

Requirements Analysis Document:

Maintenance

JAMES Project

15-413 Software Engineering

Fall 1997

Carnegie Mellon University

Pittsburgh, PA 15213

Revision History:

Version 1.2 - Last Update: December 1, 1997

Preface:

This document addresses the requirements of the JAMES system. The intended audience for this document are the designers and the clients of the project.

Target Audience:

Client, Developers

JAMES Members:

Gordon Cheng, Li-Lun Cheng, Christopher Chiappa, Arjun Cholkar, Uhyon Chung, Aveek Datta,

John Doe, Phillip Ezolt, Eric Farnig, William Ferry, Sang Won Ham, Kevin Hamlen, Pradip Hari, Yenni Kwek, Tze Bin Loh, Alexander Lozupone, Christopher Lumb, Vincent Mak, Darren Mauro, Hoda Moustapha, Venkatesh Natarajan, Stan Pavlik, Michael Poole, Bob Poydence, Kalyana Prattipati, Luis Rico-Gutierrez, Michael Samuel, Michael Scheinholtz, Joel Slovacek, Ann Sluzhevsky, Marc Snyder, Steve Sprang, Paul Stadler, Herbert Stiel, Patrick Toole, Idan Waisman, Aaron Wald, Andrew Wang, Zhongtao Wang, Nathaniel Woods, Jaewoo You, Bin Zhou.

1.0 General Goals

General goals for the project are not discussed in this team specific document.

2.0 Current System

The current system is documented very well in the Mercedes-Benz CarCard manual. A copy of the manual can be found under http://cascade1.se.cs.cmu.edu/JAMES/pre45end/CCV1_ENG.doc. This should be considered a small description of the system; reading this manual will give you a more complete view of the current system.



Figure 1: Screen shot of the CarCard Application

Figure 1 shows a sample screen from the CarCard application. This application is a Microsoft Windows program with the standard toolbars, pull-down menus and buttons. In this screen shot you see the main window along with the File pull-down menu being opened. The program requires a plugged in card-reader to operate, which goes

through the COM ports of the user's PC. It also reads data from a card that is inserted into the card reader.

The user interface above is very simple and what Windows users are used to: pull-down menus bring up all necessary options and most choices are made by mouse clicks while data is entered through the keyboard. Major options are allocated Function keys and space in the toolbar for easy access.

The basis behind this application is to allow the user to view information. There are also facilities for authorized users to update information, if they have the right access. (Options that can not be updated by an unauthorized user are grayed out.) After all this is done data can be written back to the card.

The screenshot shows a software window titled "Mercedes-Benz CarCard - ES-CZ 6675 - Meier". The window contains a "Vehicle data" section with a car icon and a "CC" icon. The data is organized into several sections:


- Manufacturer data:** Chassis number: WDB2021211A4046705; Model: C220D; Country: 513; CarCard number: 6015570000000091; Commission No.: 0651313697.
- Due dates:** TUEV (7/10/99) and ASU (7/10/99). There are also empty fields for due dates and a "miles" field.
- Text:** A large empty text area.
- Registration and Technical Data:** Registration No.: ES-CZ 6675; Date 1st Reg.: 7/10/97; Delivery date: 7/10/97; Engine No.: 60491000075158; Gearbox No.: 71741607010575; Tyres: UNIROYAL; Headlights: VALEO; Front axle; Rear axle; Trim code: 041; Trim colour; Paint 1: 721; Paint 2; Hardtop No.; Hazard class.

At the bottom, there are "Cancel" and "OK" buttons, and two buttons labeled "Options" and "Tech. measures".

Figure 2: Vehicle Data

Figure 2 shows what happens when the user presses F4 or clicks on the button marked with the vehicle. It displays basic information about the vehicle useful to both the mechanic and the driver.

Mercedes-Benz CarCard - ES-CZ 6675 - Meier


 *Driver's name and address details*

<u>T</u> itle	Herr	<u>M</u> aintenance dealer	78765
<u>S</u> urname	Meier	<u>S</u> upplying dealer	66765
<u>F</u> irst name	Franz	<u>C</u> ustomer information	Kunde hat die Absicht eine S-Klasse zu kaufen !!
<u>A</u> ddress	Uhlandweg 7		
<u>P</u> O box			
<u>P</u> ost code	73246		
<u>T</u> own	Aichwald		
<u>T</u> el. <u>B</u> usiness	07365 6575		
<u>T</u> el. <u>H</u> ome	07365 4533		
<u>F</u> ax	07365 3221		
<u>V</u> AT Reg. No			

Figure 3: Driver's Name and Address Details

Figure 3 shows what happens when the user presses F5 or clicks on the button marked with the two people. It displays contact information for the driver.

Mercedes-Benz CarCard - ES-CZ 6675 - Meier

 *Contract information (1)*


Contract type:	Leasing	Contract partner:	Mercedes-Benz Finanz
Contract No.:	676766	Address:	Epplestraße 225
Start date:	7/10/97	PO box:	
End date:		Postcode:	70322
End mileage:		Town:	Stuttgart
Scope:		Country:	D
		Issued by:	Auto Lange
		Contact person:	H.Braun
		Telephone:	0711 676767
		Fax:	0711 464546

Switch Changes Cancel OK

Figure 4: Contract Information (1)

Figure 4 shows what happens when the user presses F6 or clicks on the button marked with someone signing a piece of paper. It displays the contract between the driver and a Mercedes dealership/ representative so mechanics will know who will pay for service covered under warranty, etc.

Mercedes-Benz CarCard - ES-CZ 6675 - Meier

 *Warranty information*

	Months	End date
<input checked="" type="checkbox"/> <u>T</u> ouring guarantee	24	7/9/99
<input checked="" type="checkbox"/> <u>M</u> obility service	12	7/9/98
<input checked="" type="checkbox"/> <u>W</u> orks guarantee	12	7/9/98

Extended warranty

Changes Cancel OK

Figure 5: Warranty Information

Figure 5 shows what happens when the user presses F7 or clicks on the button marked with someone holding a sheet paper. This shows the details of what is under warranty for this car.

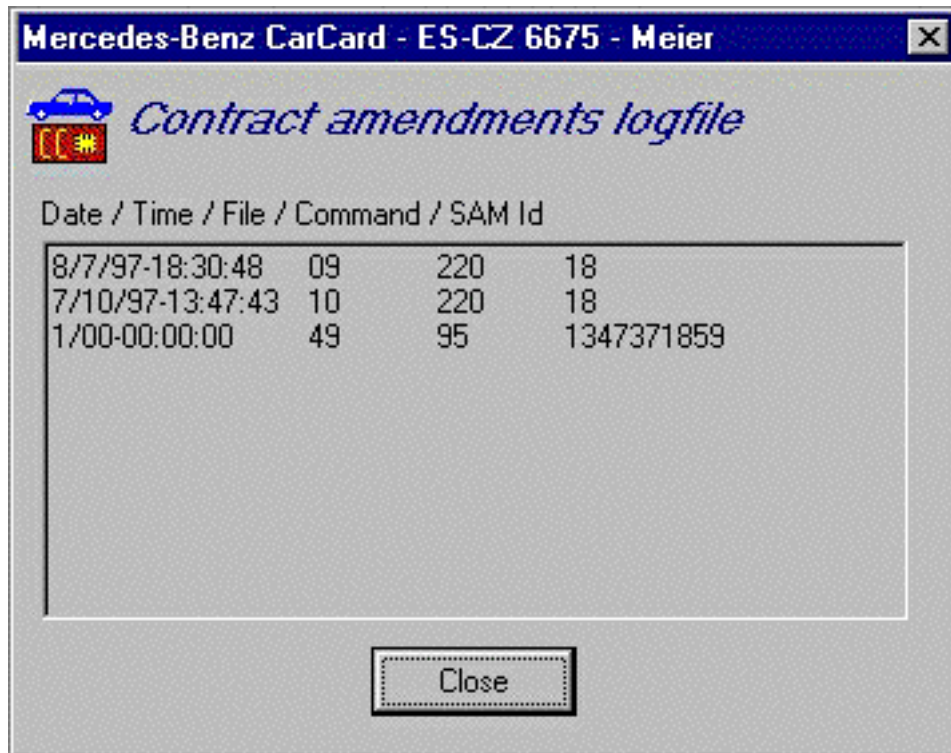


Figure 6: Contract Amendments Logfile

Figure 6 shows what happens when the user clicks on Changes after hitting F7. This lists amendments to the warranty using special codes established by Mercedes.

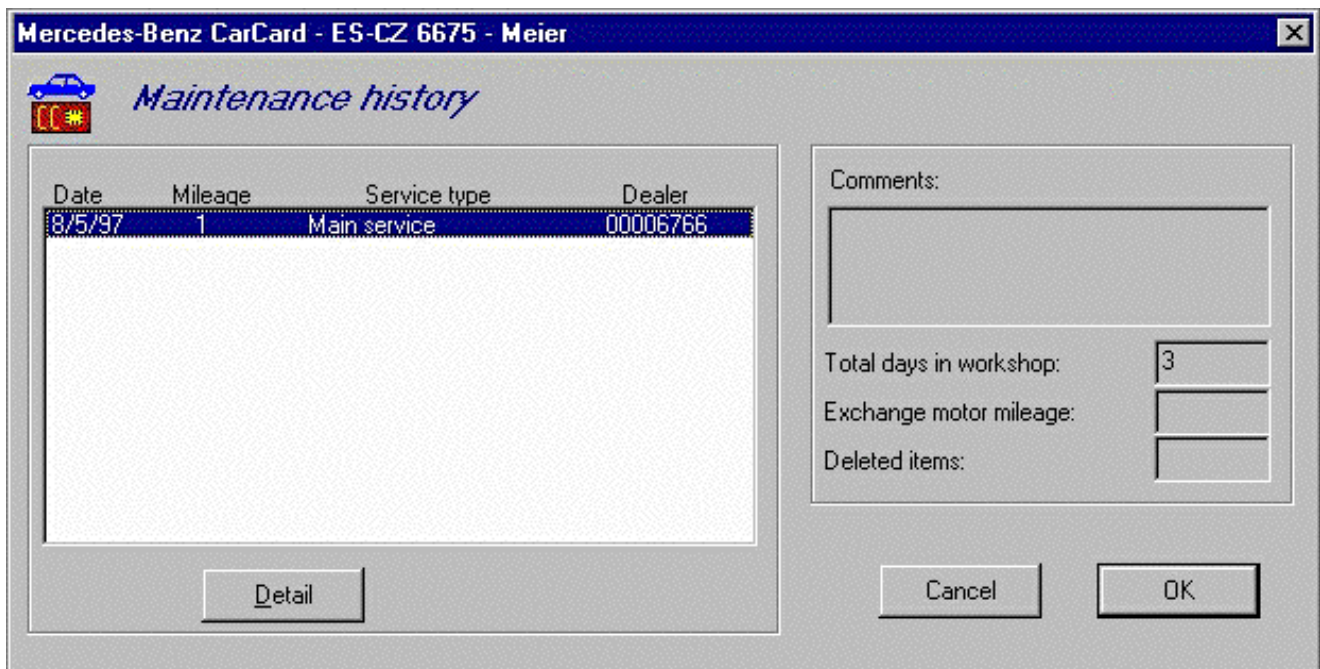


Figure 7: Maintenance History

Figure 7 shows what happens when the user presses F8 or clicks on the button marked with an oil can. This shows a list of maintenance events that the car has gone through.

