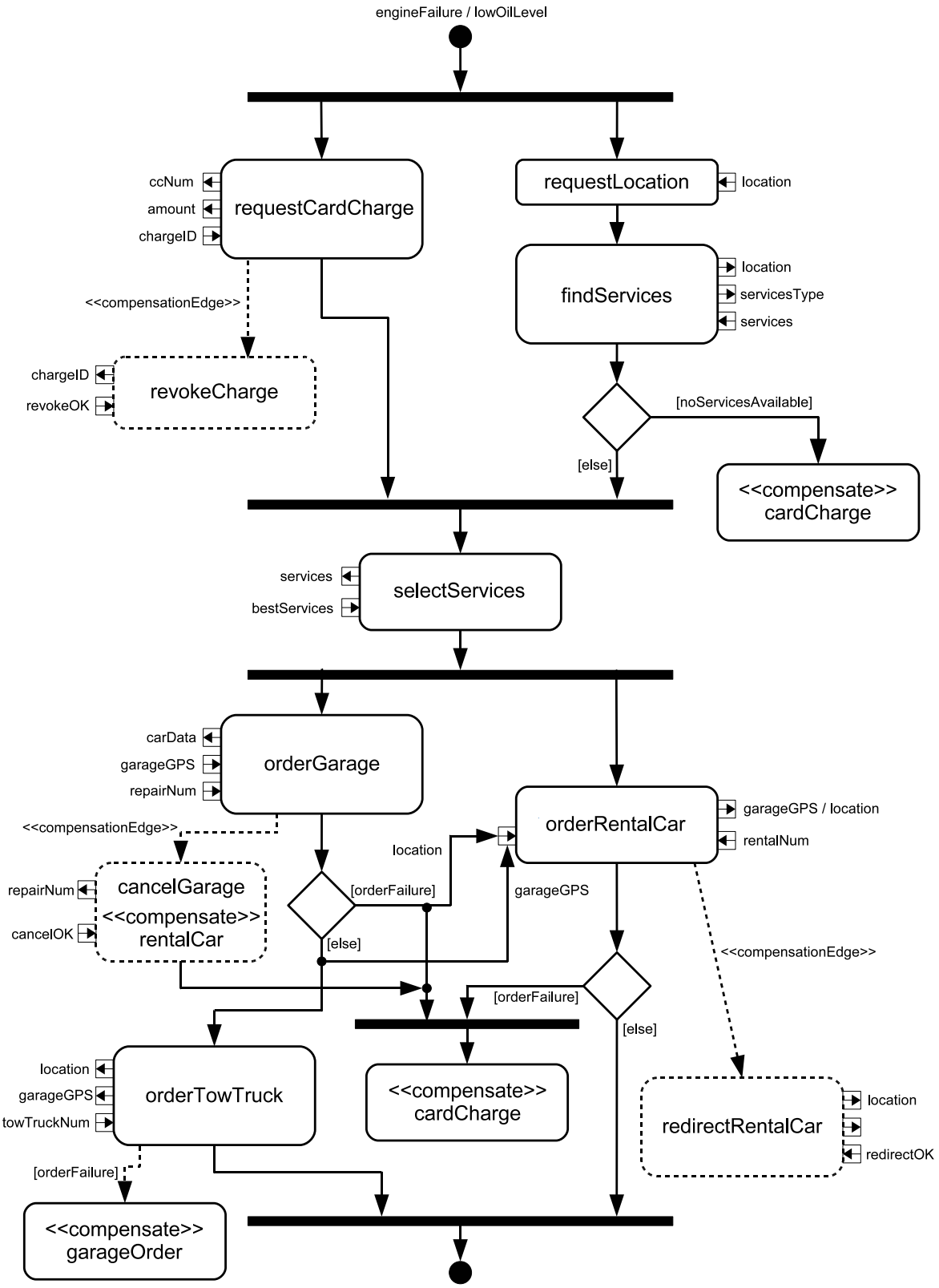

Figure 1: Orchestration in the on road assistance scenario

In the following COWS terms, for readability sake, fault and compensation activities are highlighted by a gray background to distinguish them from 'normal behaviour' activities.

