TRAMP

Traveling Repair And Maintenance Platform

Requirements Analysis Document

Authors: TRAMP Students
Version: 1.1
Date: 13.01.2002

Table Of Contents

1. Problem Statement

1.1. Introduction

By introducing the assembly line for building cars in 1923, Henry Ford opened the era of mass production. A car is a complex system and maintaining it requires a very good knowledge of the functionality, structure and dynamic behavior of its many parts. The 90s introduced the era of mass customization, the ability to mass manufacture products, but tailor them to individual people. The maintenance of such an individual product is now complicated, because the maintainer has to identify the particular system and its parts before they can start with the maintenance. Combining both, complex systems and mass customization, introduces problems for the car mechanics: The required knowledge for the repair processes for all the systems is now quite substantial. Furthermore, they are never up to date: the knowledge base is never constant, because new customized models with new variants are constantly produced.

1.1.1. Purpose of the System

The recent progress in computer hardware, networking technologies and software engineering has opened up an opportunity to deal with these problems. TRAMP investigates the use of augmented reality, wearable and mobile computers for the maintenance of cars. We are moving away from the traditional desktop solution where the mobile mechanic moves to the computer to get the desired information. Instead, we want the information to move where ever the mechanic is currently located. The system to be developed is a functional prototype of a mobile wireless system for supporting the repair and maintenance of complex systems like for example cars.

1.1.2. Scope of the System

As mentioned above TRAMP is intended to be a working sample for a traveling repair and maintenance platform. However it will also provide capabilities that can be used as part of a visionary UMTS Mobile Maintenance system.

1.1.3. Objectives and Success Criteria of the Project

TRAMP's objectives can be divided in two parts. First, there are some overall goals that have to be kept in mind especially during the system design phase. o Overall Goal: Develop a Car Maintenance System that supports the TRAMP scenario and provides as much capability as possible as constrained by the functional and nonúfunctional requirements and the time assigned to the class o Secondary Goal: Identify and evaluate new UMTS applications to support vehicle maintenance Second, another objective that focuses on the actual development process exists: o Merge new system components with existing components from Inmedius and TUM. The success criteria of the project is the acceptance by the client.

1.2. Proposed System

1.2.1. Overview

The system we propose is designed as a collection of components that interact flexibly with each other to build an efficient and wearable mobile wireless repair and maintenance system. These components can run on separate hardware components that are then networked together to provide the desired functionality. There are five main components we have identified: Application, Session, User Interface, Context and Network. The Network component is responsible for connecting the other components. It

will provide the middleware and support the following technologies: UMTS, WaveLAN, Network Poll, HTTP, OpenSLP, GPRS and GSM. The Context component provides information about the users current position, tracks optical markers (e.g. open fuse box) and plans routes. The User Interface component is responsible for realizing a multi-modal user interfacing including speech recognition and Flash for the Graphical User Interface. The Session component provides a session concept for wearable applications coordinating several interacting users. Finally the Application component is the glue between the other components mentioned before. It links them together in order to realize a mobile maintenance application.

2 . Requirements

This section describes the user needs that the system has to support in terms of actors, user tasks, domain constraints and quality constraints on user tasks.

2.1. Actors

Actors are entities that interact with the system.

2.1.1. Customer

Description:

A A:Customer is a driver that is responsible for a car's normal operation. A driver can obtain preventive or corrective maintenance from the car manufacturer and emergency road side assistance.

Initiated User Tasks:

Perform maintenance at dealership
Repair car on roadside

Initiated Use Cases:

Find Nearest Garage
Perform maintenance at dealership using TRAMP
Repair car on road side using TRAMP
Request assistance
Request maintenance

Participating Use Cases:

Cash-Payment
E-Payment
Find Nearest Garage
Guide customer to parking spot
Issue Bill to customer
Request assistance
Request maintenance

Instances:

Anton
John

Open Questions:

Inconsistency Issue: How does the customer know the garage

2.1.2. Customer Representative

Description:

No description specfied.

Initiated User Tasks:

No user tasks specified.

Initiated Use Cases:

Guide customer to parking spot
Handover maintenance task

Participating Use Cases:

Guide customer to parking spot
Handover maintenance task
Repair car on road side using TRAMP
Request assistance

Request maintenance

Instances:

Toni

Open Questions:

Inconsistency Issue: How does the customer know the garage

2.1.3. Mechanic

Description:

A mechanic is a professional who can repair or maintain a car. A:Mechanics can provide services either in a fixed garage or by traveling to the car to be repaired.

Initiated User Tasks:

No user tasks specified.

Participating User Tasks:

Repair car on roadside

Initiated Use Cases:

Calibrate system
Cash-Payment
E-Payment
Execute procedure
Find customer at parking spot
Find stranded customer
Issue Bill to customer
Request expert
Retrieve Status from Embedded System
Retrieve maintenance Instructions
Retrieve maintenance records

Participating Use Cases:

Calibrate system
Cash-Payment
Find customer at parking spot
Handover maintenance task
Issue Bill to customer
Repair car on road side using TRAMP
Request expert
Retrieve Status from Embedded System

Instances:

Brandon
Manfred

Open Questions:

Challenge on content: Is ES part of TRAMP or another actor

2.1.4. Remote Expert

Description:

A:Remote Experts are persons that can assist a A:Mechanic in a repair task by providing knowledge remotely via the system.

Initiated User Tasks:

No user tasks specified.

Participating User Tasks:

Repair car on roadside

Initiated Use Cases:

No use cases specified.

Participating Use Cases:

Execute procedure
Request expert

2.1.5. Synthetic Expert

Description:

Sythetic Experts are computer agents that can actively support A:Mechanics via the system and that mimic a A:Remote Expert.

Initiated User Tasks:

No user tasks specified.

Initiated Use Cases:

No use cases specified.

Participating Use Cases:

- Execute procedure
- Request expert

2.2. User Tasks

User tasks are activities initiated by actors that are supported by the system.

2.2.1. Perform maintenance at dealership

Initiating Actor:

- Customer

Task Description:

The A:Customer drives to the car dealership and requests preventive or corrective maintenance. A A:Customer Representative welcomes the customer, records the requests and any diagnostic information from the customer, and finds an available A:Mechanic.

The A:Mechanic takes over the information from the A:Customer Representative, obtains the maintenance record of the car, and performs the requested maintenance.

Once the maintenance is completed, the A:Customer pays and reclaims his car.

Domain Constraints:

- Multiple Users
- Wearability

Realized in Use Cases:

- Execute procedure
- Find customer at parking spot
- Guide customer to parking spot
- Handover maintenance task
- Perform maintenance at dealership using TRAMP
- Request maintenance
- Retrieve maintenance Instructions
- Retrieve maintenance records

Open Questions:

- Omission Issue: find the parking spot
- Challenge on content: Is ES part of TRAMP or another actor

2.2.2. Repair car on roadside

Initiating Actor:

- Customer

Participating Actors:

- Mechanic
- Remote Expert

Task Description:

The driver requests assistance from the car manufacturer, via a hotline.

The car manufacturer notifies the A:Mechanic of a A:Customer in need of assistance. The A:Mechanic obtains information about the breakdown and the location of the car. He loads the spare parts that might be needed and drives to the car to be repaired.

During the car repair, the A:Mechanic can obtain further information about diagnostics and repair procedures from the home office, either by searching a knowledge base, talking with a remote or a A:Synthetic Expert.

Domain Constraints:

- Hostile Environment
- Multiple Users

Wearability

Realized in Use Cases:

Find stranded customer

Repair car on road side using TRAMP

Request assistance

Request expert

Open Questions:

Inconsistency Issue: How does the customer know the garage

Challenge on content: Is ES part of TRAMP or another actor

2.3. Domain Constraints

Domain constraints are facts that the system must take into account.

2.3.1. Hostile Environment

Type:

Domain Constraint

Description:

The system should be able to operate in the presence of dirt, noise or other environmental difficulties.

Associated User Tasks

Repair car on roadside

2.3.2. Multiple Users

Type:

Domain Constraint

Description:

The system should support tasks that are performed by NFC:Multiple Users in concert, supplying each with the necessary and correct information at the appropriate time.

Associated User Tasks

Perform maintenance at dealership

Repair car on roadside

2.3.3. Wearability

Type:

Domain Constraint

Description:

The wearable system should be designed not to encumber the user.

Associated User Tasks

Perform maintenance at dealership

Repair car on roadside

2.4. Quality Constraints on User Tasks

Constraints the user tasks have to meet.

2.4.1. Minimize Customer Time

Type:

Quality Constraint on User Task

Description:

The A:Customer should spend the least amount of time possible at the dealership or waiting on the roadside after requesting assistance.

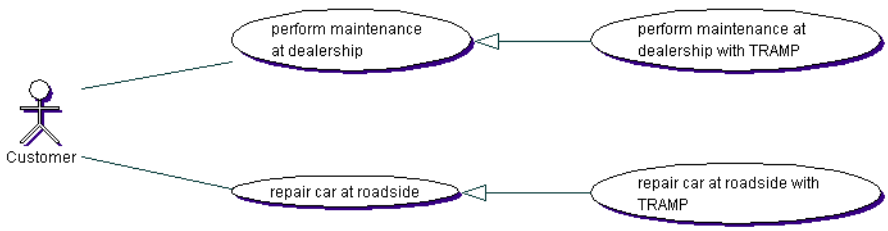
3 . Specification

This section describes the specification of the system in terms of use cases, services, and quality constraints.

3.1. Use Cases

Use cases describe sequences of interactions between actors and the system.