Mercedes-Benz CarCard - ES-CZ 6675 - Meier

Maintenance job details

Dealer Country: 513 Dealer No: 6766 Order number: 1 Service advisor: 1	Repair Date: 8/5/97 Mileage: 1 Days in w/shop: 1 Serv. Type: Main service	Amendments Date: Amended by: Delete code:
---	--	---

Operations

00330001 * Delwechsel (falsche Oelorte)

'G' = Warranty, 'K' = Goodwill, '*' = manual description

Cancel OK

Figure 8: Maintenance Job Details

Figure 8 shows what happens when the user clicks on Details after hitting F8. This shows the details of one event. Notice how most fields are grayed out; as a user you are not allowed to change these fields.

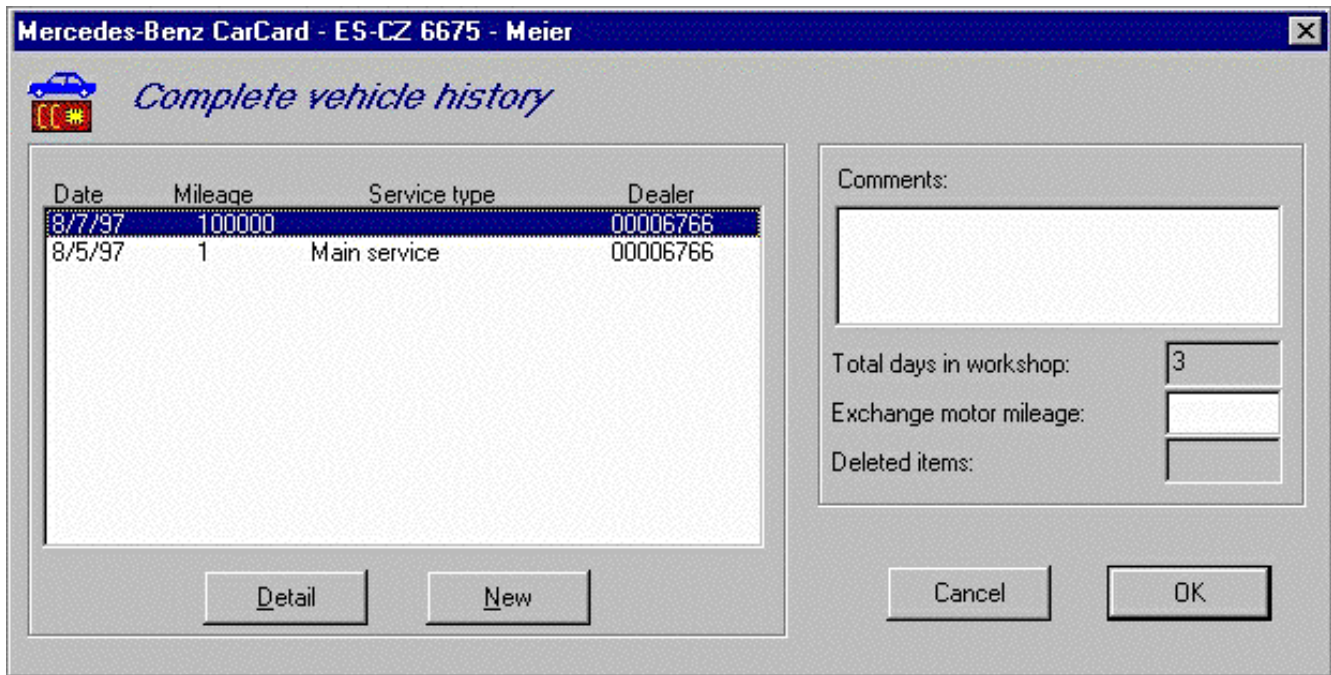



Figure 9: Complete Vehicle History

Figure 9 shows what happens when the user presses F9 or clicks on the button marked with a vehicle with an X through it. This shows the complete vehicle history, including maintenance events.

Mercedes-Benz CarCard - ES-CZ 6675 - Meier

 *Amend workshop job details*

Dealer		Repair		Amendments	
Country:	513	Date:	8/7/97	Date:	
Dealer No:	6766	Mileage:	100000	Amended by:	
Order number:	46756	Days in w/shop:	2	Delete code:	
Service advisor:	1	Serv. Type			

Operations

00016676 * Austausch der Auspuffanlage

'G' = Warranty, 'K' = Goodwill, '*' = manual description

Insert Delete

Cancel OK

Figure 10: Amend Workshop Job Details

Figure 10 shows what happens when the user request details on a particular event in the car history (F9 and then Details button). This is similar to Figure 7 in functionality and use.

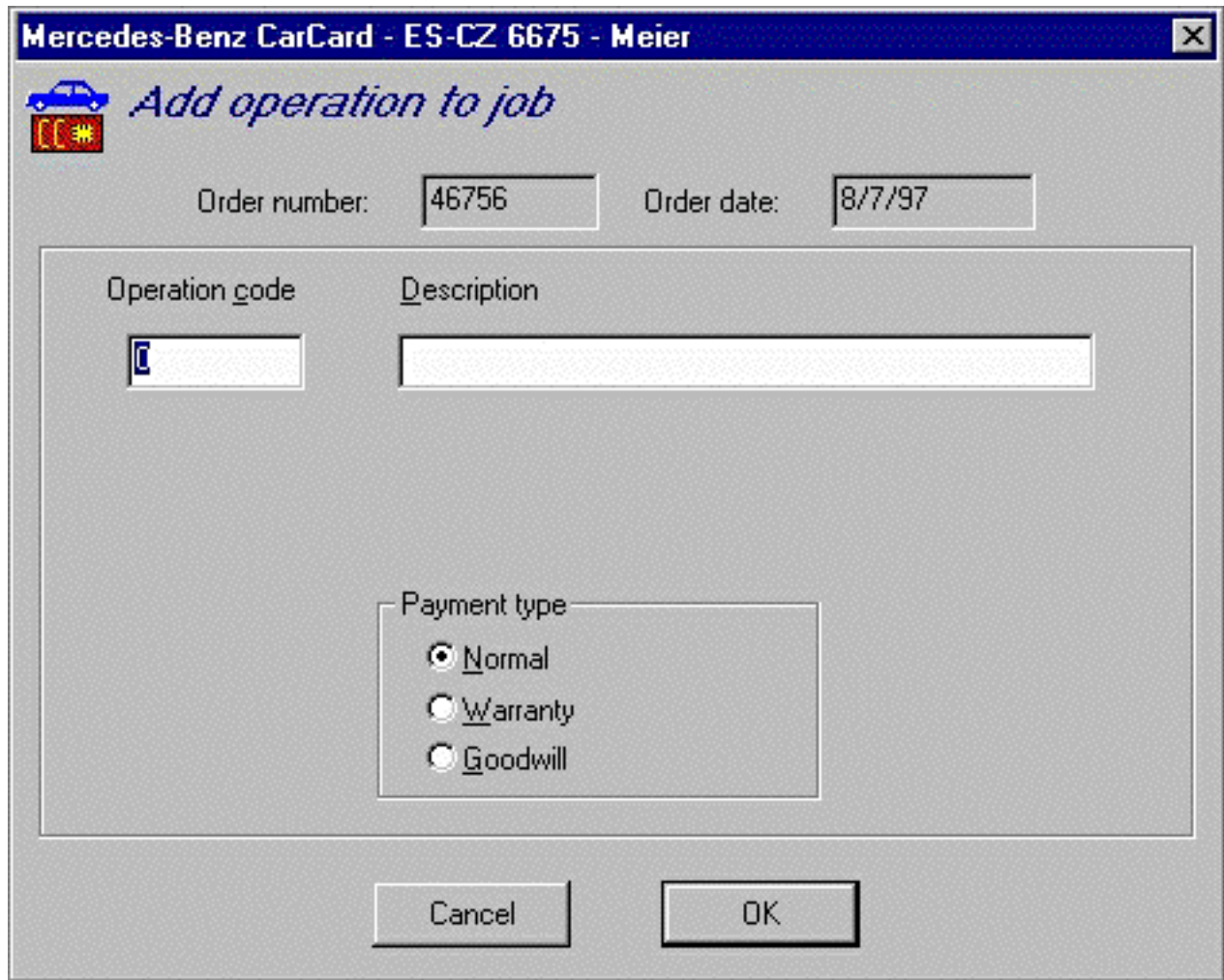


Figure 11: Add Operation to Job

Figure 11 shows what happens when the user chooses to Insert an operation to a given maintenance event (F9, then Details, the Insert). This allows the user to insert a new operation to a given event (such as an oil change, etc.) and how they paid for it.

In summary the current system provides an intuitive, simple way to track basic car and driver information along with the maintenance history of the car. It does not provide recommendations at this point. The program itself is not portable, it only works on the Microsoft Windows platform. One of the nonfunctional requirements of the JAMES project is to create a new portable Maintenance Assistant that will work on many different platforms.

3.0 Proposed System

3.1 Overview

The proposed system builds on the current system by adding access to bonus points, a web interface, and a recommendation system, by which users can get recommendations on required and optional maintenance, such as replacing faulty seat belts in a given model or an optional tune-up. In addition the proposed system will be written in Java to be portable and accessible through any Java enhanced Web-browser like Netscape.