2.1 *Orchestrator*

Orchestrator, the most important component of the in-vehicle platform, is

$$[x_{carData}]\,(\,p_{car} \bullet o_{engfail}?\langle x_{carData}\rangle.s_{engfail} \;+\; p_{car} \bullet o_{lowoil}?\langle x_{carData}\rangle.s_{lowoil}\,)$$

The recovery behaviour $s_{engfail}$ executed when an engine failure occurs is

$$\begin{array}{l}[p_e, o_e, x_{loc}, x_{list}, o_{undo}, k\,] \\ (\,(\,\mathit{requestCardCharge} \mid \mathit{requestLocation}.\mathit{findServices}\,) \\ \quad \mid\; p_e \bullet o_e?\langle\rangle.\; p_e \bullet o_e?\langle\rangle.\; \mathit{selectServices}. \\ \qquad\qquad [x_{garageGPS}]\,(\,\mathit{orderGarage}.\mathit{orderTowTruck} \mid \mathit{orderRentalCar}\,)\,)\end{array}$$

where

$$\begin{array}{rcl}
\mathit{requestCardCharge} & \triangleq & p_{bank} \bullet o_{charge}!\langle p_{car}, ccNum, amount\rangle \\
& & \mid \{\!|\, [x_{chargeID}]\, p_{car} \bullet o_{chargeFail}?\langle\rangle.\; \mathbf{kill}(k) \\
& & \qquad\qquad + \; p_{car} \bullet o_{chargeOK}?\langle x_{chargeID}\rangle. \\
& & \qquad\quad (\,p_e \bullet o_e!\langle\rangle \mid p_{car} \bullet o_{undo}?\langle cc\rangle.p_{car} \bullet o_{undo}?\langle cc\rangle. \\
& & \qquad\qquad\quad (\,p_{bank} \bullet o_{revoke}!\langle x_{chargeID}\rangle \\
& & \qquad\qquad\quad \mid p_{car} \bullet o_{revokeOK}?\langle\rangle\,)\,)\,|\!|\}\} \\
\mathit{requestLocation} & \triangleq & p_{car} \bullet o_{reqLoc}!\langle\rangle \mid p_{car} \bullet o_{respLoc}?\langle x_{loc}\rangle \\
\mathit{findServices} & \triangleq & p_{car} \bullet o_{find}!\langle x_{loc}, servicesType\rangle \\
& & \mid p_{car} \bullet o_{servicesFound}?\langle x_{list}\rangle.\; p_e \bullet o_e!\langle\rangle \\
& & \quad + p_{car} \bullet o_{servicesNotFound}?\langle\rangle. \\
& & \qquad (\,\mathbf{kill}(k) \mid \{\!|\, p_{car} \bullet o_{undo}!\langle cc\rangle \mid p_{car} \bullet o_{undo}!\langle cc\rangle \,|\!|\}\,) \\
\mathit{selectServices} & \triangleq & p_{car} \bullet o_{select}!\langle x_{list}\rangle \\
& & \mid [x_{garage}, x_{towTruck}, x_{rentalCar}]\, p_{car} \bullet o_{best}?\langle x_{garage}, x_{towTruck}, x_{rentalCar}\rangle \\
\mathit{orderGarage} & \triangleq & x_{garage} \bullet o_{order}!\langle p_{car}, x_{carData}\rangle \\
& & \mid [x_{repairNum}]\, p_{car} \bullet o_{garageFail}?\langle\rangle. \\
& & \qquad\qquad (\,p_{car} \bullet o_{undo}!\langle cc\rangle \\
& & \qquad\qquad \mid [p, o]\,(p \bullet o!\langle x_{loc}\rangle \mid p \bullet o?\langle x_{garageGPS}\rangle)\,) \\
& & \qquad + \; p_{car} \bullet o_{garageOK}?\langle x_{repairNum}, x_{garageGPS}\rangle. \\
& & \qquad\quad p_{car} \bullet o_{undo}?\langle garage\rangle. \\
& & \qquad\quad (\,x_{garage} \bullet o_{cancel}!\langle x_{repairNum}\rangle \\
& & \qquad\quad \mid p_{car} \bullet o_{cancelOK}?\langle\rangle \\
& & \qquad\quad \mid p_{car} \bullet o_{undo}!\langle cc\rangle \mid p_{car} \bullet o_{undo}!\langle rentalCar\rangle\,) \\
\mathit{orderTowTruck} & \triangleq & x_{towTruck} \bullet o_{order}!\langle p_{car}, x_{loc}, x_{garageGPS}\rangle \\
& & \mid [x_{towTruckNum}]\, p_{car} \bullet o_{towTruckFail}?\langle\rangle.\; p_{car} \bullet o_{undo}!\langle garage\rangle \\
& & \qquad\qquad + \; p_{car} \bullet o_{towTruckOK}?\langle x_{towTruckNum}\rangle
\end{array}$$