3.1.0. Overview



3.1.1. Calibrate system

Sequence Diagram

Initiating Actor:
Mechanic
Participating Actors:
Mechanic
Realized User Task:
No user task specified.
Flow of events:
1. - *Actor* - The A:Mechanic activates the "calibration" function of her/his wearable.
2. - *System* - The system initiates the calibration. (use Initiate Calibration)
3. - *System* - The system displays a calibration pattern in the HMD. (use Display Calibration)
4. - *Actor* - The A:Mechanic aligns the pattern with some real-world object. (Details of this process need to be specified as soon as the exact system configuration is known.)
5. - *System* - The system repeatedly responds to user input. (use Get Calibration Input)
6. - *Actor* - The A:Mechanic confirms the alignment. [Not properly calibrated]

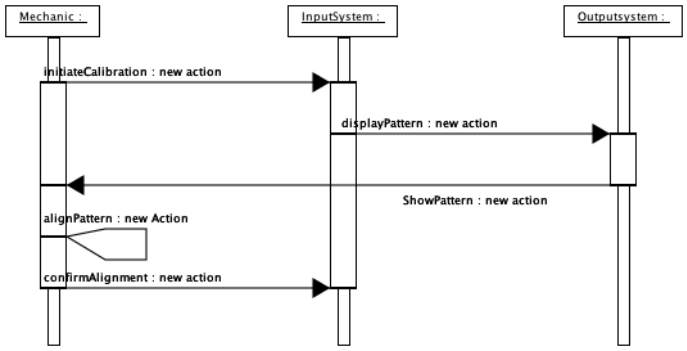
Exceptions:
[Not properly calibrated]
1. The A:Mechanic repeats from step 1 until s/he is satisfied with the result.

Preconditions:
The tracking system is properly configured and running.
The objects displayed on the HMD are not displayed at the right spot.

Exit conditions:
The display and tracking systems are calibrated.

Quality Constraints:
calibration duration
calibration necessity

Used Services:
Display Calibration
Get Calibration Input
Initiate Calibration



3.1.2. Cash-Payment

Sequence Diagram

Initiating Actor:

- Mechanic

Participating Actors:

- Customer
- Mechanic

Realized User Task:

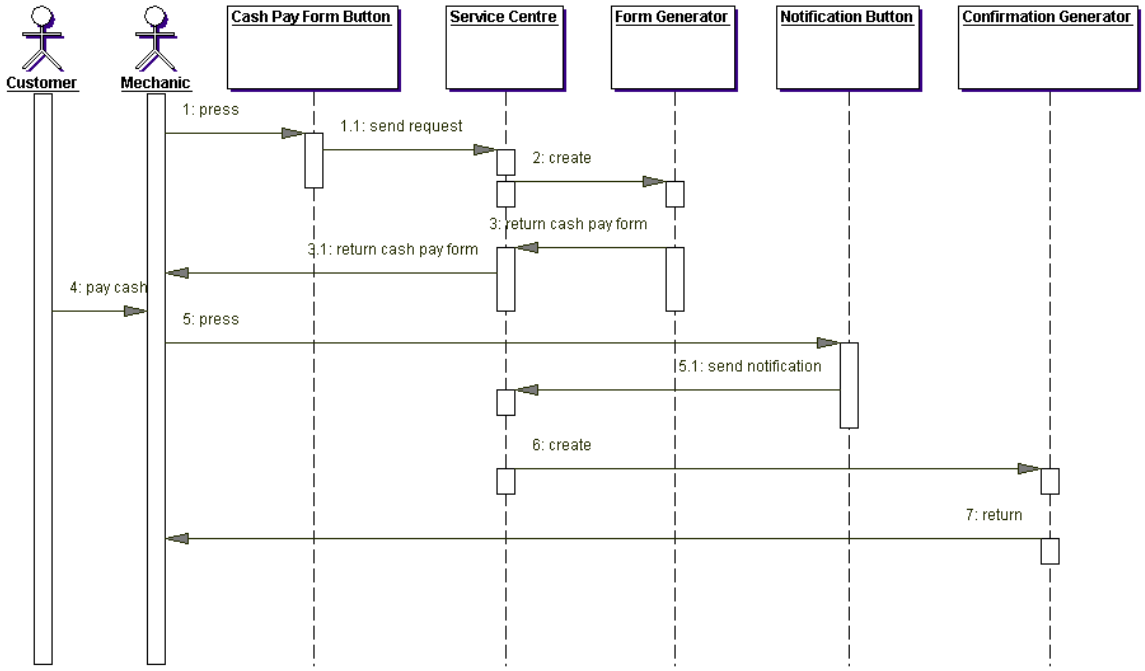
- *No user task specified.*

Flow of events:

1. - *Actor* - The A:Mechanic requests cash pay form
2. - *System* - System issues cash pay form (use Send Cash Payment Form)
3. - *Actor* - A:Customer pays cash and A:Mechanic Send
- Payment Notification to the service center.
4. - *System* - The system receives the cash payment notification from the A:Mechanic (use Receive Cash Payment Notification)
5. - *System* - The system issues payment confirmation back to the A:Mechanic. (use Send Payment Confirmation)

Used Services:

Receive Cash Payment Notification
- Send Cash Payment Form
Send Payment Confirmation
-

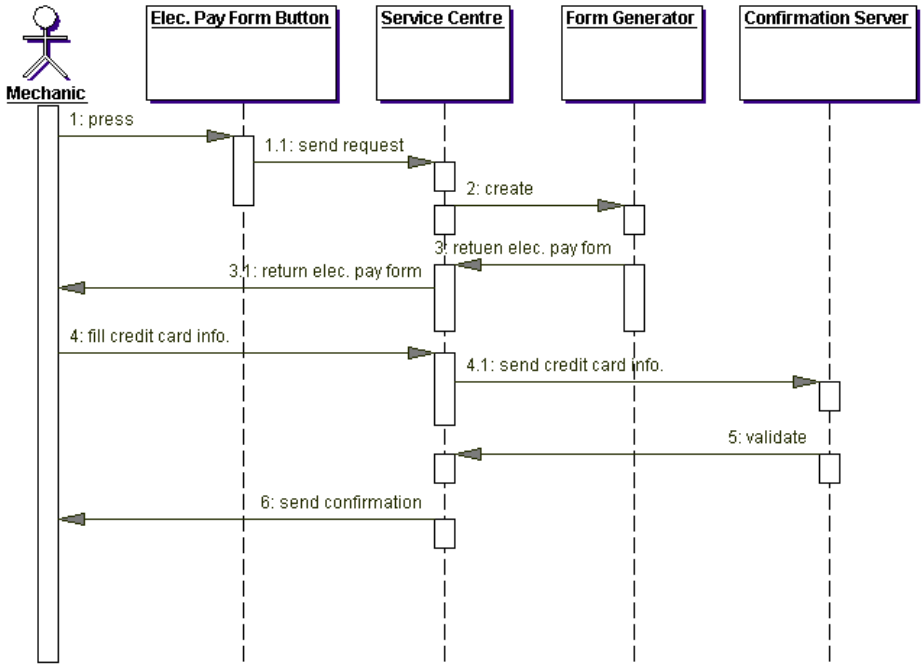


3.1.3. E-Payment

Sequence Diagram

Initiating Actor:
Mechanic
Participating Actors:
Customer
Realized User Task:
No user task specified.
Flow of events:
1. - *Actor* - A:Mechanic requests electronic payment form for an issued bill.
2. - *System* - System sends payment form to the A:Mechanic (use send E-payment form)
3. - *Actor* - A:Customer fills in credit data and sends the form back to the system.
4. - *System* - System receives payment data (use Receive Credit Card Data)
5. - *System* - System validates payment data (use Validate Payment Data)
6. - *System* - System sends payment confirmation (use Send Payment Confirmation)
7. - *Actor* - Receives payment confirmation from multi-modal output device

Exceptions:
Data transfer is interrupted by loosing connection to server
Payment info is not valid
Preconditions:
An accepted bill exists and the A:Customer want to pay electronically
Exit conditions:
Electronic payment was successfully completed, A:Customer got receipt of payment
Used Services:
Receive Credit Card Data
Send Payment Confirmation
Validate Payment Data
send E-payment form
Open Questions:
Inconsistency Issue: No need to enter payment info



3.1.4. Execute procedure

Sequence Diagram

Initiating Actor:

- Mechanic

Participating Actors:

- Remote Expert
- Synthetic Expert

Realized User Task:

- Perform maintenance at dealership

Flow of events:

1. - *Actor* - The A:Mechanic enters his desired command via the *input system*. (include Retrieve maintenance Instructions)
2. - *System* - The *input system* initiates the *taskflow* which handles the wished action. (use Initiate Taskflow)
3. - *System* - *Repair information* is requested and displayed to the *A:Mechanic* via the *output system*. [IETM error] (use Transfer Requested Technical Data)
4. - *Actor* - The A:Mechanic performs his wished action or - if desired - requests help via the *input/output system* from an *expert system* or a *remote expert*. [no A:Remote Expert result] (include Request expert)
5. - *Actor* - If the problem is solved the A:Mechanic sends a "problem solved" message via the *input system* to the taskflow, that is closed afterwards. Otherwise he repeats steps 2 to 5. [no spare part]
6. - *System* - The A:Mechanic gets informed about the closing of the taskflow (use Close Taskflow)

Exceptions:

[no spare part]
means, necessary spare-part is not available
[IETM error]

- occurs, when no IETM for problem available/ IETM-documentation leads to no result -> contact remote-expert
[no A:Remote Expert result]
means, contacting remote-expert leads to no result

Preconditions:

* pre-diagnosis is available
* vehicle data available
* list of all possible spare-parts concerning the problem is available
* most likely spare-parts for repair-process are available (and are already prepared)
* necessary tools are prepared
* A:Mechanic is at car's location
* necessary IETM-Documentation as records and instructions have been downloaded to SPOT. (UC:Handover maintenance task)