This system is a re-engineering effort in which we examine the current system and add functionality and portability. We would like to retain many of the features of the current system in addition to the functional and nonfunctional requirements described in sections 3.2 and 3.3.

3.2 Functional Requirements

Note: The card used by the owner will herein be referred to as the *SmartCard*. This card that we are currently using is a CyberFlex JavaCard. That may change in the future though. Check the System Design Documentation for more updated information.

The JAMES Maintenance Assistant should:

- Identify customers and car they drive based on information stored on the SmartCard
- Provide access to features only to authenticated users
- Recommend maintenance tasks based on data about the vehicle, the last recorded date of service and the individual driving behavior of the customer
- Provide the service history of the vehicle to authorized users
- Allow dealers to add information about newly performed maintenance tasks to the vehicle's history
- Inform drivers and dealers about service pertaining to all vehicles of a particular model and year
- Bonus Point System functionality may be added by an additional Bonus Assistant

3.3 Nonfunctional Requirements

3.3.1 User Interface and Human Factors

This system will be used by two main groups of people: First it can be used by an authorized representative of the owner of the car which could be the driver, friend of the driver, fleet manager or someone else. Second, it can be used by the mechanic or some other representative of the garage, such as a manager. The complete user definitions are described in section 3.5.2.1 [User Model](#).

For simplicity, in the rest of this document *driver* will refer to the driver or any authorized representative of the driver, while *mechanic* will refer to any authorized representative of the garage.

The central database, as used in this document is physically several databases, including several legacy servers. These servers will be distributed across separate sites on the Inter/IntraNet. It will be used for long term storage of the maintenance history and possibly other information. See the System Design Document for additional information on databases.

The assistant will exist on several different locations. There will be a home assistant for the owner to use at home, an assistant on the vehicle for the driver to use, as well as an assistant in the dealer's garage for him to use. For the owner/driver, the assistant should be simple and easy-to-learn. This is especially important for the home assistant, as users will have only occasional interaction with them. The mechanic will have more frequent interaction with the assistant, so the interface in the dealer garage should enable mechanics to enter transactions quickly and easily. In both cases, users should be protected from making significant errors, as these records are maintained for the life of the car.

In terms of hardware, the driver will need a computer and a card reader in his home to download the maintenance applet and view the maintenance history. Software will be needed to interface with the card reader. The mechanic will also need a similar computer and card reader in the garage. In the vehicle, the maintenance system expects access to a touch-sensitive screen that can display our assistant and receive input from the driver.

A final design goal for the user interface is to achieve the same elegant simplicity of interaction for which Mercedes-Benz vehicles are known. The interface should not appear complicated, nor require extensive explanation.

3.3.2 Documentation

This system is designed to be very simple from the driver's perspective, but the user will probably need some training on how to download the maintenance applet onto their Java Card and on how to use the web to browse their maintenance history. This will be covered in a short user manual or an online walkthrough/tutorial.

The mechanic on the other hand will need more training on interfacing with the system in the garage and how to interpret results and errors. This information will also be detailed in the user manual.

3.3.3 Hardware Consideration

The driver's and mechanic's hardware consists of basically a computer and a card reader. The computer must be compatible with the card reader and have some sort of network intranet or extranet connection so it may connect to the central databases. In the car there will also be another card reader.

The Maintenance Assistant will require some sort of external database, the specifics of which are to be described in the System Design Document. This database system will probably be built alongside the existing legacy systems.

3.3.4 Performance Characteristics

There are no inherent requirements for the speed of the system suffice to say that it be "reasonable." Reasonable could mean that the system responds in a manner that is not annoying to the user. This usually requires that the system react to requests from the user within 2 seconds. This does not necessarily mean that the system completes the request in that time, just that it notifies the user that it is performing the request. Most requests should be fulfilled within a few seconds. This is similar to the amount of time that a user waits for a web page to load on a browser so it's something that the user will be familiar and comfortable with.

Capacity constraints on the smart card are a total of 2k of data, though a somewhat larger version debuted within a week of this writing. Check the System Design Document for updated information on the card. The external databases for the dealers can be quite large. Its exact size and function will depend on each individual dealer. Since there are no figures on how many users this system will have, it is impossible to estimate the size requirements. It would be possible to imagine that a dealer could potentially have several hundred or thousands of customers. Each customer could have a few megabytes of worth of information so the system could be several gigabytes large which is fairly reasonable for a modern database.

3.3.5 Error Handling and Extreme Conditions

There is a Maintenance Transaction Manager that will handle all of the requests from the user. See the System Design Document and the Object Design Document for more details on the Maintenance Transaction Manager. The user interface to the Transaction Manager will only allow the user to perform actions that it considers legal. Any illegal action, such as requests from an unauthorized user, will be grayed out by the user interface so that an illegal request cannot be made. With this mechanism in place, no illegal actions or errors can occur as a result of user actions.

Problems with the hardware itself can cause errors though in extreme conditions. The Maintenance Transaction Manager will report any hardware failures on the database, the PC, the card, reader or any communications point in between. It will simply note that there is a problem and not allow the user to interact with the problem parts. An example would be notifying the user that it cannot write data to the card because there is a hardware problem with the card or that the card is full.

3.3.6 System Interfacing

This system is primarily standalone. It interfaces with the authentication subsystem to authenticate the driver and dealer. There is a Bonus Points system to reward the driver for maintenance requests. This will be implemented in

the future. See the System Design Document for further information. There will be an Event Notification System and a Name Service that will allow the Maintenance Assistant to communicate with its user interface. It will also allow it to communicate with any other Assistant or system that uses the Event Notification and Name Service. They will communicate using standard Java and CORBA. CORBA was decided over RMI because it allows the system to talk to other systems there are not necessarily written in Java.

3.3.7 Quality Issues

The database system must be reliable in terms of availability and correctness of information. The Maintenance Assistant's primary feature is to keep a history of the maintenance performed on the vehicle. For it to be of any use, the information must always be correct, otherwise it is useless. To increase its reliability, the database should be backed up regularly but this will be left to the discretion of the dealer and driver.

3.3.8 System Modifications

One of the goals of the JAMES project is to design an extensible, scalable and platform independent system. A set of API's have been designed for the Maintenance Assistant and its interface to external databases. See the Object Design Document for more details. These API's can be used to write new programs to allow other JAMES assistants to interact with the Maintenance Assistant and with an external database. The external database can be of any type (relational, object oriented, Ö) because that is implementation specific. As long as it follows the API's, it will be compatible with the rest of the system.

3.3.9 Physical Environment

There is a card reader and onboard computer in the vehicle to allow the driver to use the Maintenance Assistant from within the vehicle. Accessing the history on the card will not be a problem although accessing an external database would be more difficult.

The owner of the vehicle can run the Maintenance Assistant from his home PC, with a card reader attached to it. The owner can maintain his own personal database on his home PC.

The dealer or mechanic will have a computer in the garage that is connected to a card reader and possibly a database. The database can also be used to store dealer specific data.

3.3.10 Security Issues

Security is a major issue in authenticating mechanics, and a relatively minor issue for authenticating drivers, since drivers will have read-only access to information that is not explicitly confidential. Access rights to the system and information will be determined once the user is authenticated. Reasonable security measures should be taken to ensure that communication between client computers and the central databases are secure.

3.3.11 Resource Issues

Ideally the database system should be backed up continuously. At least once a day at the central level should be considered a bare minimum in case of catastrophic failure, to minimize data loss. Mercedes or contracted support staff should maintain the central databases and ensure their integrity and availability at the dealer sites.

3.3.12 Synchronization Issues

The model supported the Maintenance Assistant recognizes two "locations" for data: dealer databases and the SmartCard. When a user goes to a dealer for service, the dealer first synchronizes his personal databases with that of the SmartCard, adding "new" records as identified by date and time. The dealer then uses his synchronized database for viewing information about the vehicle. The system should support multiple dealer databases of various implementations. In addition, a dealer may store proprietary information in his personal database in addition to standard information stored on the card.

When new items are added to the Maintenance History of a vehicle, the new item will be stored in both databases. The SmartCard will hold as many maintenance items as space permits using a first-in-first-out (FIFO) methodology for disposing of old items.

3.4 Constraints

The system will be developed with Java (JDK 1.1.4) with CORBA using VisiBroker. Object and case models will be constructed using the CASE tool Rational Rose 4.0. Source control will be handled using Perforce. See Section 9. Issues of the System Design Document for further discussion. There will be an interface to the legacy system, depending on the access granted to us by Mercedes.

3.5 System Model

3.5.1 Scenarios

Scenario 1: Take Car for Service

Mr. Smith is at the dealer garage. He gives his card to the dealer. From the card, the dealer gathers information about the user and diagnostic data about the vehicle. The dealer also gathers user, address, contract, diagnostic, model, and other information from Mercedes-Benz databases. Based on information provided to dealers by Mercedes-Benz, the dealer generates a list of recommendations for maintenance on the vehicle. For the M-class vehicle that Mr. Smith owns, the seat belts have been recalled for safety reasons. The dealer downloads and checks the maintenance history of the vehicle. This replacement has not yet been made, so the dealer offers to replace them free of charge. The garage mechanic installs the seat belts and adds the record of this service to the maintenance history log for the vehicle.

Scenario 2: User checks Maintenance Information from Home

Mr. Smith, the owner of a Mercedes-Benz M-class vehicle, is unable to remember the date of his last oil change. He decides to review the maintenance record of his car. He inserts his card into his home PC card reader and opens the James Maintenance Assistant. The Assistant fetches the most recent maintenance history from the card and displays it on the user's home PC. Using the information about the vehicle provided by the card, the PC application checks against information provided by Mercedes-Benz on the application and generates a list of maintenance recommendations for the vehicle. Mr. Smith sees that his last oil change was only 1000 miles ago, and an oil change is not listed as a recommended maintenance. The Assistant then allows Mr. Smith to connect to a Mercedes Benz site and check for any updates and announcements for his car. He sees that the seat belts for his M-class have been recalled and that they can be replaced by his local dealer free of charge. He decides to go to the garage to have this service performed.

Scenario 3: User checks Diagnostic Information from Vehicle

Mr. Smith is in his M-class vehicle and wants to view the diagnostic information about his vehicle. On the screen for the James Maintenance Assistant, he presses a button to view this information. The information is displayed on the screen.

