



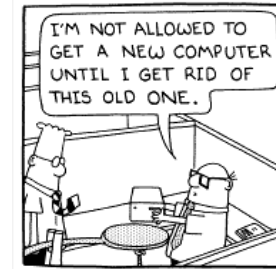
Poly Fleet Hire Case Study

Applying JSD to a Simple System Design Problem

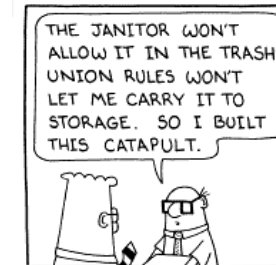


DILBERT Copyright © 1998 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited.

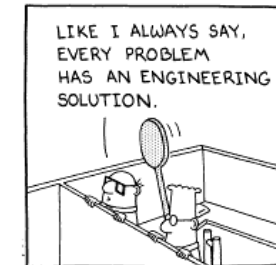
Paul Vickers



S. Adams E-MAIL: SCOTTADAMS@aol.com



© 1994 United Feature Syndicate, Inc.



Copyright © UFS, Inc.
Redistribution in whole or in part prohibited.

Problem Description

The Poly Fleet Hire company is a medium-sized, single location vehicle hire company employing three administrative staff and three technical (mechanical engineering) staff. The current vehicle fleet comprises 80 saloon cars.

It is anticipated that the fleet size will be increased to 120 saloon cars based in Merseyside and there are two other possible areas of expansion: the opening of a further similar sized operation in the midlands and the broadening of the hiring facility to include vans.

The management of Poly Fleet Hire wants to monitor the servicing and replacement of vehicles. The company replacement policy is to replace vehicles when the mileage exceeds 60,000 or, in the case of vehicles over two years old, when the mileage exceeds 45,000. The company servicing policy is to service all vehicles at least every ninety days irrespective of mileage.

When a car is hired a form is completed in duplicate; one copy for the file and one copy to be retained by the customer. This form records the customer's details and the details of the car including the mileage when the car goes out. When the car is returned the file copy is updated with the latest mileage figure.

Any system produced must allow for the provision of output in a timely manner, in respect of cars due for replacement or servicing. Also, management information to allow the regulation of the servicing and replacement policies may well be required. The hire fees collection and accounting is not to be considered at this time.

Use JSD to design but NOT implement a suitable system (i.e., you should proceed up to and including the SID and its supporting structure text). You will not be given any more information about Poly Fleet Hire, so if you consider there to be insufficient information you may make any valid assumptions as long as these are clearly stated, either in the design documents, or in the supporting material.

Real World Modelling Step

Assumptions from Analysis Notes

The specification for the work does not contain all that a real investigation would, as that would obviously be far too long for a short case study. To put the system herein described and designed into context, a list of assumptions has been made and is shown below:

- A customer of Poly Fleet Hire may have many cars on hire at any single moment.
- A customer can place a hiring and then amend the hiring details prior to taking the vehicle away. Vehicle hirings can also be cancelled before the vehicle is taken out. Once the vehicle is taken, or hired, the hiring is confirmed.
- Before a vehicle can start its life within the company, it must be acquired. This will form the starting point of a vehicle's life with the company.
- During a hiring the vehicle is unavailable for any hire or servicing activity. Similarly, after replacement, the vehicle is also unavailable.
- When a vehicle is required for service, it is first booked in, and when servicing is completed, it is booked out.
- The customer will neither steal nor bring back late the vehicle. Further, on return, the car will be ready for hiring, that is, it will not need repairing.

Candidate Action List

The following actions are derived from the specification and any of the previously presented assumptions.

- REPLACE
- SERVICE
- HIRE
- RETURN
- PLACE
- AMEND
- CANCEL
- USE
- ACQUIRE
- BOOK-IN
- BOOK-OUT

Revision of Actions

On reflection, the SERVICE action is not atomic, it comprises the two actions BOOK-IN and BOOK-OUT and is thus rejected.

The action USE does not affect any entity in any way, and does not cause any change and is also rejected.