Exit conditions:

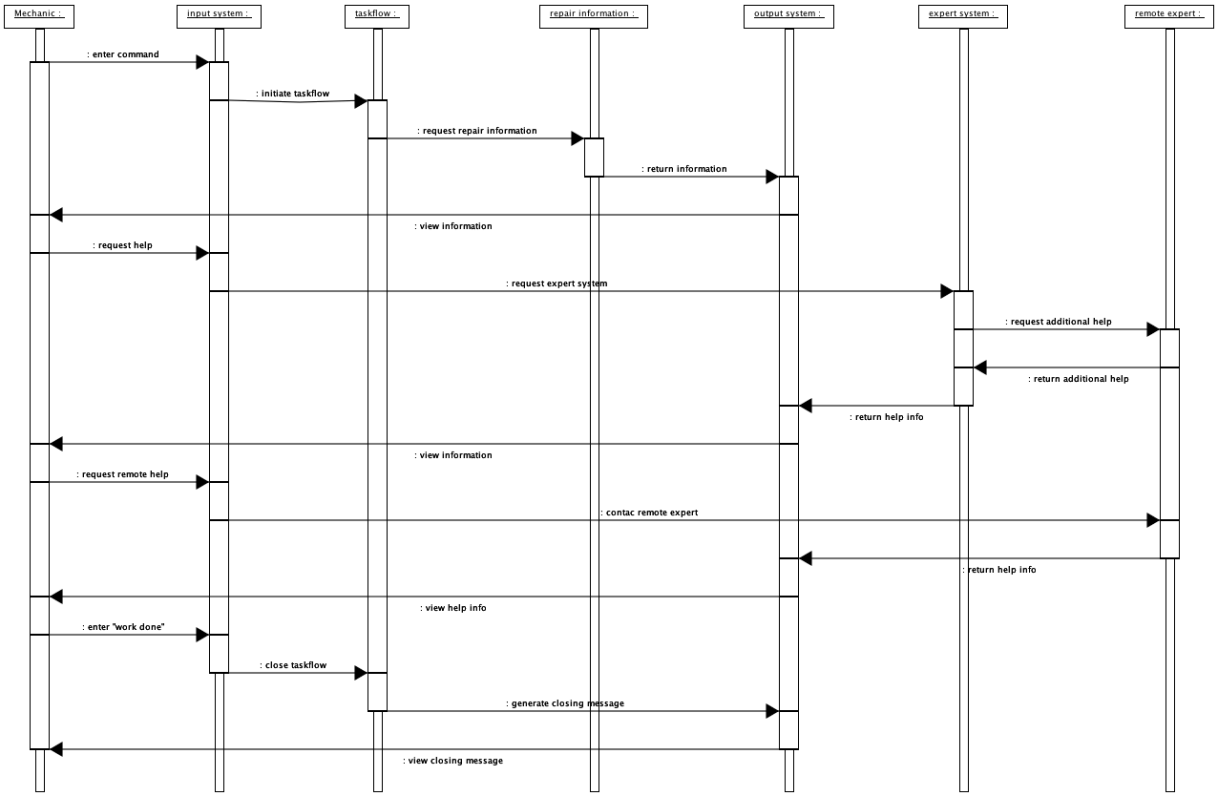
* repair-process is successful

Used Services:

Close Taskflow
Initiate Taskflow
Transfer Requested Technical Data

Open Questions:

Omission Issue: Should system be proactive?
Omission Issue: Don't zou want to include the service "Detect
Mark



3.1.5. Find Nearest Garage

Sequence Diagram

Initiating Actor:
- Customer

Participating Actors:
- Customer

Realized User Task:
- *No user task specified.*

Flow of events:

1. - *Actor* - A:Customer calls a service number of his car manufacturer with his mobile phone .
2. - *System* - At the service point the A:Customer´s position is investigated. (use Get User Position)
3. - *Actor* - The A:Customer hands over his current location.
4. - *System* - The nearest garage address is retrieved from a database and given back to the A:Customer together with the description of the shortest way to get there. (use Find Shortest Path)
5. - *Actor* - The A:Customer receives the address and hangs up.

Exceptions:
- The mobile connection interrupts due to wave interferences/low battery power/...

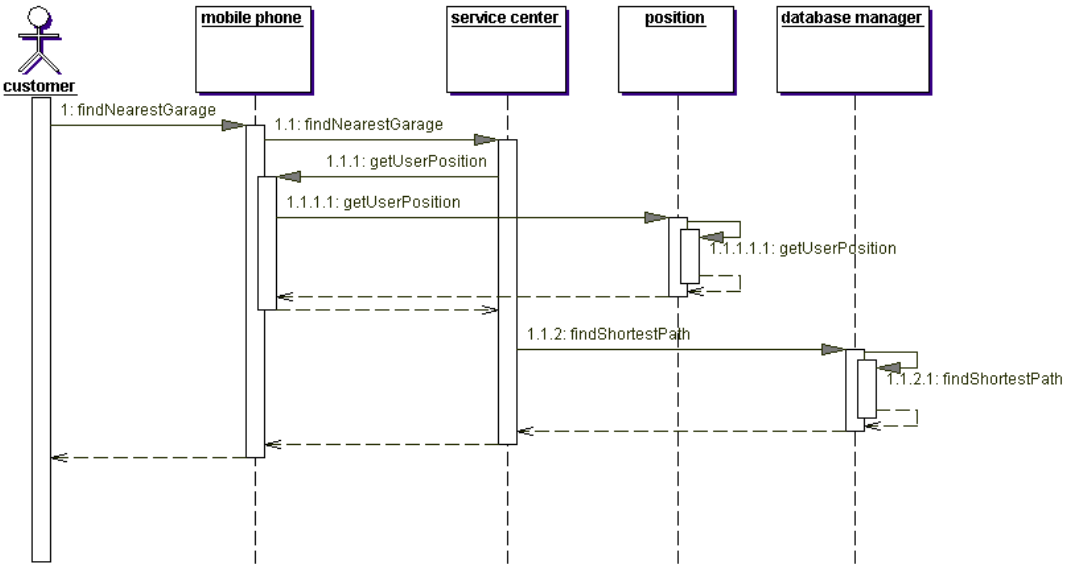
Preconditions:
- A:Customer wants to know where the nearest garage is located

Exit conditions:
- A:Customer has the adress of the closest garage to his location

Used Services:
- Find Shortest Path
- Get User Position

Open Questions:

Omission Issue: Should we include a call center agent?
Inconsistency Issue: Do "request assistance" and "find nearest garage" overlap?
Inconsistency Issue: How does the customer know the garage



3.1.6. Find customer at parking spot

Sequence Diagram

Initiating Actor:
- Mechanic

Participating Actors:
- Mechanic

Realized User Task:
- Perform maintenance at dealership

Flow of events:

1. - *System* - The A:Mechanic retrieves the number of A:Customer 's parking spot along with the estimated arrival time of A:Customer from the job database
2. - *System* - The shortest way to the parking spot will be calculated. (use Find Shortest Path)
3. - *System* - A map of the company complex (taking the changing viewpoint of the A:Mechanic into consideration) is displayed on the A:Mechanic's HMD. The assigned parking spot as well as a the shortest route to get there are highlighted on the map.
- [Tracking Failure] (use Navigate Mechanic)
4. - *Actor* - A:Mechanic uses the tranmitted map to navigate to the parking spot. Gesture Recognition might be used to help him/her achieving this task. So looking up will cause the displayed map to shrink and move to a spot where it will not hinder his/her visual perceptions and looking down will enlarge the map and show it in greater detail.
5. - *System* - Continually update the map until A:Mechanic arrives at the assigned parking spot. This may also include Enlargments of the map. (use Enlarge Map)

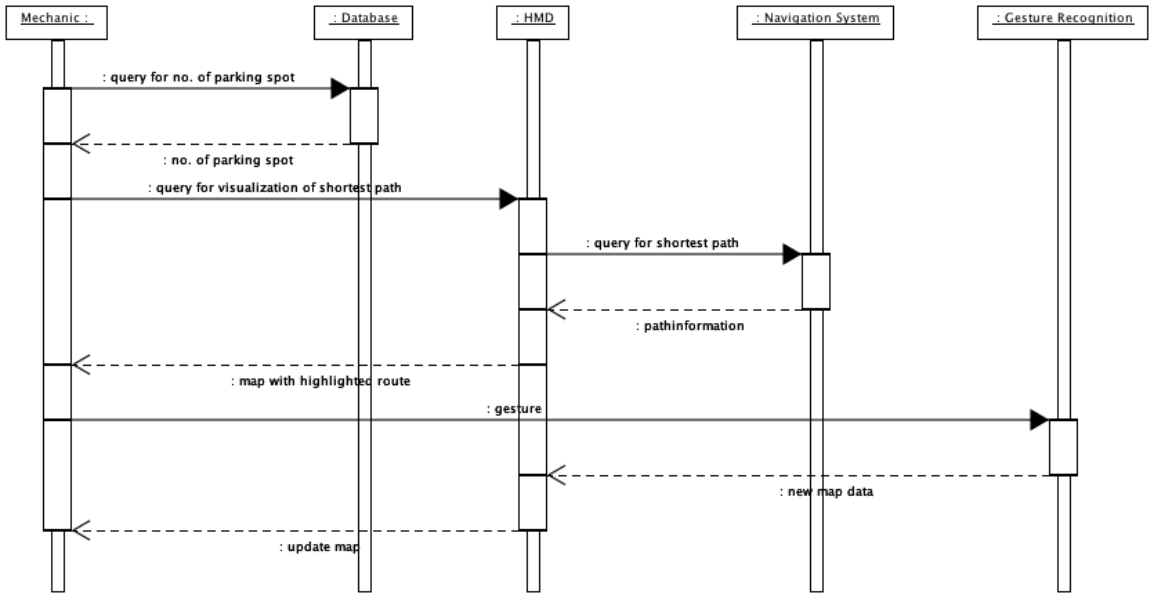
Exceptions:
- [Tracking Failure] => Inform the A:Mechanic, so he can go to a GPS sensible location.