3.5.2 Use Case Models

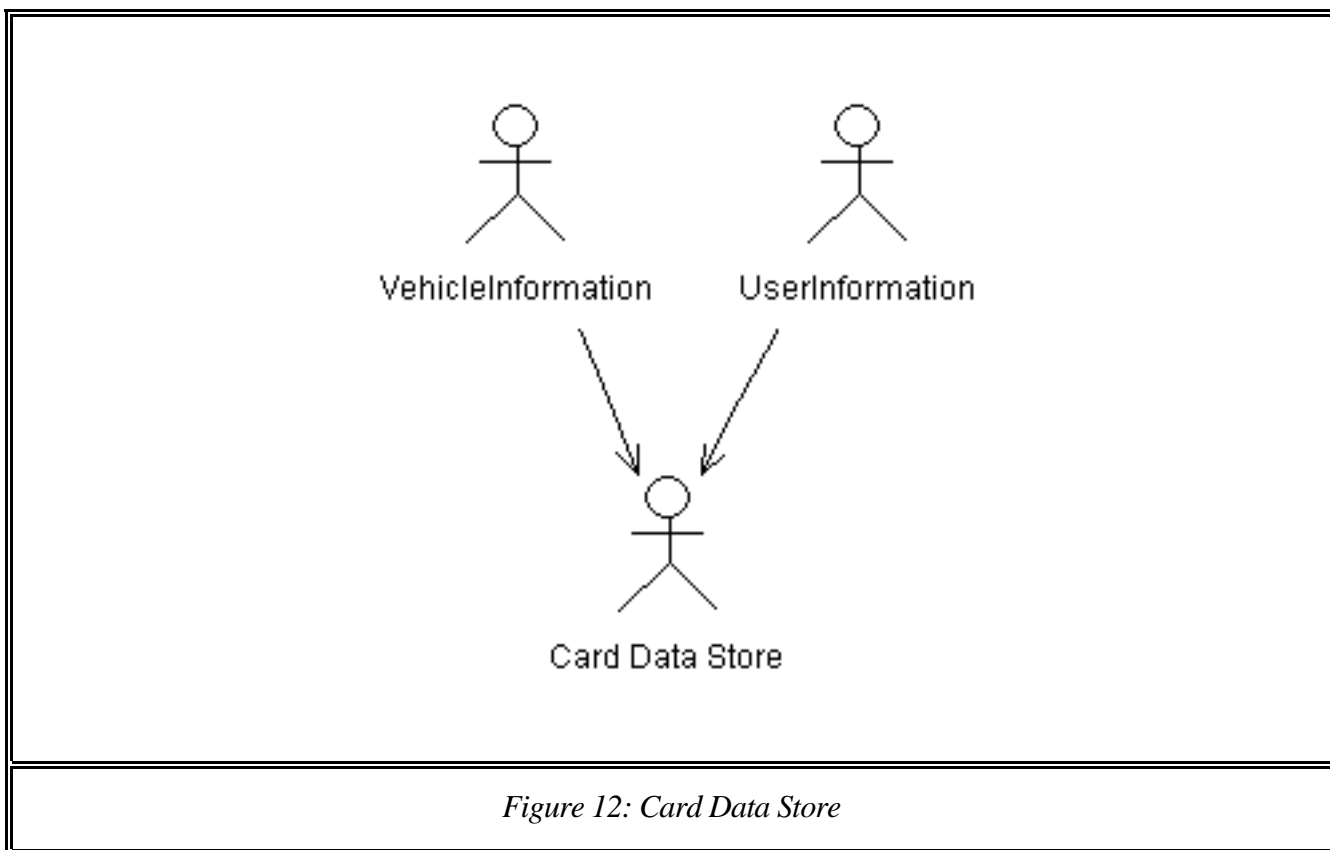
3.5.2.1 Actors

Card Data Store

This represents information that is stored on the SmartCard. This may consist of several types of information including:

UserInformation

VehicleInformation



VehicleInformation

A number of data from the car will be needed to compute a user and car specific service proposal.

These data are:

- Motordrehzahl (Nm),
- Motorlast
- engine temperature
- time interval between the last service and now
- oil amount sensor

UserInformation

This is the user specific information present on the card.

Recommendation Service

This actor computes a service proposal depending on the current state of the car.

The computation is based on the data coming out from the MaintenanceTransactionManager.

The MaintenanceTransactionManager uses data from the CarInfoService, data from the card (delivered through the GatherCardInformation use case) and the Maintenance History Service.

CarInfo Service

This service will provide additional non-standard information about a vehicle. For example, if the car were in an accident, the mechanic might need a wiring diagram for a damaged area of the vehicle.

Maintenance History Service

The ManageMaintenanceTransaction use case needs information related to the car (type or model, year of building, car identifier), and the car related service history (mile dif, time dif, car and aggregation type, numbers of activities, service activity or additional activity).

Also the number of all services 1, 2, ...n until now and their related data (like above).

The complete documentation of the service for release in the check book.

A service paper which can be delivered to the customer.

Dealer Representative

The car service support center, a dealer shop or a garage will be able to make a maintenance service for a customer. The Dealer Representative does the service and has access to the specific information bases and their data depending on his role. The access rights on the maintenance data related to the dealer representative role will be described within a access matrix. Note that this is an important special case of the MaintenanceUser which is described below.

Maintenance User

A user of the Maintenance system wants to interact with the system. This user can be a:

- Company supplier, who want to activate the maintenance assistant for a customers car.
- A carpool driver, who want to activate the maintenance assistant, but is not the owner of the car
- A fleet manager, perhaps want to activate the maintenance assistant for one or a number of cars.
- A normal user/driver, normally owner of the car, who may want to activate his/her maintenance assistant for his/her car
- Dealer Representative as mentioned above.

User Model

This diagram provides a description of the various user roles within the system.

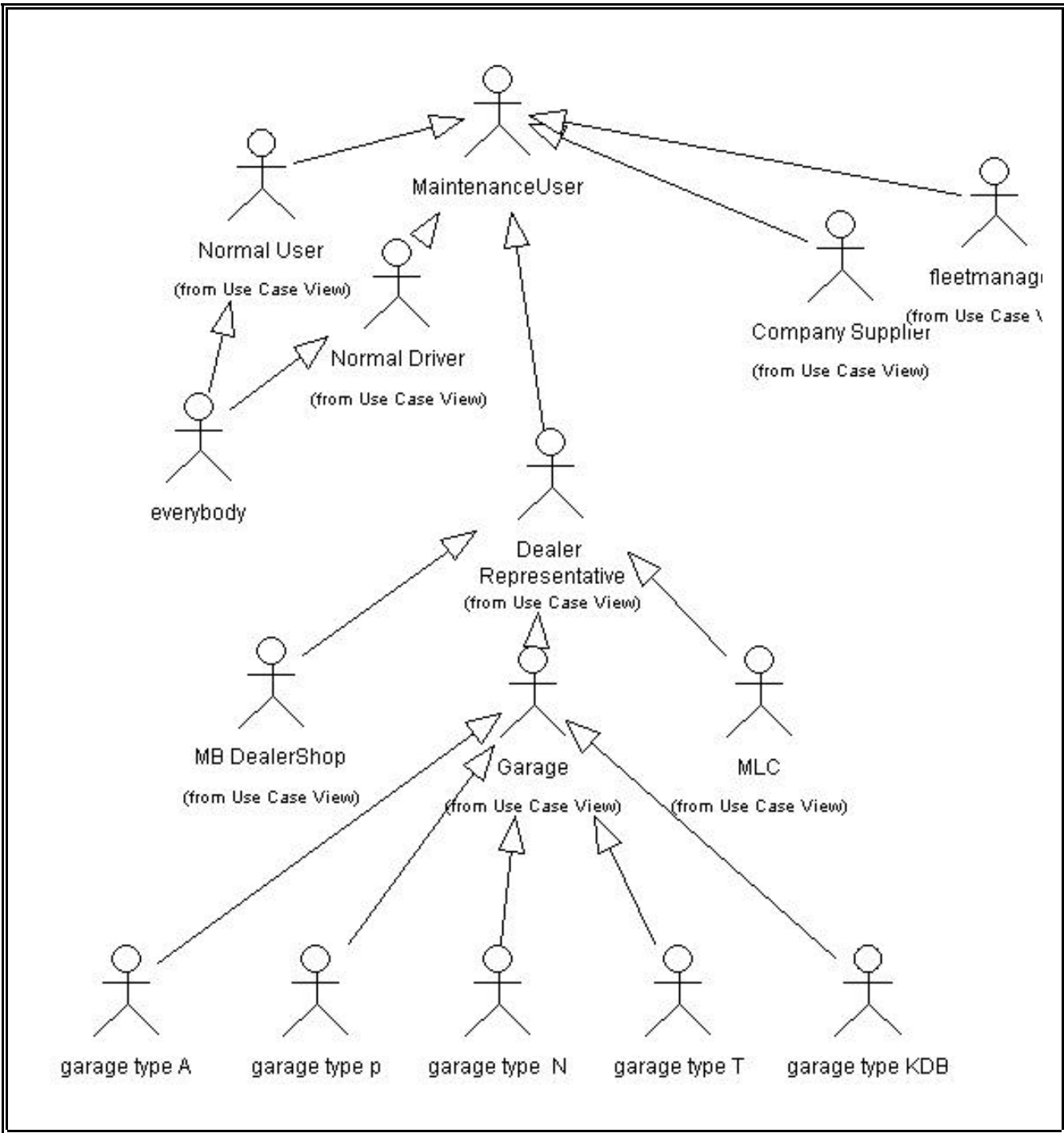


Figure 13: User Model

Normal User

User is a not known by the bonus system of a company

Normal Driver

User is known by the maintenance system of the company and want to have maintenance support for his car.

Dealer Representative

The car service support center, a dealer shop or a garage will be able to perform a maintenance service for a customer.

The Dealer Representative completes the service and has access to the specific information bases and their data depending on his role.

The access rights on the maintenance data related to the dealer representative role will be described within a access matrix.

MB DealerShop

This type of user is a Mercedes Benz dealer, a KDB or SVB.

This user type can read (R):

radio data, car data, history, address data, contract data, dealer file and MLC file

This user type can write (W):

radio data, car data, history, address data, contract data, dealer file

MLC

This type of user is a Mercedes Benz MLC or SVB.