$$
\begin{aligned}
\mathit{orderRentalCar} \triangleq\ & x_{rentalCar} \bullet o_{order}!\langle p_{car}, x_{garageGPS} \rangle \\
& \mid [x_{rentalNum}]\, p_{car} \bullet o_{rentalCarFail}?\langle\rangle.\ p_{car} \bullet o_{undo}!\langle cc \rangle \\
& \qquad + \ p_{car} \bullet o_{rentalCarOK}?\langle x_{rentalNum} \rangle. \\
& \qquad\ p_{car} \bullet o_{undo}?\langle rentalCar \rangle. \\
& \qquad\quad (x_{rentalCar} \bullet o_{redirect}!\langle x_{rentalNum}, x_{loc} \rangle \\
& \qquad\quad \mid p_{car} \bullet o_{redirectOK}?\langle\rangle)
\end{aligned}
$$

2.2 The other car's components

$$
\begin{aligned}
\mathit{GPS} \triangleq\ & * p_{car} \bullet o_{reqLoc}?\langle\rangle. \\
& [x_{gps}]\, (< \texttt{compute GPS location and assign it to } x_{gps} > \\
& \qquad \mid p_{car} \bullet o_{respLoc}!\langle x_{gps} \rangle)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Discovery} \triangleq\ & * [x_{loc}, x_{type}, o_{found}, o_{notFound}, x_{list}] \\
& p_{car} \bullet o_{find}?\langle x_{loc}, x_{type} \rangle. \\
& (< \texttt{compute the services list, assign it to } x_{list}, \\
& \qquad \texttt{and reply on } o_{found} \texttt{ or } o_{notFound} > \\
& \quad \mid p_{car} \bullet o_{found}?\langle\rangle.\ p_{car} \bullet o_{servicesFound}!\langle x_{list} \rangle \\
& \qquad + \ p_{car} \bullet o_{notFound}?\langle\rangle.\ p_{car} \bullet o_{servicesNotFound}!\langle\rangle)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Reasoner} \triangleq\ & * [x_{services}, x_{best\_g}, x_{best\_t}, x_{best\_r}] \\
& p_{car} \bullet o_{select}?\langle x_{services} \rangle. \\
& (< \texttt{compute best services and assign their} \\
& \qquad \texttt{partner names to } x_{best\_g},\ x_{best\_t},\ x_{best\_r} > \\
& \quad \mid p_{car} \bullet o_{best}!\langle x_{best\_g}, x_{best\_t}, x_{best\_r} \rangle)
\end{aligned}
$$

$$
\mathit{SensorsMonitor} \triangleq p_{car} \bullet o_{engfail}!\langle diagnosticData \rangle
$$