Preconditions:
- A:Mechanic has been assigned the job of repairing the car of A:Customer

Exit conditions:
- A:Mechanic has reached the parking spot, assigned to A:Customer

Quality Constraints:
- Performance of AR-related tasks

Used Services:
- Enlarge Map
- Find Shortest Path
- Navigate Mechanic



3.1.7. Find stranded customer

Sequence Diagram

Initiating Actor:

Mechanic

Participating Actors:

No participating actors specified.

Realized User Task:

Repair car on roadside

Flow of events:

1. - *System* - The system retrieves the A:Customer's location from the job database (use Get User Position)
2. - *System* - The system has to navigate the A:Mechanic to the A:Customer. (use Navigate Mechanic)
3. - *System* - The system continually displays the fastest route from the A:Mechanic's actual position to the position of the A:Customer's car on the A:Mechanic's HMD.[Tracking Failure] (use Find Shortest Path)
4. - *System* - The A:Mechanic should also have the option to enlarge the map if needed. (use Enlarge Map)
5. - *Actor* - The A:Mechanic drives to the A:Customer's car

Exceptions:

[Tracking Failure] => Inform the A:Mechanic, so he can go to a GPS sensible location.

Preconditions:

Job is assigned to A:Mechanic

Exit conditions:

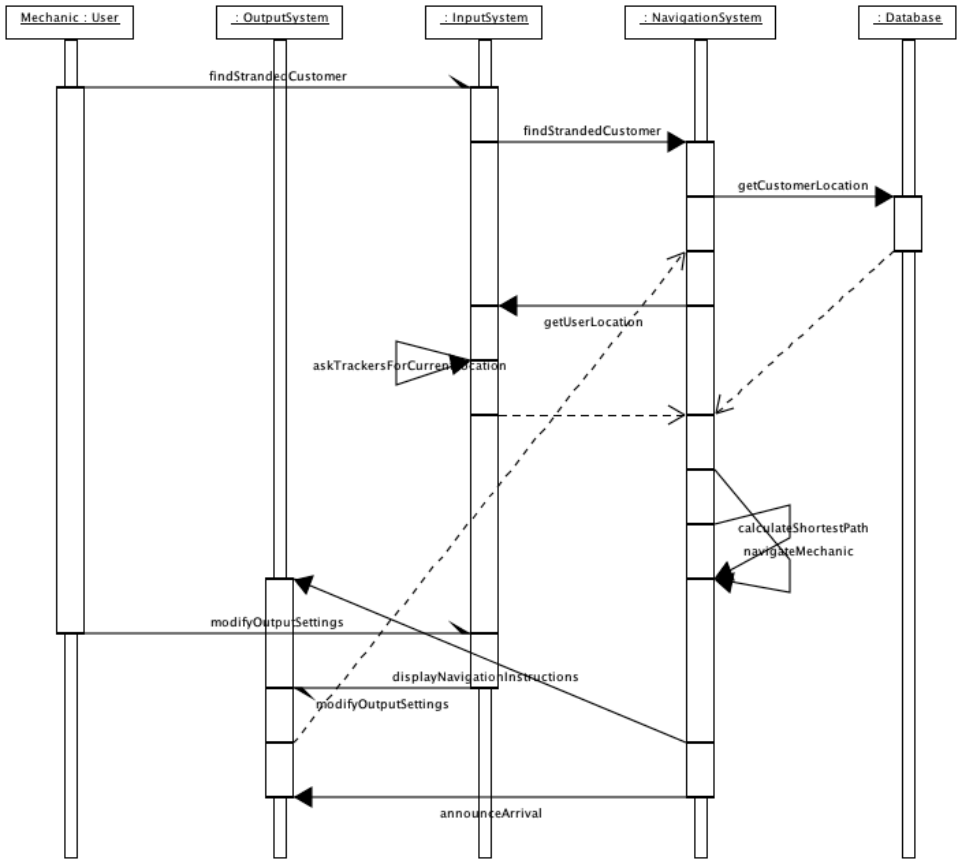
A:Mechanic is at the A:Customer's broken down car

Used Services:

Enlarge Map
Find Shortest Path
Get User Position
Navigate Mechanic

Open Questions:

Challenge on content: You can't drive with an HMD on
Inconsistency Issue: How does the customer know the garage



3.1.8. Guide customer to parking spot

Sequence Diagram

Initiating Actor:
- Customer Representative

Participating Actors:
- Customer
- Customer Representative

Realized User Task:
- Perform maintenance at dealership

Flow of events:

1. - *System* - system assigns parking spot
[no parking spot available] (use Assign Parking Spot)
2. - *System* - system transfers parking spot location to car navigation system
[transfer to A:Customer's navigation system fails] (use Transfer Navigation Information)
3. - *Actor* - A:Customer is guided to parkingspot

Exceptions:

[no parking spot available]
=> find out when this will change
inform the user
- eventually try to find alternatives

[transfer to custumer's navigation system fails]
=> verbal or paper-based navigation instructions are transfered

Preconditions:
- system got A:Customers data
maintainance by a A:Mechanic is needed

Exit conditions:
- A:Customer is at parking spot

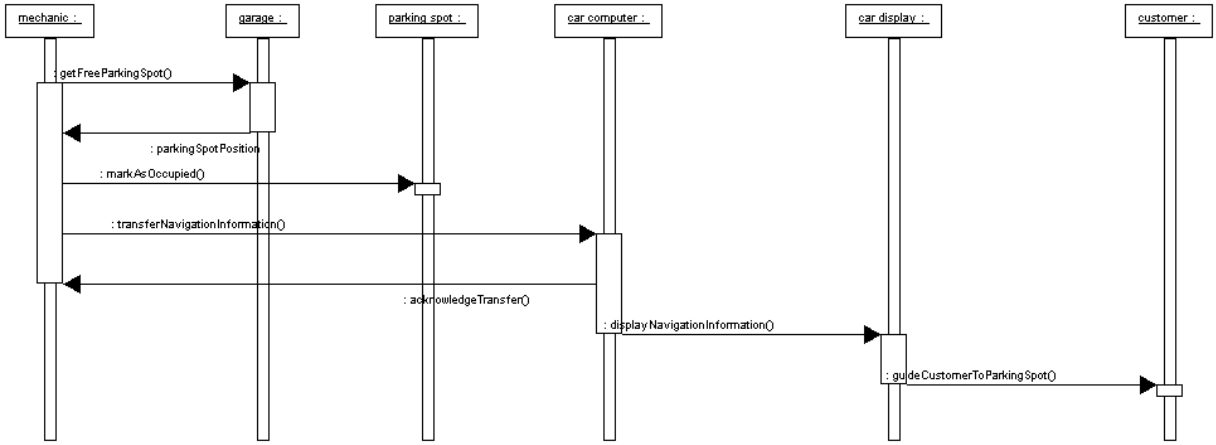
Quality Constraints:

available service resources must be known
- didactic quality of route discription

transfer method must be based on common standards

Used Services:

- Assign Parking Spot
- Transfer Navigation Information



3.1.9. Handover maintenance task

Sequence Diagram

Initiating Actor:

- Customer Representative

Participating Actors:

- Customer Representative
- Mechanic

Realized User Task:

- Perform maintenance at dealership

Flow of events:

1. - *Actor* - The A:Customer Representative instructs the system to handover a running job without taking care of further steps.
2. - *System* - S:Initiate Job Assignment offers several roles specific to this job. (use Initiate Job Assignment)
3. - *System* - S:Notify Mechanic is called for all A:Mechanics to send a notification to receive job acceptance. (use Notify Mechanic)
4. - *Actor* - This step is equal to all notificated A:Mechanics: A notified A:Mechanic receives the notification on his HMD. He takes a certain role by a confirmation for registration.
5. - *System* - S:Get Job Confirmation receives the A:Mechanics confirmation and registers him as busy in the current job. (use Get Job Confirmation)
6. - *System* - The stationary terminal sends all initial information retrieved from the stationary terminal's database to the HMD of the A:Mechanic. The information is specific to the taken role. (use Transfer Information Package)

Exit conditions:

- The A:Mechanic has received all information for his taken role.

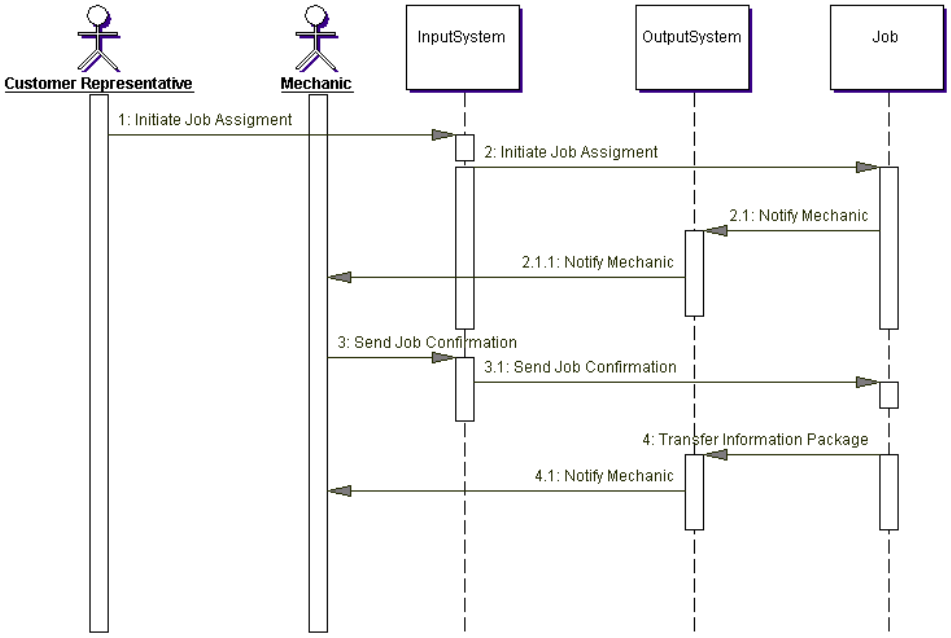
Used Services:

- Get Job Confirmation
- Initiate Job Assignment
- Notify Mechanic
- Transfer Information Package

Open Questions:

- Request for Clarification: What is the Registrationservice good for?
- Inconsistency Issue: change Registrationservice
- Challenge on content: Notification on HMD?