This user type can read (R):

radio data, car data, history, address data, contract data, dealer file and MLC file

This user type can write (W):

radio data, car data, history, address data, contract data, dealer file and MLC file

Garage

A garage consists of one of the following garage types: A, P, N, T, or KDB.

Garage Type A

This type of user is a non Mercedes Benz garage or the AAA (in europe called ADAC).

This user type can read (R):

radio data, car data, history

Garage Type P

This type of user is a non-Mercedes Benz garage or the police

This user type can read (R):

radio data, car data, history, address data, contract data, dealer file and MLC file

Garage Type N

This type of user is a non common Mercedes Benz garage

This user type can read (R):

radio data, car data, history

This user type can write (W):

history

Garage Type T

This type of user is a tire dealer with a Mercedes Benz service contract and 24 h service.

This user type can read (R):

radio data, car data, history, address data, contract data

This user type can write (W):

history

Garage Type KDB

This type of user is a Mercedes Benz service consultant, called KDB

This user type can read (R):

radio data, car data, history, address data, contract data, dealer file and MLC file

This user type can write (W):

radio data, car data, history, address data,, dealer file

Company Supplier

The user is a employee of a dealer or a garage which offers maintenance support to their customers related to the maintenance assistant.

Fleetmanager

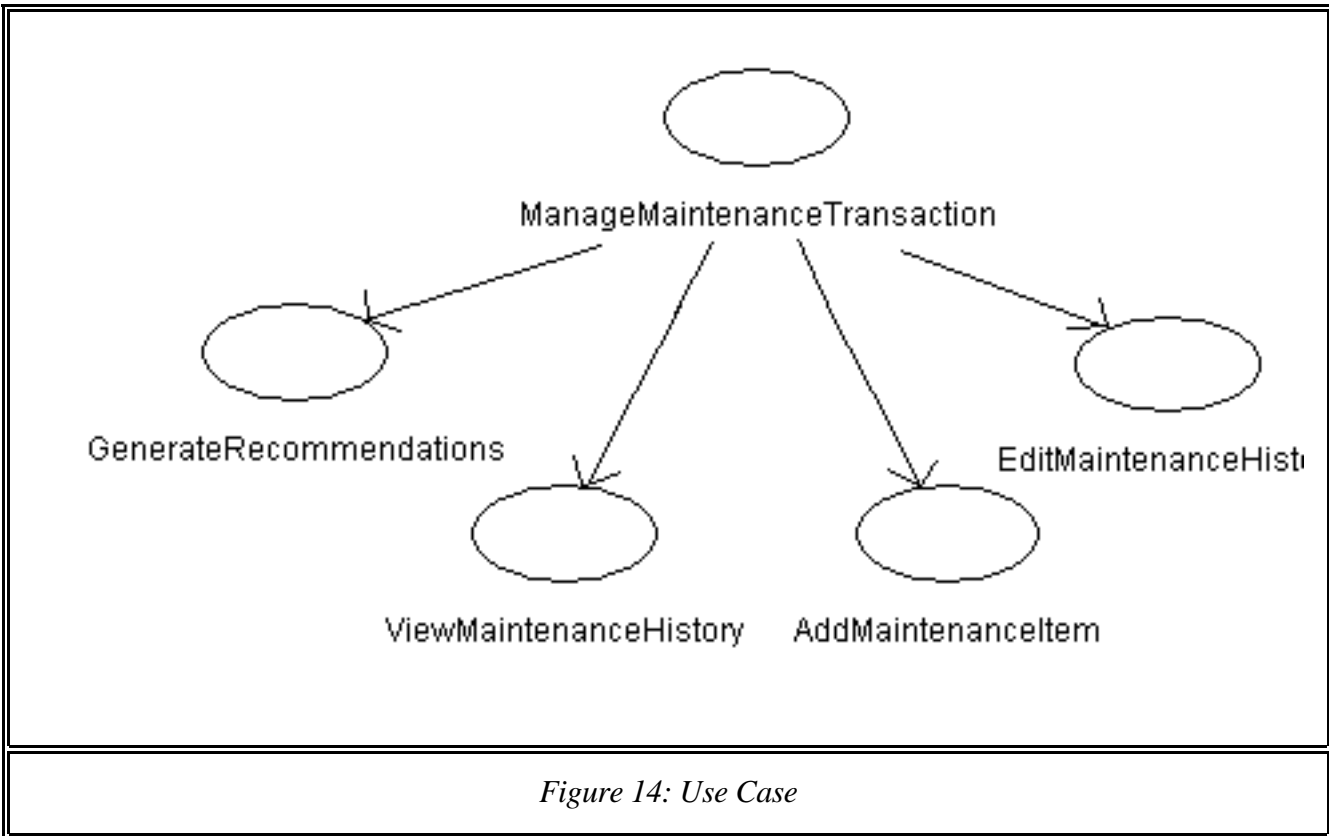
This user has a car pool and wants to maintain his cars by the dealer or the garage.

3.5.2.2 Use Cases

ManageMaintenanceTransaction

This use case describes the major actions that a user would perform through the Maintenance assistant. This use case consists of the following use cases:

- GenerateRecommendations
- ViewMaintenanceHistory
- AddMaintenanceItem
- EditMaintenanceHistory



GenerateRecommendations

Using data provided by GatherCardInformation, request a list of recommendations for service for this car from the RecommendationService.

Entry Conditions:

1. Maintenance information is available from GatherCardInformation.
2. Dealer representative requests recommendation information.

Flow of Events:

1. Send vehicle information to the RecommendationService.
2. Receive the recommendations from the RecommendationService

Exit Conditions:

1. Recommendations are successfully loaded from RecommendationService.

Special Requirements:

There must be a timeout for the communications layer.

ViewMaintenanceHistory

Obtain information about the history of the vehicle.

Entry conditions:

1. The MaintenanceUser or DealerRepresentative inserts a card into the card reader.
2. The Maintenance History Service is available.
3. The MaintenanceUser requests a portion of the information (or the entire history)

Event flow:

1. User inserts card
2. Sends request to Maintenance History Service
3. Information (or error status) is returned by communications layer.

Exit conditions:

1. The Maintenance History Service returns the requested information or the service is not available.

Special Requirements:

There must be a timeout for the communications layer.

AddMaintenanceItem

Dealer Representative adds a new piece of maintenance information to the Maintenance History Service.

Entry Conditions:

1. The Dealer Representative has inserted a card into card reader and is properly authenticated.
2. Dealer Representative indicates (perhaps by a button click) he/she wants to add new item of maintenance information.
3. The Maintenance History Service is available.

Event Flow:

1. Data is entered by authorized user.
2. Data is sent to Maintenance History Service
3. A confirmation is received.

Exit Conditions:

1. The item is added to the Maintenance History Service and confirmation is received

Special Requirements:

There must be a timeout for the communications layer.

EditMaintenanceHistory

This allows the administrator of the system to edit existing maintenance history information.

GatherCardInformation

Find out, what kind of user wants to activate the maintenance assistant. The use case decides if a service activity will be started or not (User authentication, validation, access depending on the user role).

Also maintenance data which will be transferred by the card can be sent to the MaintenanceTransactionManager.

Entry conditions:

Maintenance user information is available

Flow of events:

Maintenance user inserts a card

Receive the maintenance user data and possibly current maintenance data from the card

Exit conditions:

Maintenance information can be sent to activate the MaintenanceTransactionManager

Special Requirements:

if no maintenance information is available set maintenance data default = noCurrentData

AuthenticateUser

Determine user role from card information. If no maintenance user information exists on the card, the user is assigned a "guest" role, with minimal access to JAMES information. See [User Roles](#) section for more details.

ViewMaintenanceData

The Monitor of the Maintenance system should only be able to use a subset of methods (based on the user's access rights) that the Maintenance Transaction Manager supports.

A Maintenance User might be interested in viewing the maintenance data of his/her model of vehicle(s). This could be the driver or a fleet manager looking for information on his car fleet.

A company supplier which offers maintenance support to his customers related to the maintenance assistant should also be able to view a number of maintenance data, depending on queries on the user's data and on his role.