2.3 The external services

$$
\begin{aligned}
\mathit{Bank} \triangleq\ & * [x_{cust}, x_{cc}, x_{amount}, o_{checkOK}, o_{checkFail}] \\
& p_{bank} \bullet o_{charge}?\langle x_{cust}, x_{cc}, x_{amount} \rangle. \\
& (< \texttt{perform some checks and reply on } o_{checkOK} \texttt{ or } o_{checkFail} > \\
& \quad \mid p_{bank} \bullet o_{checkFail}?\langle\rangle.\ x_{cust} \bullet o_{chargeFail}!\langle\rangle \\
& \qquad + \ p_{bank} \bullet o_{checkOK}?\langle\rangle.\ [chargeID]\, (x_{cust} \bullet o_{chargeOK}!\langle chargeID \rangle \\
& \qquad\qquad \mid p_{bank} \bullet o_{revoke}?\langle chargeID \rangle. \\
& \qquad\qquad\ < \texttt{revoke } chargeID >.\ x_{cust} \bullet o_{revokeOK}!\langle\rangle))
\end{aligned}
$$

$$
\begin{aligned}
\mathit{OnRoadRepairServices} \triangleq\ & \mathit{Garage}_1 \mid \mathit{Garage}_2 \mid \mathit{TowTruck}_1 \mid \mathit{TowTruck}_2 \mid \mathit{RentalCar}_1 \\
& \mid \mathit{RentalCar}_2
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Garage}_i \triangleq\ & * [x_{client}, x_{sensorsData}, o_{checkOK}, o_{checkFail}] \\
& p_{garage\_i} \bullet o_{order}?\langle x_{client}, x_{sensorsData} \rangle. \\
& (< \texttt{perform some checks and reply on } o_{checkOK} \texttt{ or } o_{checkFail} > \\
& \quad \mid p_{garage\_i} \bullet o_{checkFail}?\langle\rangle.\ x_{client} \bullet o_{garageFail}!\langle\rangle \\
& \qquad + \ p_{garage\_i} \bullet o_{checkOK}?\langle\rangle. \\
& \qquad\ [repairNum]\, (x_{client} \bullet o_{garageOK}!\langle repairNum \rangle \\
& \qquad\qquad \mid p_{garage\_i} \bullet o_{cancel}?\langle repairNum \rangle. \\
& \qquad\qquad\ < \texttt{cancel } repairNum >.\ x_{client} \bullet o_{cancelOK}!\langle\rangle))
\end{aligned}
$$

$$
\begin{aligned}
\mathit{TowTruck}_i \triangleq\ & *[x_{client}, x_{carLoc}, x_{garageLoc}, o_{checkOK}, o_{checkFail}] \\
& p_{towTruck\_i} \bullet o_{order}?\langle x_{client}, x_{carLoc}, x_{garageLoc}\rangle. \\
& (\ <\ \texttt{perform some checks and reply on}\ o_{checkOK}\ \texttt{or}\ o_{checkFail}\ > \\
& \ \mid p_{towTruck\_i} \bullet o_{checkFail}?\langle\rangle.\ x_{client} \bullet o_{towTruckFail}!\langle\rangle \\
& \ \ + p_{towTruck\_i} \bullet o_{checkOK}?\langle\rangle. \\
& \ \ [\mathit{towTruckNum}]\ x_{client} \bullet o_{towTruckOK}!\langle \mathit{towTruckNum}\rangle\ )
\end{aligned}
$$

$$
\begin{aligned}
\mathit{RentalCar}_i \triangleq\ & *[x_{client}, x_{garageLoc}, o_{checkOK}, o_{checkFail}] \\
& p_{rentalCar\_i} \bullet o_{order}?\langle x_{client}, x_{garageLoc}\rangle. \\
& (\ <\ \texttt{perform some checks and reply on}\ o_{checkOK}\ \texttt{or}\ o_{checkFail}\ > \\
& \ \mid p_{rentalCar\_i} \bullet o_{checkFail}?\langle\rangle.\ x_{client} \bullet o_{rentalCarFail}!\langle\rangle \\
& \ \ + p_{rentalCar\_i} \bullet o_{checkOK}?\langle\rangle. \\
& \ \ [\mathit{rentNum}]\ (\ x_{client} \bullet o_{rentalCarOK}!\langle \mathit{rentNum}\rangle \\
& \qquad \mid [x_{newDest}]\ p_{rentalCar\_i} \bullet o_{redirect}?\langle \mathit{rentNum}, x_{newDest}\rangle. \\
& \qquad\quad <\ \texttt{redirect}\ \mathit{rentNum}\ \texttt{to}\ x_{newDest}\ >. \\
& \qquad\quad x_{client} \bullet o_{redirectOK}!\langle\rangle\ )\ )
\end{aligned}
$$