Request for Clarification: No mechanic available right away, or maintenance takes more days??
Request for Clarification: No response to the notification

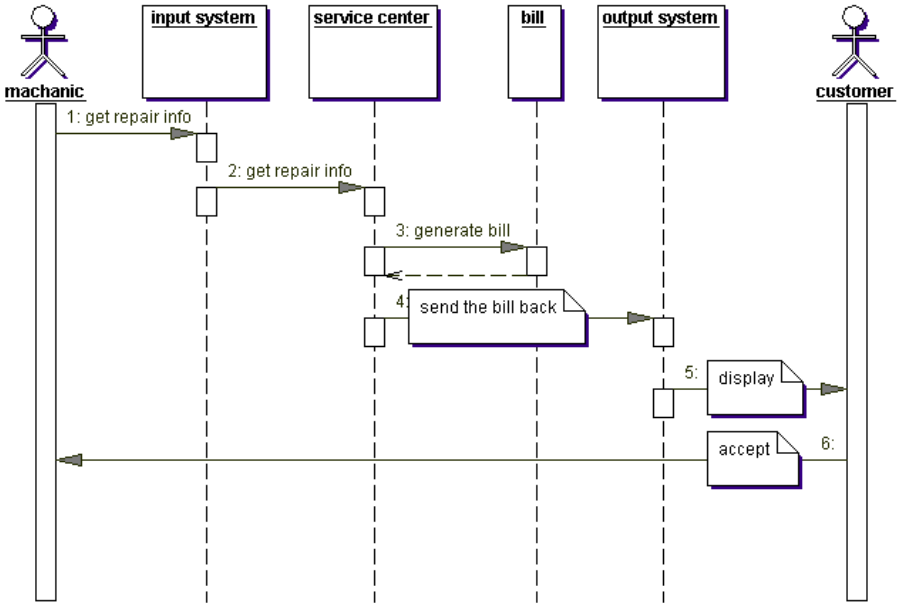


3.1.10. Issue Bill to customer

Sequence Diagram

Initiating Actor:
Mechanic
Participating Actors:
Customer
Mechanic
Realized User Task:
No user task specified.
Flow of events:
1. - *Actor* - The A:Mechanic sends the repair information to the service centre.
2. - *System* - The system receives the repair information and generates bill. (use Get Repair Information)
3. - *System* - The System calculates the total bill and sends it back. (use Generate Bill)
4. - *Actor* - The A:Mechanic displays the bill to the A:Customer with multi-modal output devices.
The A:Customer accepts the bill.

Exceptions:
The transfer is interrupted. The connection should be reset automatically.
Preconditions:
The A:Mechanic completed successfully the repairing procedure of the car.
Exit conditions:
The bill is issued and accepted by the A:Customer.
Used Services:
Generate Bill
Get Repair Information



3.1.11. Perform maintenance at dealership using TRAMP

Use Case Diagram

Initiating Actor:

Customer

Participating Actors:

No participating actors specified.

Realized User Task:

Perform maintenance at dealership

Flow of events:

1. - *Actor* - The A:Customer requests a corrective or preventive maintenance procedure. (include Request maintenance)
2. - *Actor* - The A:Customer Representative assigns the maintenance task to a A:Mechanic. (include Handover maintenance task)
3. - *Actor* - The A:Mechanic locates the car. (include Guide customer to parking spot)
4. - *Actor* - The A:Mechanic performs the maintenance procedure (include Execute procedure)
5. - *Actor* - The A:Customer pays and reclaims his car.

Preconditions:

The A:Customer is at the dealership with his car.

Exit conditions:

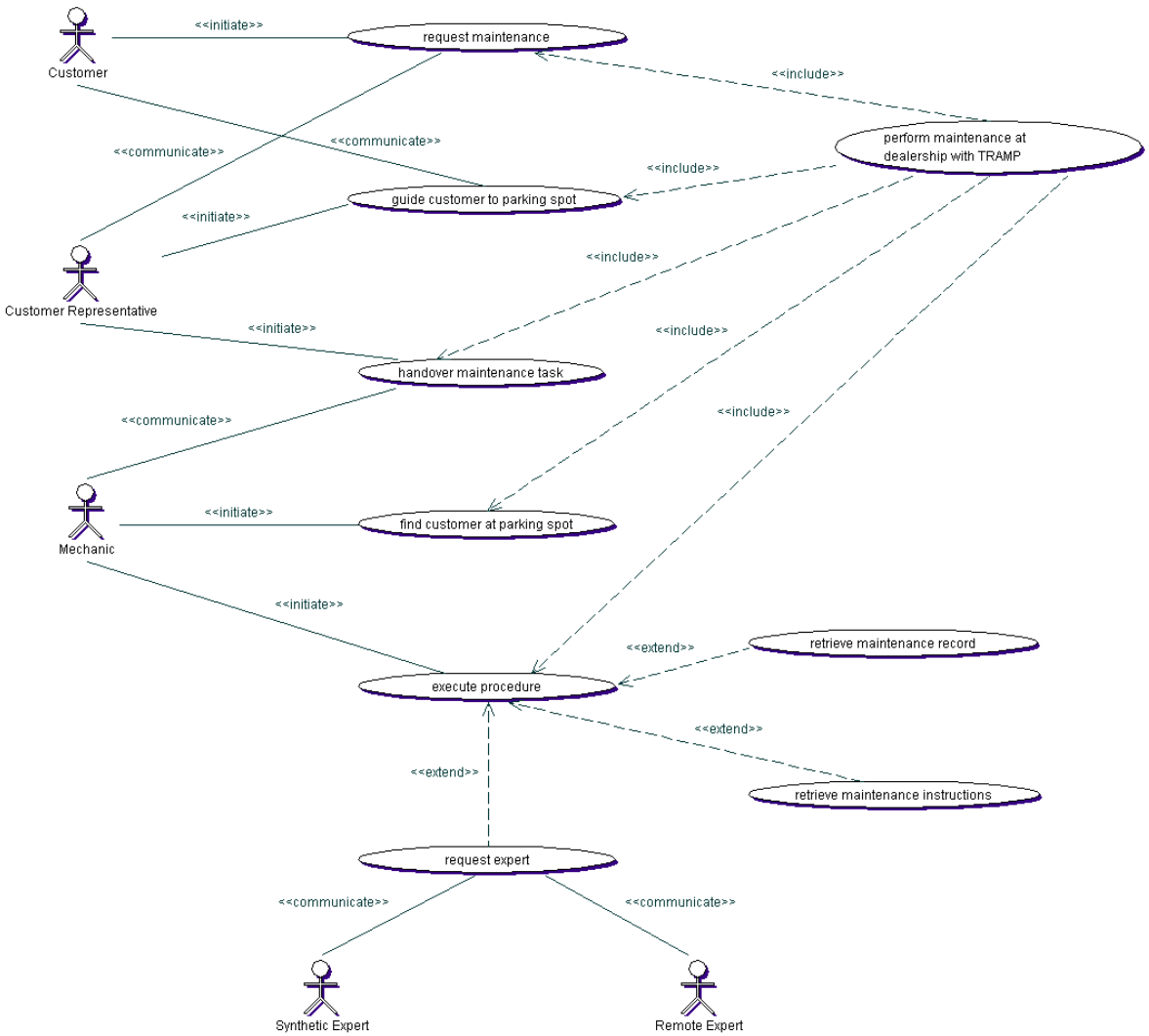
The car is repaired or the maintenance procedure has been completed.

Used Services:

No services specified.

Open Questions:

Omission Issue: find the parking spot
Challenge on content: Should we include "find nearest garage"?