3.5.2.3. Use Case Diagram

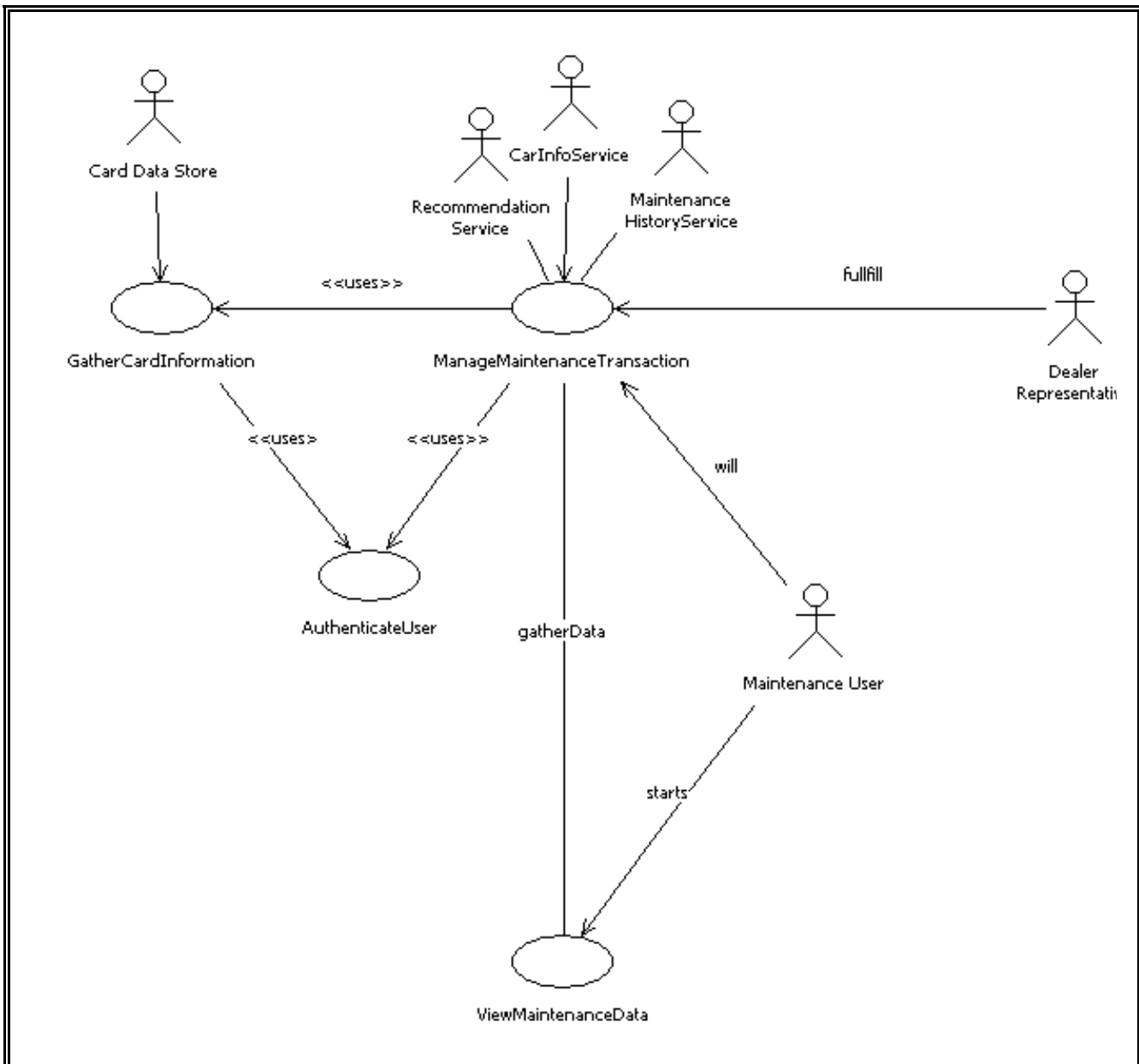


Figure 15: Use Case Diagram

3.5.3 Object Models

3.5.3.1 Data Dictionary

Card

Public Methods:

initializeCardlet () : Boolean
 setMaintHistIdentifier (Identifier : String = default) : Boolean
 getNewHistory (LastMaintItem : MaintenanceItem = default) : MaintenanceHistory
 updateHistory (OldMaintenanceHistory : MaintenanceHistory = default, UpdateCode : Integer = default,

NewMaintenanceHistory : MaintenanceHistoryBuffer = default) : Boolean

Protected Methods:

compare (FirsrtMaintItem : MaintenanceItem = default, SecondMaintItem : MaintenanceItem = default) : Boolean
sortByDate (MaintHistory : MaintenanceHistory = default) : Boolean

MaintenanceTransactionManager implements Edit

Public Methods:

stopCard () : boolean
synchronize () : boolean
getHistory ()
addEntry (Item : Object = default) : Object
deleteEntry (Item : Object = default) : Object
editEntry (Item : Object = default) : Object

MaintenanceHistoryBuffer implements DataStore, Edit

Public Methods:

getCount () : Long
setCount (Count : Long = default) : Boolean
setName (Identifier : Object = default) : Boolean
getName () : Object
open () : Boolean
first () : Object
last () : Object
next (MaintenanceItem : MaintenanceItem = default) : Object
prev (MaintenanceItem : MaintenanceItem = default) : Object
find (MaintenanceItem : MaintenanceItem = default) : Boolean
filter (FilterCriteria : Object = default) : Object
fetch (FromObject : Object = default, ToObject : Object = default) : Object

Item

Public Methods:

getCode () : byte
setCode (MaintenanceCode : byte = default) : Boolean

MaintenanceItem extends Item

Public Methods:

getDate () : Date
setDate (MaintenanceDate : Date = default) : boolean

DealerMaintenanceItem extends MaintenanceItem

Public Methods:

getDealerID () : String

```
setDealerID ( DealerID : String = default ) : boolean
getMechanicID ( ) : String
setMechanicID ( MechanicID : String = default ) : boolean
getMaintenanceItemDetail ( ) : String
setMaintenanceItemDetail ( MaintenanceItemDetail : String = default ) : boolean
```

CardMaintenanceHistory implements Edit, DataStore

Public Methods:

```
getCount ( argname : argtype = default ) : short
setCount ( Count : short = default ) : Boolean
setName ( Identifier : Object = default ) : Boolean
getName ( ) : Object
open ( ) : Boolean
first ( ) : Object
last ( ) : Object
next ( MaintenanceItem : MaintenanceItem = default ) : Object
prev ( MaintenanceItem : MaintenanceItem = default ) : Object
find ( MaintenanceItem : MaintenanceItem = default ) : Boolean
filter ( FilterCriteria : Object = default ) : Object
fetch ( FromObject : Object = default, ToObject : Object = default ) : Object
```

CardMaintenanceHistory implements Edit, DataStore

Public Methods:

```
getCount ( argname : argtype = default ) : short
setCount ( Count : short = default ) : Boolean
setName ( Identifier : Object = default ) : Boolean
getName ( ) : Object
open ( ) : Boolean
first ( ) : Object
last ( ) : Object
next ( MaintenanceItem : MaintenanceItem = default ) : Object
prev ( MaintenanceItem : MaintenanceItem = default ) : Object
find ( MaintenanceItem : MaintenanceItem = default ) : Boolean
filter ( FilterCriteria : Object = default ) : Object
fetch ( FromObject : Object = default, ToObject : Object = default ) : Object
addEntry ( Item : Object = default ) : Object
deleteEntry ( Item : Object = default ) : Object
editEntry ( Item : Object = default ) : Object
```

DealerMaintenanceHistory implements DataStore

Public Methods:

```
getCount ( argname : argtype = default ) : short
setCount ( Count : short = default ) : Boolean
setName ( Identifier : Object = default ) : Boolean
getName ( ) : Object
open ( ) : Boolean
first ( ) : Object
last ( ) : Object
next ( MaintenanceItem : MaintenanceItem = default ) : Object
prev ( MaintenanceItem : MaintenanceItem = default ) : Object
find ( MaintenanceItem : MaintenanceItem = default ) : Boolean
filter ( FilterCriteria : Object = default ) : Object
```

fetch (FromObject : Object = default, ToObject : Object = default) : Object

MaintenanceCardlet

Public Methods:

MaintApplet (Message : byte[] = default, Length : byte = default) : byte

Interface DataStore

Public Methods:

setName (Identifier : Object = default) : Boolean
getName () : Object
open () : Boolean
first () : Object
last () : Object
next (MaintenanceItem : MaintenanceItem = default) : Object
prev (MaintenanceItem : MaintenanceItem = default) : Object
find (MaintenanceItem : MaintenanceItem = default) : Boolean
filter (FilterCriteria : Object = default) : Object
fetch (FromObject : Object = default, ToObject : Object = default) : Object

Interface Edit

Public Methods:

addEntry (Item : Object = default) : Object
deleteEntry (Item : Object = default) : Object
editEntry (Item : Object = default) : Object

3.5.3.2 Class Diagrams

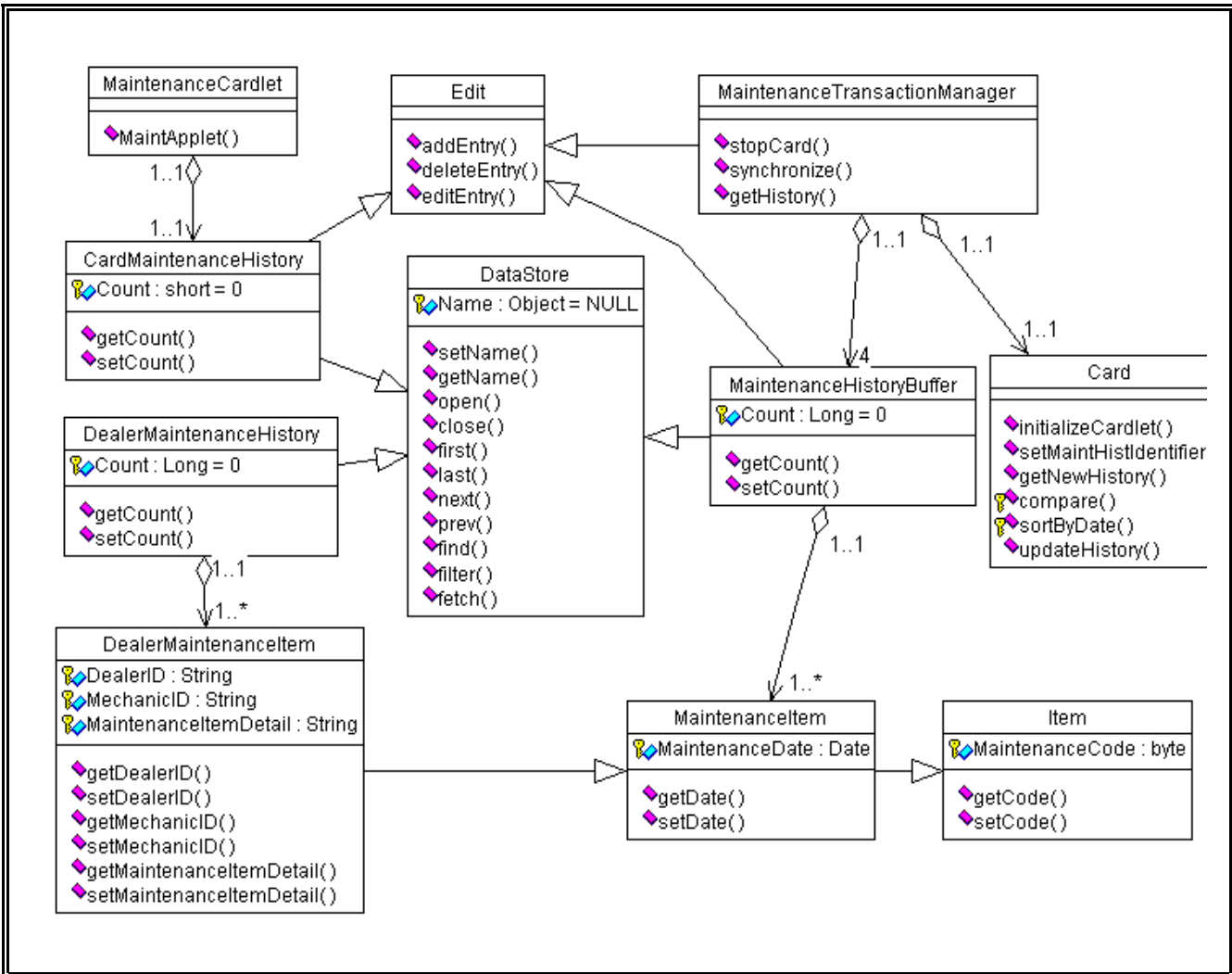


Figure 16: Class Diagram

3.5.4 Dynamic Models

The following models describe the sequence of interaction between the various actors and objects in our system. Please note that the Vehicle object includes all information about the status of the vehicle, including any information about requested maintenance operations.