The remaining actions were retained for the subsequent stages.

Actions and Attributes

Action REPLACE

Summary When a vehicle has done more than 60,000 miles and is 2 years old or less, or if it is older than 2 years and has done more than 45,000 miles then it is replaced and is not available for hire.

Attribute List :

VEHICLE NUMBER
DATE OF REPLACEMENT

Action HIRE

Summary After placing a hiring, the customer takes the vehicle out for use on the agreed date.

Attribute List :

HIRE NUMBER
OUT DATE
OUT TIME

Action RETURN

Summary When a customer has finished using the vehicle he brings it back to the company on the agreed date.

Attribute List :

HIRE NUMBER
NEW MILEAGE

Action PLACE

Summary The customer makes a request for a vehicle for a given time and date. This request is open to amendments.

Attribute List :

HIRE NUMBER
HIRE DETAILS

Action AMEND

Summary After placing a hiring a customer may change the details before taking the car away.

Attribute List :

HIRE NUMBER
AMENDED DETAILS

Action CANCEL

Summary The customer may cancel a hiring that has been previously made prior to taking the vehicle away.

Attribute List :

HIRE NUMBER

Action ACQUIRE

Summary Before a vehicle can begin its useful life it must be acquired by the company in the usual way (e.g. buying or leasing).

Attribute List :

VEHICLE NUMBER
VEHICLE MAKE
VEHICLE TYPE
START DATE
MILEAGE
LAST SERVICE DATE

Action BOOK-IN

Summary When a vehicle has been tagged for servicing or is overdue (>90 days since the last service) it is booked in for a service. The car becomes unavailable for hire.

Attribute List :

VEHICLE NUMBER

Action BOOK-OUT

Summary After a vehicle has been services it is booked out of its service and is made available for hire again.

Attribute List :

VEHICLE NUMBER
DATE OUT

Candidate Entity List

From reading the analysis notes, the following were identified as candidate entities for possible inclusion in the model:

- POLY FLEET HIRE COMPANY
- STAFF
- FLEET
- VEHICLE
- CAR
- VAN
- MANAGEMENT
- HIRE FORM
- COPY
- CUSTOMER
- HIRING

Revised Entity List

The list of candidate entities is now refined to remove and duplicates, phantoms and those entities that lie outside the model boundary (OMB).

It can be seen that the entities CAR, VEHICLE and VAN are, in the context of this exercise, synonyms as the system will be interested in the servicing and replacement of vehicles, regardless of class. Therefore, these three entities can be represented by VEHICLE.

The entity FLEET is merely a collection of vehicles and so is discarded.

The POLY FLEET HIRE COMPANY entity is not the basis of any needed function; indeed it represents the whole real world of the exercise. Therefore it is regarded as being OMB and is rejected. Similarly the MANAGEMENT entity is not the basis of any needed function and so is also OMB; it too is rejected.

COPY is regarded as being synonymous with HIRE FORM as the former is merely an updated version of the latter. Further, it can be argued that the HIRE FORM is more an artifact of the current manual process and what is actually of interest are the details it holds which are really attributes of a hiring. It is rejected.

The HIRING entity is one that is implied by the analysis notes; when a customer places a request a hiring begins which terminates upon return of the vehicle or cancellation.

Entity/Action Cross-Reference

The identified actions are now cross-referenced with the candidate entities to give them their time ordered set of events and to identify any phantom entities, or those with no time ordering.

Entity VEHICLE

Summary This entity models the activities of a vehicle in the fleet from initial acquisition to final replacement.

Action List :

ACQUIRE
BOOK-IN
BOOK-OUT
REPLACE
HIRE
RETURN

Entity CUSTOMER

Summary This entity models the activities of a customer during his dealings with the Poly Fleet Hire Company.

Action List :

PLACE
AMEND
CANCEL
HIRE
RETURN

Entity HIRING

Summary This entity models all the activities of the administration of the hiring activities from the customer placing a request to returning the vehicle.