3 CMC specification

```
let
 GPS(car) = *car.reqLoc?<>. car.respLoc!<gps>

 Discovery(car) = *[LOC] [TYPE]
                   car.find?<LOC,TYPE>.
                   [p#][o#] (p.o!<> | p.o?<>. car.servicesFound!<list>
                                    + p.o?<>. car.servicesNotFound!<>)

 Reasoner(car) = [SERVICES] car.select?<SERVICES>.
                            car.best!<garage1,towTruck2,rentalCar1>

 SensorMonitor(car) = car.engfail!<diagnosticData>

 cardChargeAndFindServices(car,LOC,LIST,end,undo) =
  [k]( (bank.charge!<car,ccNum,amount>
        | { [CHARGEID]
            (car.chargeFail?<>.kill(k)
            + car.chargeOK?<CHARGEID>.
              ( end.end!<>
                | car.undo?<cc>. car.undo?<cc>.
                  ( bank.revoke!<CHARGEID>
                    | bank.revokeOK?<>) ) ) }
       )
       |
       (car.reqLoc!<>
        | car.respLoc?<LOC>.
          (car.find!<LOC,servicesType>
           | car.servicesFound?<LIST>. end.end!<>
             + car.servicesNotFound?<>.
               (kill(k) | { car.undo!<cc> | car.undo!<cc> } ) )
       )
    )


 ChooseAndOrder(car,LOC,LIST,CARDATA,undo) =
```

```
  [GARAGEGPS]
  ( car.select!<LIST>
    | [GARAGE] [TOWTRUCK] [RENTALCAR]
      car.best?<GARAGE,TOWTRUCK,RENTALCAR>.
      (
        ( -- orderGarage
          GARAGE.order!<car,CARDATA>
          | [REPAIRNUM](car.garageFail?<>.
                          (car.undo!<cc> | [p#][o#] (p.o!<LOC> | p.o?<
                              GARAGEGPS>.nil))
                        + car.garageOK?<REPAIRNUM,GARAGEGPS>.
                          ( ( --orderTowTruck
                              TOWTRUCK.order!<car,LOC,GARAGEGPS>
                              | [TOWTRUCKNUM](car.towTruckFail?<>. car.
                                  undo!<garage>
                                                + car.towTruckOK?<
                                                    TOWTRUCKNUM>.nil
                                              )
                            )
                            |
                            car.undo?<garage>.
                            (
                              GARAGE.cancel!<REPAIRNUM> | car.cancelOK?<>
                              | car.undo!<cc> | car.undo!<rentalCar>
                            )
                          )
                       )
        )
        |
        ( -- orderRentalCar
          RENTALCAR.order!<car,GARAGEGPS>
          | [RENTALNUM](car.rentalCarFail?<>. car.undo!<cc>
                        + car.rentalCarOK?<RENTALNUM>.
                            car.undo?<rentalCar>.
                              (RENTALCAR.redirect!<RENTALNUM,LOC> | car.
                                  redirectOK?<>.nil)
                       )
        )
      )
  )



Bank = * [CUST] [CC] [AMOUNT] [checkOK#] [checkFail#]
          bank.charge?<CUST,CC,AMOUNT>.
          (bank.checkOK!<> | bank.checkFail!<>
           | bank.checkFail?<>. CUST.chargeFail!<>
             + bank.checkOK?<>.
               [chargeID#] ( CUST.chargeOK!<chargeID>
                             | bank.revoke?<chargeID>. CUST.revokeOK!<>
                           )
          )

Garage(garage) = * [CLIENT] [SENSORSDATA] [checkOK#] [checkFail#]
                   garage.order?<CLIENT,SENSORSDATA>.
                   ( garage.checkOK!<> | garage.checkFail!<>
                     | garage.checkFail?<>. CLIENT.garageFail!<>
```

```
                          + garage.checkOK?<>.
                            [repairNum#]