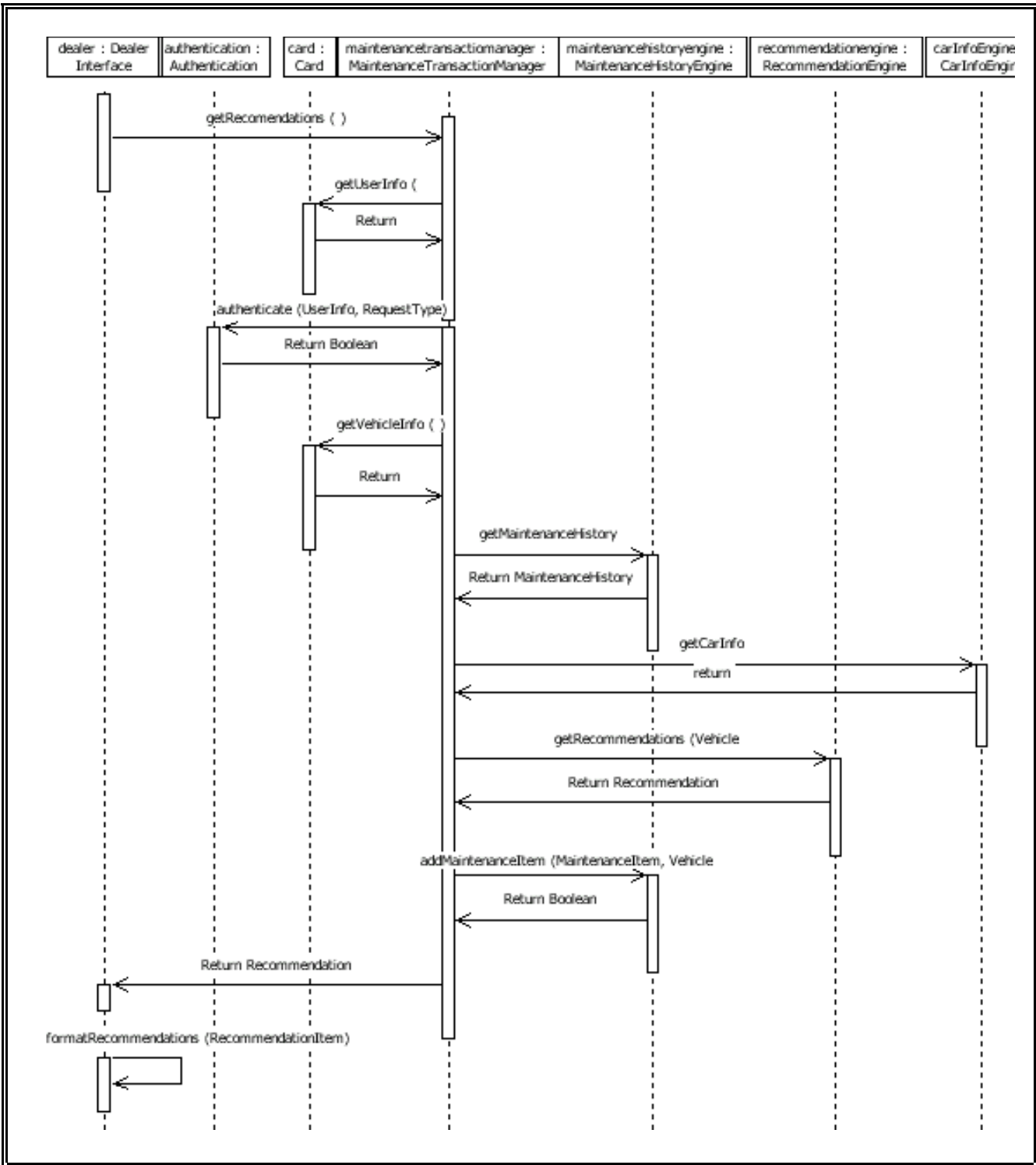


Figure 17: Sequence diagram for Car Service

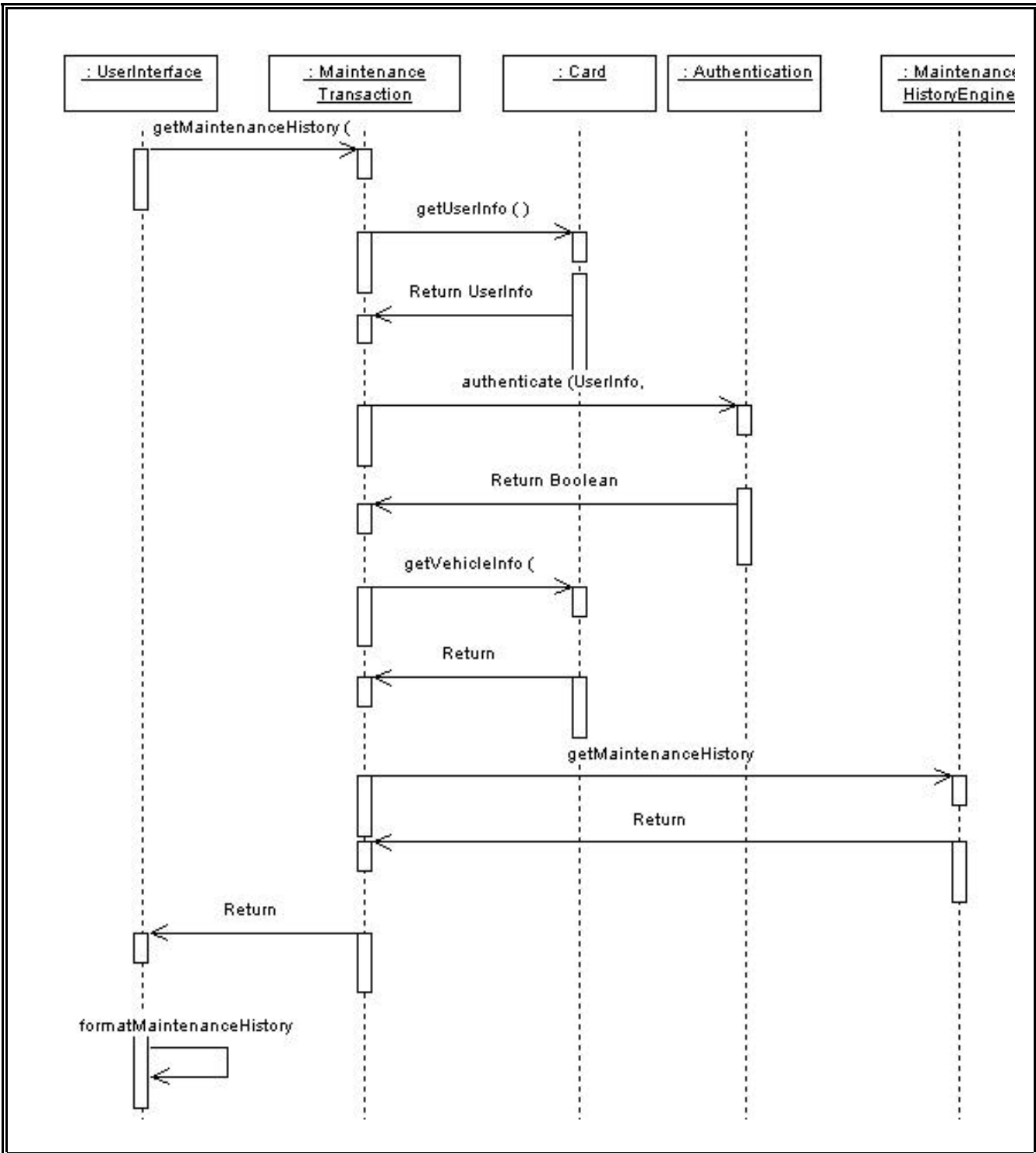
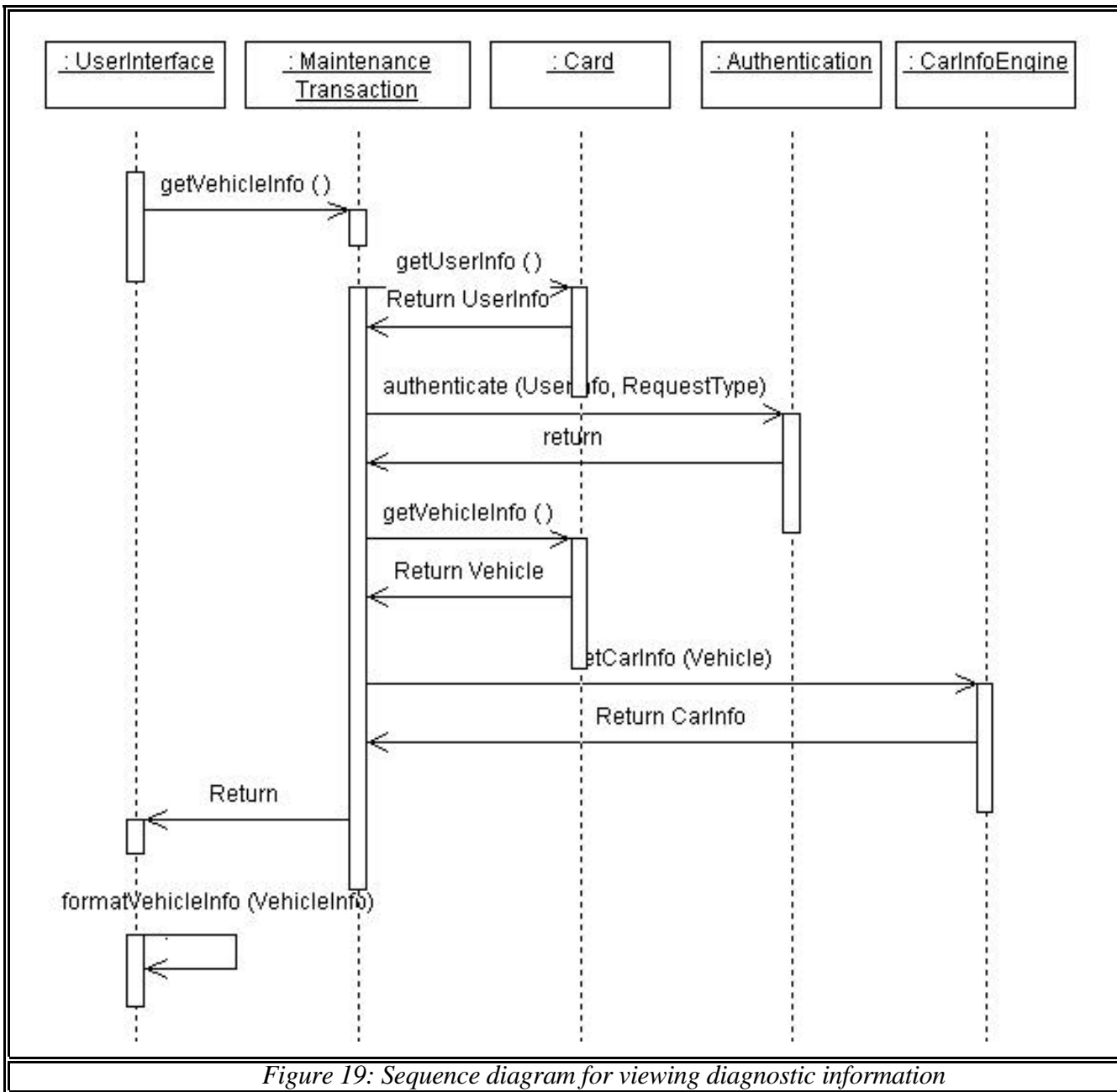