Action List :

- PLACE
- AMEND
- CANCEL
- HIRE
- RETURN

Entity Attributes

Attributes are now shown for each of the remaining entities. The primary key(s) is shown in parentheses.

Entity VEHICLE

Attribute List :

- (VEHICLE NUMBER)
- MAKE
- TYPE
- COMMISSIONING DATE
- DATE OF LAST SERVICE
- MILEAGE
- NUMBER OF HIRINGS
- WITHDRAWAL DATE

Entity CUSTOMER

Attribute List :

- (CUSTOMER NUMBER)
- NAME
- ADDRESS
- TELEPHONE NUMBER

Entity HIRING

Attribute List :

- (HIRE NUMBER)
- CUSTOMER NUMBER
- VEHICLE NUMBER
- OUT DATE
- OUT TIME
- BACK DATE
- BACK TIME

Process Structure Diagrams

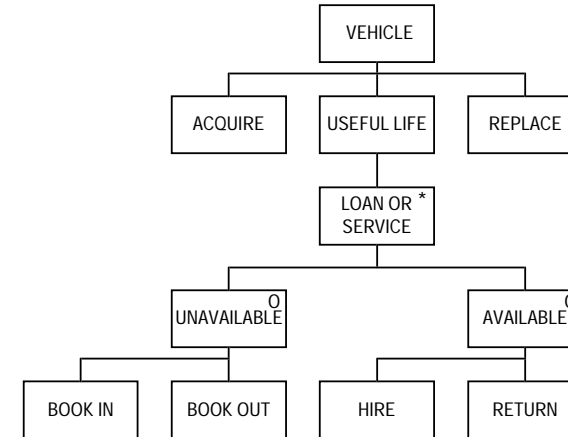


Figure 1: VEHICLE

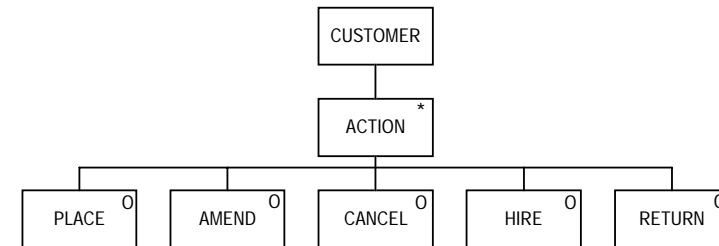


Figure 2: CUSTOMER

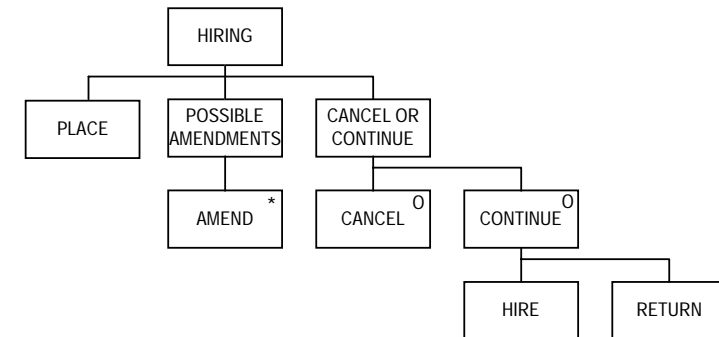


Figure 3: HIRING

Structure Texts

Entity CUSTOMER

```

Customer ITER (While still a customer)
  Action SEL (IF place)
    PLACE ;
  Action ALT (IF amendment)
    AMEND ;
  Action ALT (IF cancellation)
    CANCEL ;
  Action ALT (IF hiring)
    HIRE ;
  Action ALT (IF returning)
    RETURN ;
  Action END
Customer END.

```

Entity VEHICLE

```

Vehicle SEQ ;
ACQUIRE ;
Useful Life ITER (While car not due for replacement)
  Loan or Service SEL (IF due for service)
  Unavailable SEQ
    BOOK IN ;
    BOOK OUT ;
  Unavailable END
  Loan or Service ALT (IF available for hire)
  Available SEQ
    HIRE ;
    RETURN ;
  Available END
  Loan or Service END
Useful Life END
REPLACE ;
Vehicle END.