                            ( CLIENT.garageOK!<repairNum,garageGPS>
                              | garage.cancel?<repairNum>.
                                CLIENT.cancelOK!<>
                            )
                      )

 TowTruck(towTruck) = * [CLIENT] [CARLOC] [GARAGELOC]
                        [checkOK#] [checkFail#]
                        towTruck.order?<CLIENT,CARLOC,GARAGELOC>.
                        ( towTruck.checkOK!<> | towTruck.checkFail!<>
                          | towTruck.checkFail?<>. CLIENT.towTruckFail!<>
                            +towTruck.checkOK?<>.
                             [towTruckNum#] CLIENT.towTruckOK!<towTruckNum>
                        )

 RentalCar(rentalCar) = * [CLIENT] [GARAGELOC] [checkOK#] [checkFail#]
                          rentalCar.order?<CLIENT,GARAGELOC>.
                          ( rentalCar.checkOK!<> | rentalCar.checkFail!<>
                            | rentalCar.checkFail?<>.
                                CLIENT.rentalCarFail!<>
                              + rentalCar.checkOK?<>.
                                [rentNum#]
                                ( CLIENT.rentalCarOK!<rentNum>
                                  | [NEWDEST]
                                    rentalCar.redirect?<rentNum,NEWDEST>.
                                    CLIENT.redirectOK!<>
                                )
                          )


in
   [car#]
   ( GPS(car) | Discovery(car) | Reasoner(car) | SensorMonitor(car)
     |
     -- Orchestrator
     [CARDATA]
     (
       car.engfail?<CARDATA>.[e#] [undo#] [LOC] [LIST] (
         cardChargeAndFindServices(car,LOC,LIST,e,undo)
         | e.e?<>. e.e?<>. ChooseAndOrder(car,LOC,LIST,CARDATA,undo)
       )
       + car.lowoil?<CARDATA>.nil
     )
   )
   |
   Bank()
   |
   Garage(garage1) | Garage(garage2)
   |
   TowTruck(towTruck1) | TowTruck(towTruck2)
   |
   RentalCar(rentalCar1) | RentalCar(rentalCar2)
end
```

```
----------------------
-- ABSTRACTION RULES --
----------------------
--
Abstractions {
 Action engfail -> request(engineFailure)
 Action charge -> request(charge)
 Action chargeOK -> response(charge)
 Action garage1.order!<$1,*> -> request(garage,$1)
 Action garage2.order!<$1,*> -> request(garage,$1)
 Action $1.garageOK -> response(garage,$1)
 Action $1.garageFail -> fail(garage,$1)
 Action cancel -> cancel(garage)
 Action towTruck1.order -> request(towTruck)
 Action towTruck2.order -> request(towTruck)
 Action towTruckOK -> response(towTruck)
 Action towTruckFail -> fail(towTruck)
 Action rentalCar1.order -> request(rentalCar)
 Action rentalCar2.order -> request(rentalCar)
 Action rentalCarOK!<$1> -> response(rentalCar,$1)
 Action rentalCarFail -> fail(rentalCar)
 Action redirect!<$1,*> -> cancel(rentalCar,$1)
 State engfail -> accepting_request(engineFailure)
}



-------------------
-- SocL FORMULAE --
-------------------
--
--
-- 1. AG accepting_request(engineFailure)
--
--
-- 2. AG [request(garage,$car)]
--       AF {response(garage,%car) or fail(garage,%car)} true
--
--
-- 3. not E[ true {not response(charge)}
--             U {request(garage) or request(rentalCar)} true ]
--
--
-- 4. EF {response(rentalCar,$rentalNum)}
--       EF {fail(towTruck)} AF {cancel(rentalCar,%rentalNum)} true
--
--
-- 5. EF {fail(rentalCar)} EF {response(towTruck)} true
--
--
-- 6. AG [fail(towTruck)] AF {cancel(garage)} true
--
--
-- 7. not E[ true {not response(garage)} U {request(towTruck)} true ]
--
```