Customer
<<initiate>>
request maintenance
<<include>>
perform maintenance at dealership with TRAMP
<<communicate>>
<<communicate>>
<<include>>
guide customer to parking spot
<<initiate>>
Customer Representative
<<include>>
<<initiate>>
<<include>>
handover maintenance task
<<communicate>>
<<include>>
Mechanic
<<initiate>>
find customer at parking spot
<<initiate>>
<<extend>>
retrieve maintenance record
execute procedure
<<extend>>
<<extend>>
retrieve maintenance instructions
request expert
<<communicate>>
<<communicate>>
Synthetic Expert
Remote Expert

3.1.12. Repair car on road side using TRAMP

Use Case Diagram

Initiating Actor:

- Customer

Participating Actors:

- Customer Representative
- Mechanic

Realized User Task:

- Repair car on roadside

Flow of events:

<

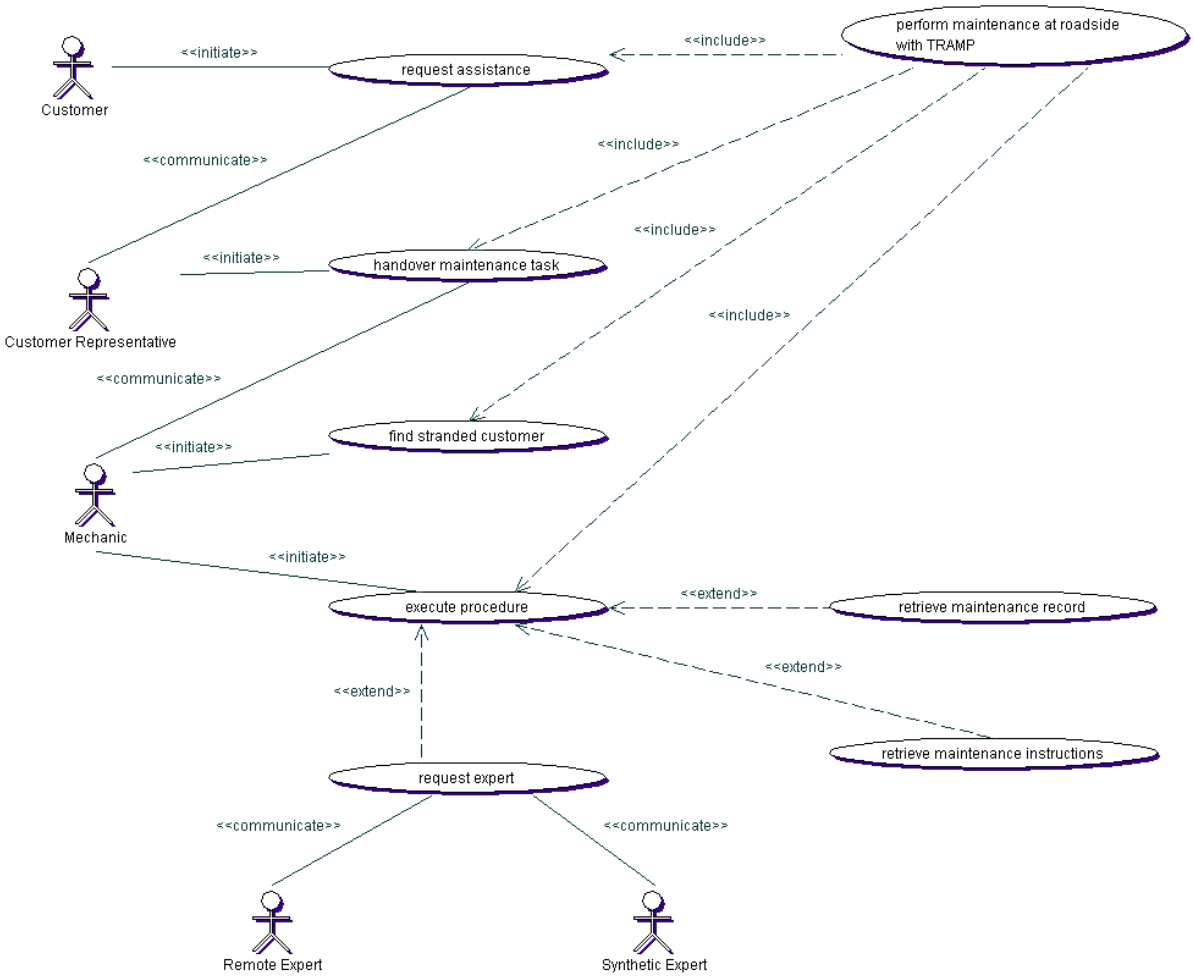
1. - *Actor* - The A:Customer requests assistance from a A:Customer Representative. (include Request assistance)
2. - *Actor* - The A:Customer Representative hands over the task to a A:Mechanic in the vincinity. (include Handover maintenance task)
3. - *Actor* - The A:Mechanic locates the driver who requested assistance and his car. (include Find stranded customer)
4. - *Actor* - The A:Mechanic performs a repair procedure. (include Execute procedure)
5. - *Actor* - If the car is successfully repaired, the A:Customer pays and drives away. Otherwise, the car is towed to the dealer ship. (include Issue Bill to customer)

Exit conditions:

- The car is repaired or the car is at the dealership.

Used Services:

- *No services specified.*



3.1.13. Request assistance

Sequence Diagram

Initiating Actor:
- Customer

Participating Actors:
- Customer
- Customer Representative

Realized User Task:
- Repair car on roadside

Flow of events:

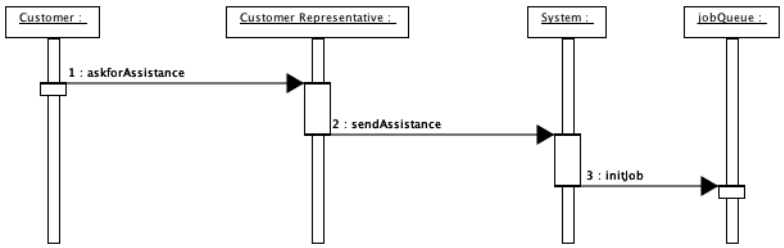
1. - *Actor* - The A:Customer contacts the A:Customer Representative to ask for assistance.
2. - *Actor* - The A:Customer Representative fills out a form with the data retrieved from the A:Customer.
3. - *System* - S:Initiate Job creates a new job and puts it into a queue. (use Initiate Job)

Exit conditions:
- A new job is created.

Quality Constraints:
- Transfer method must be based on common standards

Used Services:
- Initiate Job

Open Questions:
- Inconsistency Issue:
- Inconsistency Issue: The Jobpoolservice is redundant
- Inconsistency Issue: Do "request assistance" and "find nearest garage" overlap?
- Challenge on content: "Sync service" etc. should not be in use case
- Inconsistency Issue: How does the customer know the garage
- Inconsistency Issue: No need to enter payment info



3.1.14. Request expert

Sequence Diagram

Initiating Actor:
- Mechanic

Participating Actors:
Mechanic
- Remote Expert

Synthetic Expert

Realized User Task:
- Repair car on roadside

Flow of events:
1. - *Actor* - A:Mechanic activates the A:Remote Expert.
2. - *System* - Receive the request, connect to a stationary server. (use Receive Remote Expert Request)
3. - *Actor* - A:Mechanic inputs the problem.
4. - *System* - system searches in the database of the server(Synthetic
- Expert) for information regarding the A:Mechanic´s question. If there is suitable information available, the server transmits the repair instructions to the A:Mechanic. If there are no instructions available, the system connects the A:Mechanic with a A:Remote Expert. (use Deliver Remote Expert)
5. - *Actor* - A:Mechanic repairs the car in accordance with the information which helps him to solve the problem.

Exceptions:
- The wearable does not work. (Connection can not be established or the wearable crashes.)

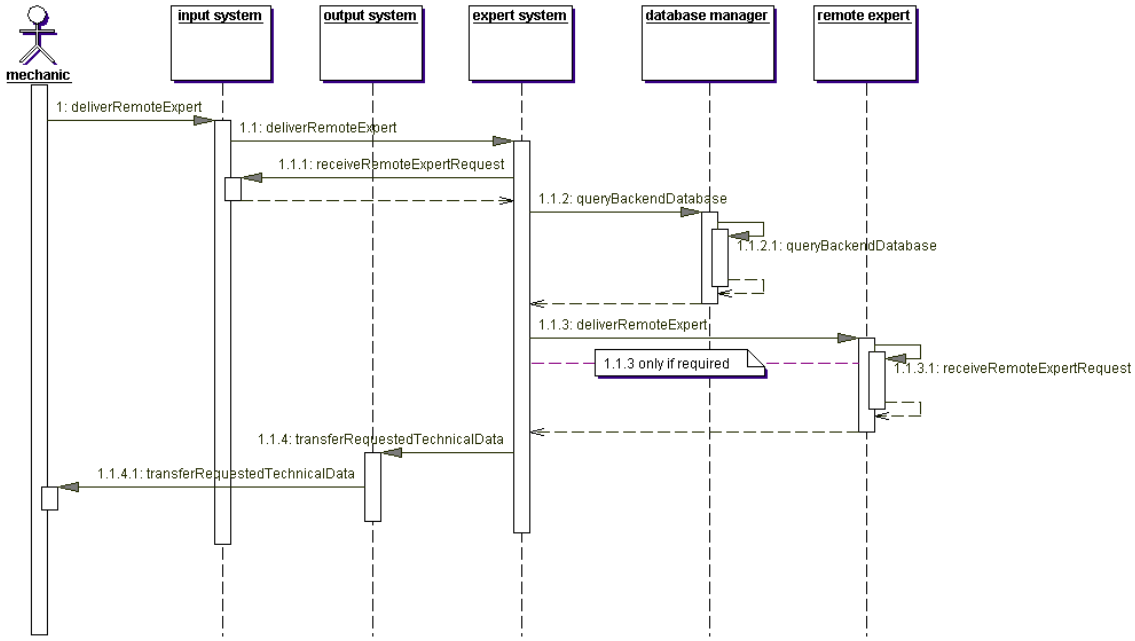
Preconditions:
1. The A:Mechanic found the A:Customer.