```

Entity HIRING

```

Hiring SEQ
  PLACE ;
  Possible Amendments ITER (While still amending)
    AMEND ;
  Possible Amendments END
  Cancel or Continue SEL (IF hiring cancelled)
    CANCEL ;
  Cancel or Continue ALT (IF proceeding with hiring)
  Continue SEQ
    HIRE ;
    RETURN ;
  Continue END
  Cancel or Continue END
Hiring END.

```

Networking Step

Initial Model Phase

The System Specification Diagram (SSD) for the initial modelling phase is given as figure 4.

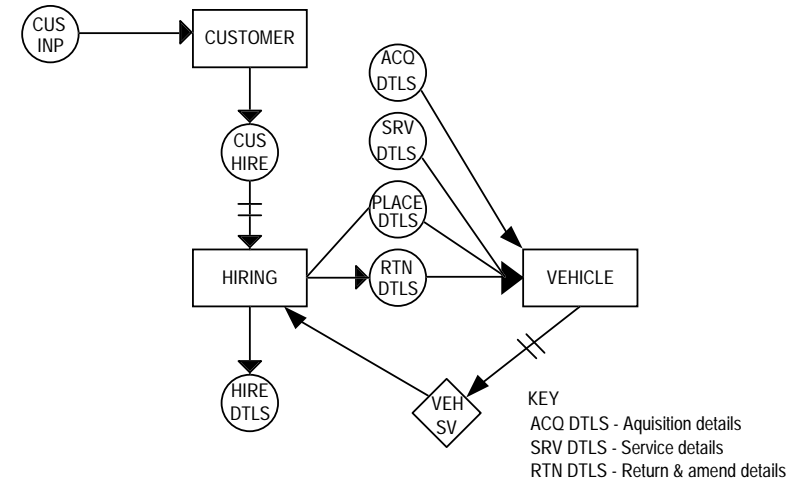


Figure 4: Initial Model SSD

The ACQ DTLS datastream is included as it was thought (and assumed) that a vehicle would have to be acquired before it could be used. Therefore, new vehicles must have their details delivered from somewhere. This would cause the ACQUIRE action to be completed and the VEHICLE life history to move into its USEFUL LIFE stage.

The SRV DTLS stream forms the messages relating to vehicle servicing. When a vehicle is booked in or out, a message must arrive from the outside world to indicate to the VEHICLE model process that an occurrence of it is being serviced or is being returned to the pool for hiring. As previously detailed, if a vehicle has exceeded its 90 day deadline then it is automatically booked in for a service and so only needs to be booked out when the service is complete.

The datastream, CUS HIRE, has a one-to-many relationship linking one CUSTOMER to many HIRINGs. This is because it is assumed that a single customer may have many hirings live at one time, and so will cause many instances of the HIRING process to exist for each CUSTOMER.

The two datastreams from HIRING to VEHICLE are PLACE DTLS and RTN DTLS. It was considered useful to have two different streams for this process connection. The PLACE DTLS stream is one of messages relating to new hirings. The RTN DTLS stream is a flow of messages relating to existing hirings and so it appeared well to separate the two streams.

The state vector connection, VEH SV, allows the process HIRING to check on the availability of vehicles when dealing with PLACE and AMEND actions.

The fixed merge into VEHICLE allows that process to be blocked until a message arrives on the datastream ACQ DTLS. This is necessary as it is clearly absurd to allow a vehicle to be hired before it has been acquired.

Elaboration Phase

The elaborated System Specification Diagram is given as figure 5.