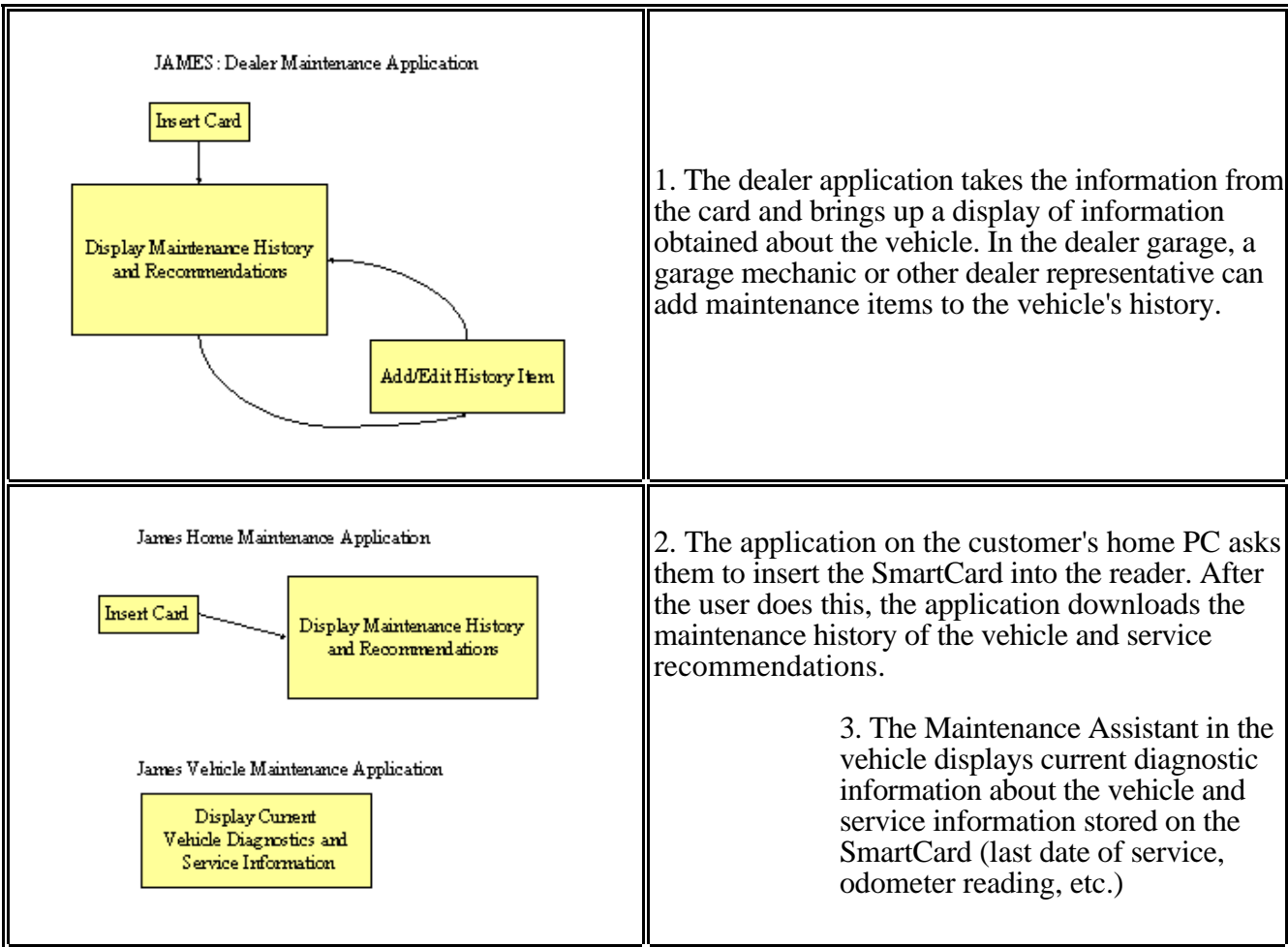
Figure 18: Sequence diagram for viewing maintenance information



3.5.5 User Interface - Navigational Paths and Screen Mockups

There are three venues for the JAMES Maintenance Assistant:

- #. In the dealer garage
- #. On a customer's home PC
- #. In the vehicle



Screen Mockups

Below is the application that a mechanic would interact with. The interface provides a way to view and edit the maintenance history of the vehicle and view recommendation. The driver would use a similar interface at home, except the option to generate a new maintenance entry would not be available.

MaintenanceHome - Form Designer

Maintenance History

Date	Service Performed	Dealer
------	-------------------	--------

New...

Service Recommendations

Importance	Description	Type
------------	-------------	------

Card Inserted

Figure 20: Maintenance History

The screen shot below is a sample of the interface a driver might see in the vehicle. It merely provides a facility for viewing data about the car. No editing is allowed.

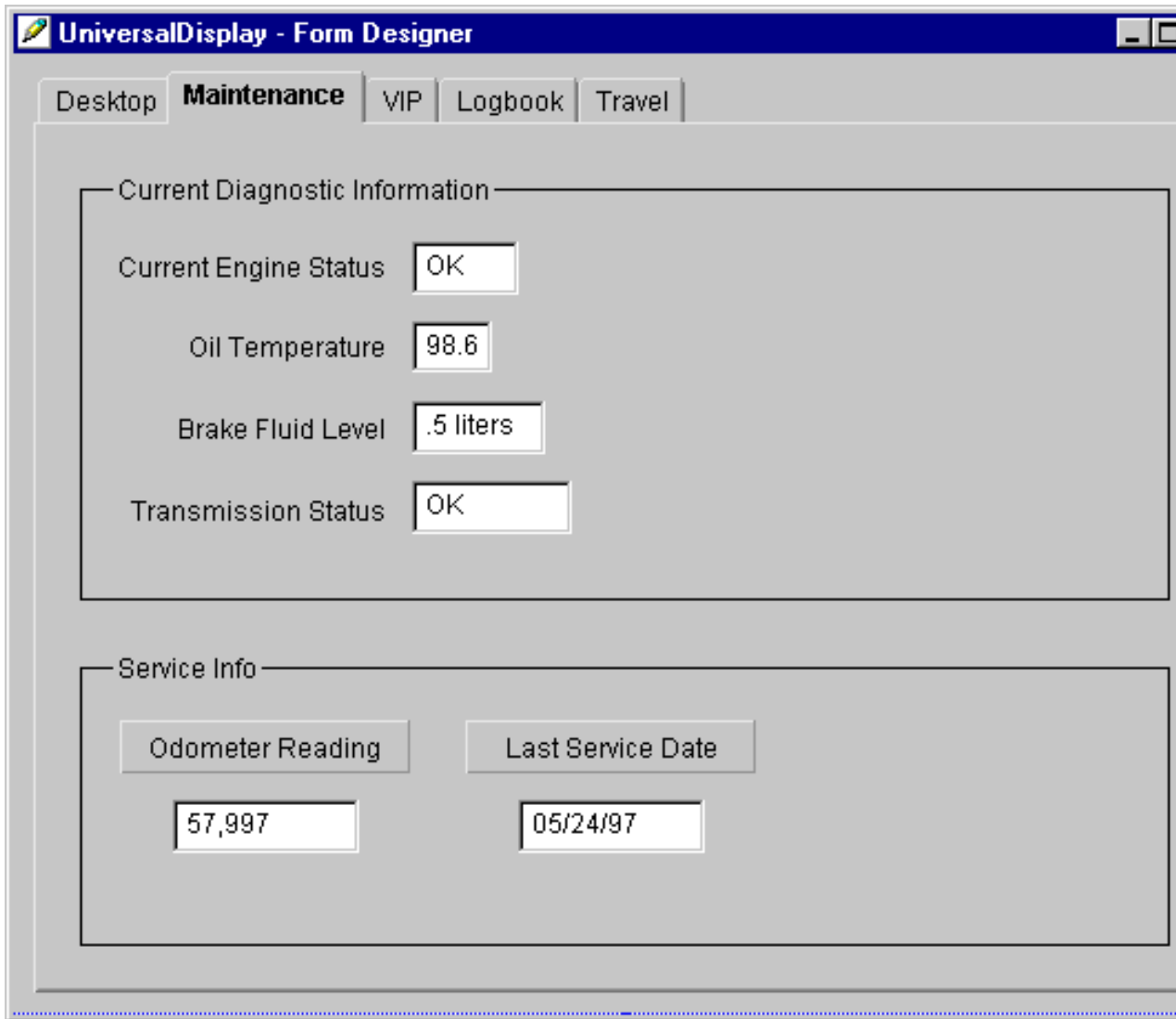


Figure 21: Maintenance Tab