1.5 The A:Mechanic has a maintenance request.

2. The A:Mechanic checked the car and found the problems that s/he could not solve by her/hiself.

Exit conditions:
- 1. The problems are solved.
2. The problems can not be solved at this moment,

Used Services:
- Deliver Remote Expert
- Receive Remote Expert Request



3.1.15. Request maintenance

Sequence Diagram

Initiating Actor:
- Customer

Participating Actors:
- Customer
- Customer Representative

Realized User Task:
- Perform maintenance at dealership

Flow of events:

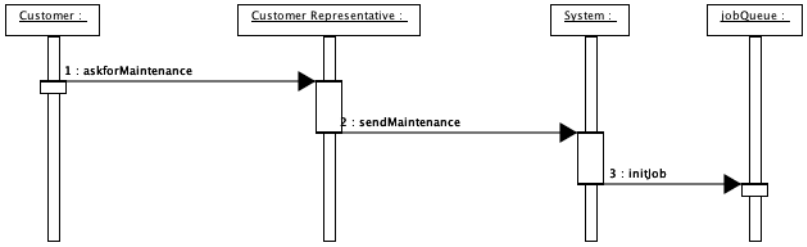
1. - *Actor* - The A:Customer contacts the A:Customer Representative to ask for maintenance.
2. - *Actor* - The A:Customer Representative fills out a form with the data retrieved from the A:Customer.
3. - *System* - S:Initiate Job creates a new job and puts it into a queue.

Exit conditions:
- A new job is created.

Used Services:
- *No services specified.*

Open Questions:

Inconsistency Issue:
Inconsistency Issue: The Jobpoolservice is redundant
Challenge on content: "Sync service" etc. should not be in use case

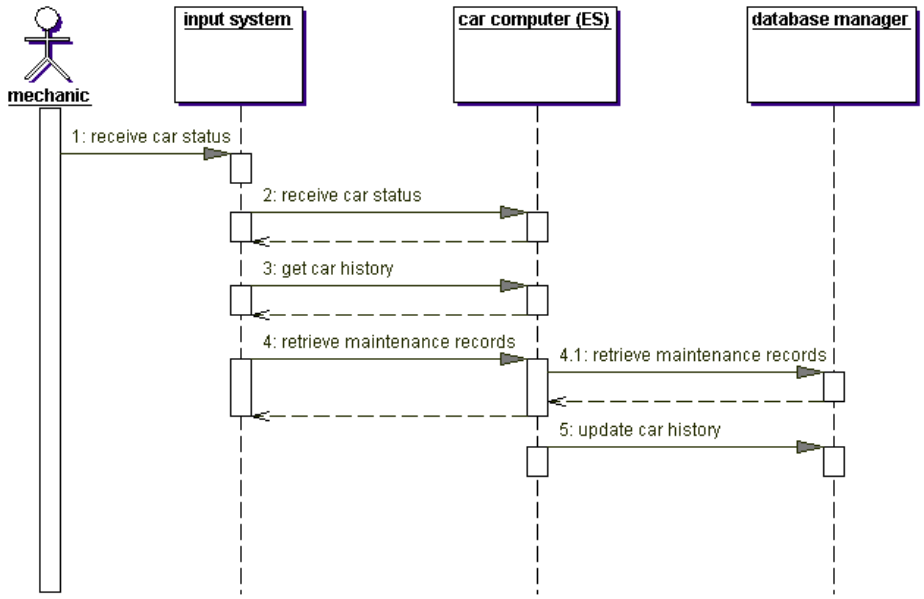


3.1.16. Retrieve Status from Embedded System

Sequence Diagram

Initiating Actor:
Mechanic
Participating Actors:
Mechanic
Realized User Task:
No user task specified.
Flow of events:
1. - *Actor* - A:Mechanic opens engine hood and plugs in his wearable computer
2. - *System* - Embedded system (ES) recognizes the connection request, identifies the authorized A:Mechanic and establishes connection. The car status has to be sent to the TRAMP system. (use Receive Car Status)
3. - *Actor* - A:Mechanic queries ES for system status data including present maintenance status.
4. - *System* - ES returns the requested information (use Get Car History)
5. - *Actor* - A:Mechanic synchronizes the retrieved data with stored maintenance records, disconnects from the ES and unplugs his wearable. (include Retrieve maintenance records)
6. - *System* - The car history has to be updated. (use Update Car History)

Exceptions:
Connection to the car's embedded system fails due to breakdown of the electronic system
Preconditions:
A:Mechanic gains access to the car's embedded system
Exit conditions:
A:Mechanic has retrieved all data concerning car status and car maintenance status from the embedded system and has synchronized it with stored maintenance records
Used Services:
Get Car History
Receive Car Status
Update Car History
Open Questions:
Omission Issue: Wireless access?
Challenge on content: Is ES part of TRAMP or another actor



3.1.17. Retrieve maintenance Instructions

Sequence Diagram

Initiating Actor:

- Mechanic

Participating Actors:

- *No participating actors specified.*

Realized User Task:

- Perform maintenance at dealership

Flow of events:

1. - *Actor* - A:Mechanic stands in front of A:Customer's car, activates the "Receive maintenance instructions" function of his HMD.
2. - *System* - Receive maintenance instructions form is shown in the HMD of A:Mechanic, which prompts A:Mechanic to input the part to be repaired via speech recognition or directly using the wearable´s button based input device. (use Send Maintenance Instruction Form)
3. - *Actor* - A:Mechanic fills in and submits the necessary information.
4. - *System* - The system receives the filled in request form. (use Receive Maintenance Instruction Form)
5. - *System* - The system retrieves the needed informations from a background database. (use Query Backend Database)
6. - *System* - The system sends the requested maintenance instruction to the A:Mechanic (use Transfer Requested Technical Data)

Exceptions:

- The transfer of the maintenance instructions is interrupted due to an unknown reason(e.g. electronic magnetic-field effect in the garage). A:Mechanic changes his position, and presses the "resume transfer" button in his HMD.

Preconditions:

- A:Mechanic is on the job repairing the A:Customer 's car and decides he needs additional instructions not present on his wearable.

Exit conditions:

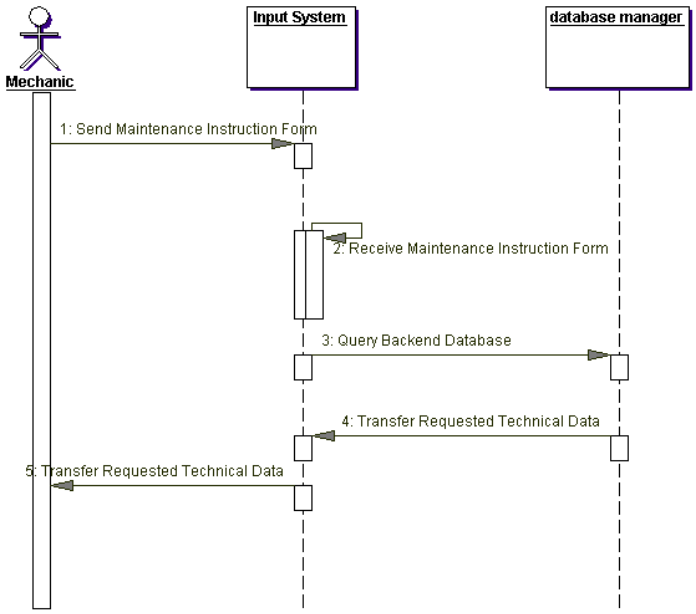
- A:Mechanic has received the maintenance instructions of the part to be repaired.

Used Services:

- Query Backend Database
- Receive Maintenance Instruction Form
- Send Maintenance Instruction Form
- Transfer Requested Technical Data

Open Questions:

- Challenge on content: Why resume manually?

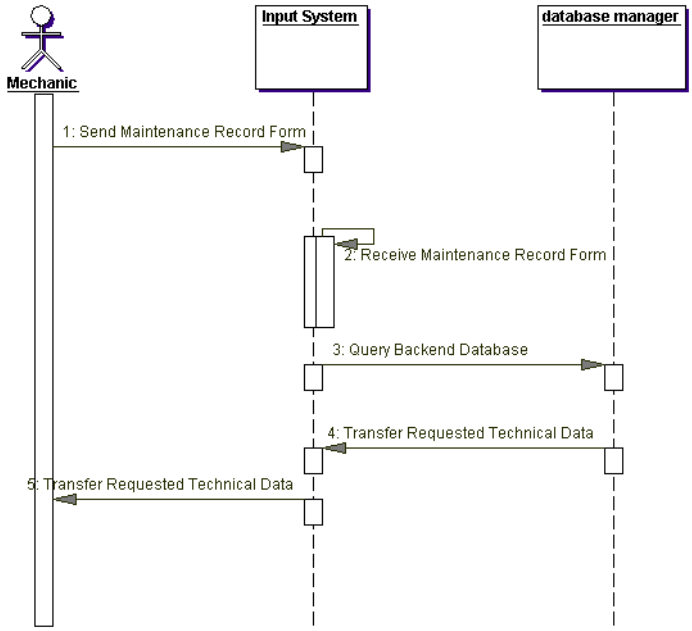


3.1.18. Retrieve maintenance records

Sequence Diagram

Initiating Actor:
Mechanic
Participating Actors:
No participating actors specified.
Realized User Task:
Perform maintenance at dealership
Flow of events:
1. - *Actor* - A:Mechanic stands in front of A:Customer's car, activates the "Receive maintenance records" function of his HMD.
2. - *System* - UC:Retrieve maintenance records form is sent to the A:Mechanic, which prompts A:Mechanic to input the A:Customer's car ID (also manufacturer, model, production date etc.). (use Send Retrieve Maintenance Record Form)
3. - *Actor* - A:Mechanic fills out and submits the form.
4. - *System* - The system receives filled in Retrieve maintenance record form. (use Receive Maintenance Record Form)
5. - *System* - The system queries a backend database to obtain the requested data. (use Query Backend Database)
6. - *System* - The sysem sends the requested maintenance records to the A:Mechanic. (use Transfer Requested Technical Data)

Exceptions:
The transfer of the maintanence records is interrupted due to an unknown reason(e.g. electronic magnetic-field effect in the garage). A:Mechanic changes his position, and presses the "resume transfer" button in his HMD.
Preconditions:
A:Mechanic decides to manually retrieve the maintenance records for a car (e.g. when starting to repair an unknown car on the roadside)
Exit conditions:
A:Mechanic receives the maintenance records of A:Customer's car.
Used Services:
Query Backend Database
Receive Maintenance Record Form
Send Retrieve Maintenance Record Form
Transfer Requested Technical Data
Open Questions:
Challenge on content: Why resume manually?