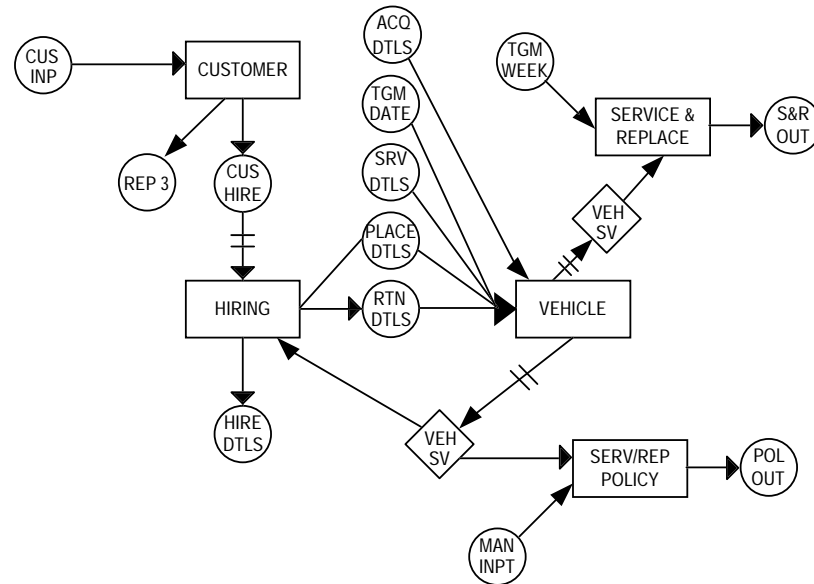


Figure 5: Elaborated SSD

Within the specified information requirements, mention is made of two reports: one about the servicing and replacement of vehicles and one for the service and replacement policy. In the elaboration phase we start to consider the functional requirements of the proposed system. Therefore, two function processes are added to the model. Each requires access to the VEHICLE state vector file (maintained and owned by the VEHICLE process) and generates as output a datastream.

The function process SERVICE & REPLACE writes the datastream S&R OUT. This contains information about vehicles that have been serviced or replaced in a particular week. As none of the existing model processes are modelled around the notion of a week, it is necessary to introduce a time grain marker, TGM WEEK, which supplies the real world time triggers necessary for the function to operate.

The function process SERV/REP POLICY meets the requirement of producing ad-hoc reports telling of any vehicles that may be due for servicing and/or replacing. As this report is of an ad-hoc nature, then it is not generated at defined intervals but at the request of the company's management. Thus it depends on the datastream MAN INPT to trigger it. The report is sent out into the POL OUT datastream.

While the detailed requirements of the two report function processes are not defined in the analysis notes, some mechanism at least is in place in the SSD for their production.

Two time grain markers have been added to the SSD. TGM DATE delivers a daily date stamp to the VEHICLE process. This allows the BOOK IN and REPLACE actions to be triggered when necessary. The BOOK IN action will automatically occur if a vehicle has exceeded its 90 days since the last service date. (Naturally, checks will have to be made at some point so that this does not happen when it is out on hire.) If a vehicle has exceeded its useful life (mileage and date dependent) the REPLACE action will be forced. The mileage is handled elsewhere, but the two-year element of the replacement policy and the 90 day service check are looked after by this daily time grain marker.

The time grain marker, TGM WEEK, is to meet the requirement for the service and replacement report to be generated in a timely manner. This time grain marker will be produced each week triggering the reporting process. As the frequency of the report was not specified, the marker could equally well be a daily or monthly one.

2.3 Elaborated Structure Texts

Following the networking, the structure texts for the model processes are amended as show below.

```

Entity VEHICLE
Vehicle SEQ ;
Read (ACQ DTLS) ;
Read (TGM DATE + SRV/PLACE/RTN DTLS) ;
ACQUIRE ;
Useful Life ITER (While car not due for replacement)
  Loan or Service SEL (IF due for service)
  Unavailable SEQ
    BOOK IN ;
    Read (TGM DATE + SRV/PLACE/RTN DTLS) ;
    BOOK OUT ;
    Read (TGM DATE + SRV/PLACE/RTN DTLS) ;
  Unavailable END
  Loan or Service ALT (IF available for hire)
  Available SEQ
    HIRE ;
    Read (TGM DATE + SRV/PLACE/RTN DTLS) ;
    RETURN ;
    Read (TGM DATE + SRV/PLACE/RTN DTLS) ;
  Available END
  Loan or Service END
Useful Life END
REPLACE ;
Vehicle END.

```

It is assumed that the handling of the VEH SV is looked after within the related VEHICLE actions and does not, therefore, need to be detailed at this stage of the design.

The Read (TGM + ...) operation is assumed to be a rough merge of the four datastreams specified.

```

Entity CUSTOMER
Read (CUS INP) ;
Customer ITER (While still a customer)
Action SEL (IF place)
  PLACE ;
  Write (CUST HIRE) ;
  Read (CUS INP) ;
  Action ALT (IF amendment)
    AMEND ;
    Write (CUS HIRE) ;
    Read (CUS INP) ;
    Action ALT (IF cancellation)
      CANCEL ;
      Write (CUS HIRE) ;
      Read (CUS INP) ;
      Action ALT (IF hiring)
        HIRE ;
        Write (CUS HIRE) ;
        Read (CUS INP) ;
        Action ALT (IF returning)
          RETURN ;
          Write (CUS HIRE) ;
          Read (CUS INP) ;
          Action END
        Customer END.

```

```

Entity HIRING
Hiring SEQ
Read (CUS HIRE) ;
GetSV (VEH SV) ;
PLACE ;
Write (PLACE DTLS) ;
Read (CUS HIRE) ;
Possible Amendments ITER (While still amending)
  AMEND ;
  Write (RTN DTLS) ;
  Read (CUS HIRE) ;
  Possible Amendments END
  Cancel or Continue SEL (IF hiring cancelled)
    CANCEL ;
    Write (RTN DTLS) ;
    Read (CUS HIRE) ;
    Cancel or Continue ALT (IF proceeding with hiring)
      Continue SEQ
    HIRE ;
    Write (RTN DTLS) ;
    Write (HIRE DTLS) ;
    Read (CUS HIRE) ;
    { Hire details }
    { Customer hire information }
  RETURN ;
  Write (RTN DTLS) ;
  Read (CUS HIRE) ;
  Continue END
  Cancel or Continue END
Hiring END.

```

Within the HIRING process, the PLACE action needs to inspect the VEH SV to determine vehicle availability. Therefore, this needs to use GetSV operations; one is included before the PLACE action to indicate the state vector's use. The rest of the state vector processing will be contained within the PLACE action.

At the HIRE action, there is a Write (HIRE DTLS) operation. This will be some form of docket or hire form that the customer can retain which would detail the hiring. A copy may also be produced for the administrative staff if the existing filing system is still required (perhaps as a form of off-line backup).

For the two function processes included on the elaborated SSD, we would need to specify some form of structure text as a basis for the program specification.

To complete the texts fully, we require more discussion with the management of Poly Fleet Hire to obtain a more detailed set of requirements. In the absence of such information, an outline sketch is given. An overview of the form of the texts for the reports is thus shown below.

```

Management Report ITER (While system in operation)
  Read (Trigger) ;
  GetSV (VEH SV) ;
  Report Body ITER (While still vehicles to process)
    Process report output line ;
    Report Body END
  Management Report END.

```

The nature of the trigger would differ for the two reports. For the SERVICE & REPLACE function, it would comprise a Read (TGM WEEK) operation while the SERV/REP POLICY function would use a Read (MAN INPT) operation.

We have assumed here that only vehicle state vectors are required. However, access to other processes' state vectors could be easily added if required.

The actual processing involved in the Process Report Output Line operation needs to be discussed with the user, but at least some idea of the function's structure is given here.

Implementation Step

The System Implementation Diagram is given as figure 6

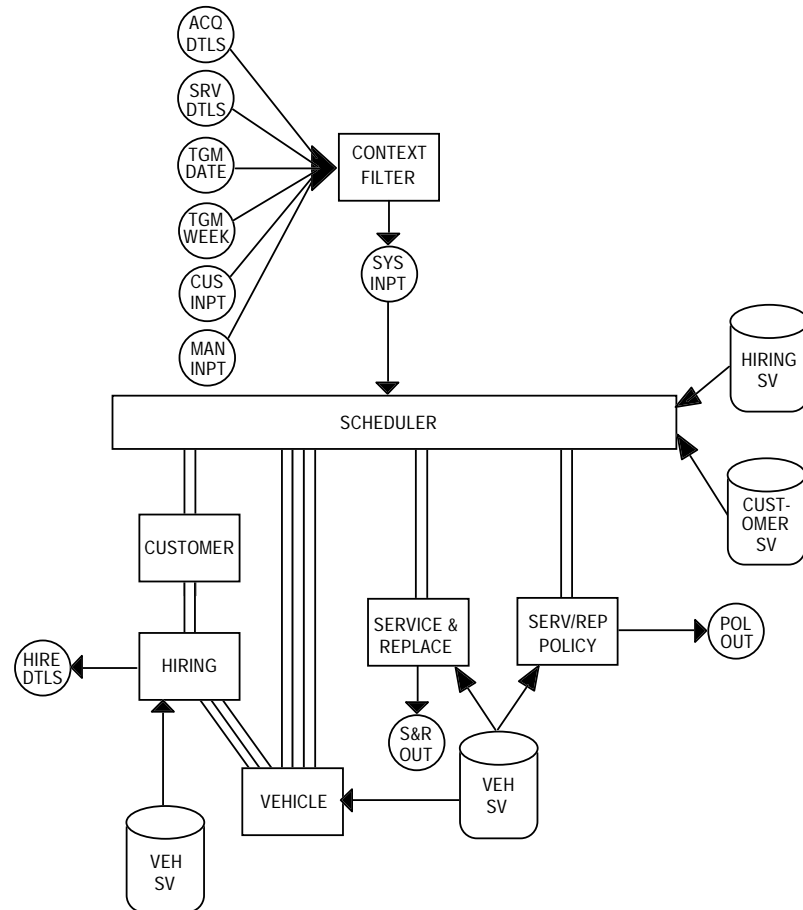


Figure 6 : SID

It is assumed that some form of filter can be used to accept the user input, validate it and pass it on to the scheduling process (and thence to the individual processes as required). The filter will also provide the user interface. This is shown on the SID as CONTEXT FILTER and receives all the system input in a rough merge as the order of arrival of data cannot be predicted. The CONTEXT FILTER then passes the messages to the SCHEDULER via the SYS INP datastream.

It is also assumed that the SCHEDULER process can look after the two state vector files for HIRING and CUSTOMER. It does not have to look after the VEHICLE's state vector file as it is used heavily by the rest of the system.

The SCHEDULER process included in the SID has the responsibility of scheduling the system's various processes and handling the two state vector files for the HIRING and CUSTOMER processes. The input from the outside world is brought by the CONTEXT FILTER process. The outline structure text for the SCHEDULER process is now given.

```
Scheduler SEQ
  Read (SYS INPT) ;
  Scheduler Body ITER (While still require input)
    Model Process SEL (IF for CUSTOMER)
      Customer Input ;
      GetSV (CUST SV) ;
      Call CUSTOMER (argument list) ;
      StoreSV (CUST SV) ;
    Model Process ALT (IF for VEHICLE)
      Vehicle Input ;
      Call VEHICLE (argument list) ;
    Model Process ALT (IF for SERVICE & REPLACE)
      Service & Replace Input ;
      Call SERVICE & REPLACE (argument list) ;
    Model Process ALT (IF for SERV/REP POLICY)
      Serv/Rep Policy Input ;
      Call SERV/REP POLICY (argument list);
    Model Process END
  Read (SYS INPT) ;
  Scheduler Body END
Scheduler END.
```

It can be seen from the structure text that the VEHICLE process handles its own state vector file. It is assumed that further input filtering and interfacing occurs in the '... Input' operation for each input type.

The argument list for each model process and function process would be dependent on the individual process needs (keys and fields to pass to the process etc.).

From here, the rest of the implementation process would continue. This would include state vector separation, and the creation of more detailed structure texts to be used as robust program specifications. Work would then proceed to coding and unit test. These tasks lie outside the scope of this exercise.