3.2. Services

Services are features that the system provides that can be used to realize use cases.

3.2.1. Assign Parking Spot

Description:

assigns unoccupied parking spot to system sensitive to the "no parking spots available" exception

Inputs:

Available parking spots (from the world model)

Outputs:

- Destination coordinates for the Shortest Path service
- mark parking spot as occupied

Use cases:

Guide customer to parking spot

3.2.2. Close Taskflow

Description:

When the A:Mechanic is finished with the job he sends a message to the system, which responds with ending the taskflow.

Inputs:

The A:Mechanic finishes the repair process by clicking on a "finalize" link.

Outputs:

The system then sends a message "taskflow finished" to be displayed.

Use cases:

Execute procedure

3.2.3. Deliver Remote Expert

Description:

The A:Mechanic calls for assistance, then a connection to a A:Remote Expert is brought up, they can solve problems via direct talk. The connection is closed by the expert.

Inputs:

Via speech recognition or a special key on a tactil input device, the A:Mechanic selects the function "get A:Remote Expert".

Outputs:

- The system opens a speech connection to an expert.

Use cases:

- Request expert

3.2.4. Detect Marker

Description:

- This service continually scans for a given marker and notifies a given object the marker is detected

Inputs:

- Marker to find
- Object to notify when the marker was detected

Outputs:

- Notifies an object

Use cases:

- *No use cases specified.*

Open Questions:

Omission Issue: Don't you want to include the service "Detect Mark

3.2.5. Display Calibration

Description:

- This service displays a calibration pattern on the HMD which the A:Mechanic is to align with some real-world object. Therefore, some information about the A:Mechanic's position is neccessary in order to know which objects are nearby.

Inputs:

- approximate position of the A:Mechanic

Use cases:

- Calibrate system

3.2.6. Display Taskflow

Description:

- The software system sends the instruction set to the display device.

Use cases:

- *No use cases specified.*

3.2.7. Enlarge Map

Description:

- The user can enlarge the map via voice recognition "zoom in" or by pushing a button "zoom in"

Use cases:

- Find customer at parking spot
- Find stranded customer

3.2.8. Find Shortest Path

Description:

- Retrieves the shortest path from the starting point to the destination

Inputs:

- Starting Point
- Destination

Outputs:

- shortest path from starting point to destination

Use cases:

Find Nearest Garage

- Find customer at parking spot

Find stranded customer

3.2.9. Generate Bill

Description:

- Using the retrieved repair information the system calculates the costs and generates the bill Perhaps displayed in a form of a list of repair elements and

single and final price to be payed

Use cases:

- Issue Bill to customer

3.2.10. Get Calibration Input

Description:

Retrieves user feedback during the calibration process.

Use cases:

- Calibrate system

3.2.11. Get Car History

Description:

The embedded system puts the information of the A:Customer's car history onto the display of the A:Mechanic's wearable computer.

Use cases:

- Retrieve Status from Embedded System

Open Questions:

Challenge on content: Is ES part of TRAMP or another actor

3.2.12. Get Job Confirmation

Description:

The service accepts a A:Mechanics confirmation followed by a role offer and sets his state to 'busy'.

Inputs:

- A:Mechanics confirmation

Use cases:

- Handover maintenance task

3.2.13. Get Payment Confirmation

Description:

No description specified.

Use cases:

No use cases specified.

3.2.14. Get Repair Information

Description:

System retrieves information from the A:Mechanic about used time and spare parts

Use cases:

- Issue Bill to customer

3.2.15. Get User Position

Description:

This service tracks the position of the user. It uses the data of several tracking devices. If some of them are not available, the position is interpolated as good as possible.

Inputs:

- Data from several tracking devices attached to the user.

Outputs:

The current location and direction of the user in world coordinates.

Quality Constraints:

- tracking speed

Use cases:

- Find Nearest Garage
- Find stranded customer

3.2.16. Initiate Calibration

Description:

This service initiates the calibration.
It is the job of the UI-team to provide a UI in which perhaps shows the user the callibration in form of a x-y axis display and allows to alter the coordinates

Use cases:

- Calibrate system

3.2.17. Initiate Job

Description:

put a new job into the jobqueue.

Inputs:

a new maintenance request from a A:Customer

Outputs:

a new job

Use cases:

- Request assistance

3.2.18. Initiate Job Assignment

Description:

The service retrieves information about available A:Mechanics for job assignment.

Outputs:

list of available A:Mechanics

Use cases:

- Handover maintenance task

3.2.19. Initiate Taskflow

Description:

A:A:Mechanic starts the taskflow that corresponds to the set of instructions that is needed to perform the current maintenance job.

Use cases:

- Execute procedure

3.2.20. Log-in SPOT for repair-process by mechanic

Description:

*log-in data of A:Mechanic is processed by SPOT

* UC is started

* UC is started

Inputs:

name
password

Outputs:

IETM-Instructions
IETM-Records

Use cases:

No use cases specified.

3.2.21. Navigate Mechanic

Description:

This service uses that shortest path from S:Find Shortest Path and generates a updated map from this information.

A map (with gesture guiding or marking with a shortest route to the location of A:Customer) will be shown on the HMD(or laptop).

Inputs:

shortest path (derived from service with the same name)

Outputs:

updated map

Use cases:

- Find customer at parking spot
- Find stranded customer

Open Questions:

Request for Clarification: whz does this service exist?

3.2.22. Notify Mechanic

Description:

After the Server recieves a request from A:Customer-Representive , a message will be sent to a A:Mechanic who is

free at that moment.
This services should be inluded into UC:Request expert

Inputs:

1.location of the A:Customer,
2.the description of problems about the car

Outputs:

1. Taking this job or not
2. the estimated arrival time.

Use cases:

No use cases specified.

3.2.23. Notify Mechanic

Description:

The service sends a given message to a A:Mechanic

Inputs:

address of A:Mechanic, notification message, need of confirmation, if so, confirmation url

Use cases:

Handover maintenance task

3.2.24. Query Backend Database

Description:

This service provides a generic access to a database with all necessary background data

Use cases:

Retrieve maintenance Instructions
Retrieve maintenance records

3.2.25. Receive Car Status

Description:

The TRAMP system gets the information about the A:Customer's car status through the embedded system on the car.

Use cases:

Retrieve Status from Embedded System

Open Questions:

Challenge on content: Is ES part of TRAMP or another actor

3.2.26. Receive Cash Payment Notification

Description:

The server receives the cash payment notification (including name, adress, unique bill number and the amount of money) from the A:Mechanic.

Use cases:

Cash-Payment

3.2.27. Receive Credit Card Data

Description:

Information regarding the type and number of the credit card, and the name of the owner and so on is retrieved for further verification (correctness, A:Customer´s ability to pay,...)

Use cases:

E-Payment

3.2.28. Receive Maintenance Instruction Form

Description:

The system receives the form requesting special maintenance instruction filled in by the A:Mechanic

Use cases:

Retrieve maintenance Instructions

3.2.29. Receive Maintenance Record Form

Description:

System receives filled form with unique car identifiers to request the car's individual history from the backend database

Use cases:

- Retrieve maintenance records

3.2.30. Receive Remote Expert Request

Description:

A stationary server receives an expert request, creates a connection to the client and forwards the request data to the A:Synthetic Expert.

Use cases:

- Request expert

3.2.31. Register a customer

Description:

- Register a new A:Customer

Inputs:

- Datas of the A:Customer and of his car

Use cases:

- *No use cases specified.*

3.2.32. Send Cash Payment Form

Description:

A payment form will be displayed. The form contains only information necessary for payment, i.e. total costs, name and adress of A:Customer and a unique bill reference

Use cases:

- Cash-Payment

3.2.33. Send Maintenance Instruction Form

Description:

- The system offers the A:Mechanic the choice which special maintenance instruction he want to request

Use cases:

- Retrieve maintenance Instructions

3.2.34. Send Payment Confirmation

Description:

A payment confirmation is sent in the case, the A:Customer´s payment information is checked for correctness and validated. This informs the A:Mechanic of the status of the payments made.

Use cases:

- Cash-Payment
- E-Payment

3.2.35. Send Retrieve Maintenance Record Form

Description:

- With this form the A:Mechanic can describe the identification of the car's properties to obtain the individual history of the car

Use cases:

- Retrieve maintenance records

3.2.36. Transfer Information Package

Description:

- The service transfers job information

Use cases:

- Handover maintenance task

3.2.37. Transfer Navigation Information

Description:

- Navigation information from the Shortest Path service must be transfered to a recipient not equipped with special hardware

Inputs:

- Shortest Path from starting point to destination

Outputs:

navigation information in a standard comprehensible and

accessible to the recipient (different representations should be possible)

Use cases:

Guide customer to parking spot

3.2.38. Transfer Requested Technical Data

Description:

Generic service to transfer to deliver requested technical information to the A:Mechanic

Use cases:

Execute procedure

Retrieve maintenance Instructions

Retrieve maintenance records

3.2.39. Update Car History

Description:

The embedded system updates the information of the A:Customer's car history, when the A:Mechanic has finished his job.

Use cases:

Retrieve Status from Embedded System

Open Questions:

Challenge on content: Is ES part of TRAMP or another actor

3.2.40. Validate Payment Data

Description:

After positively checking for correctness of the retrieved data, the system validates it.

Use cases:

E-Payment

3.2.41. send E-payment form

Description:

A payment form will be displayed. The most important fields are name, credit card type, credit card number and so on.

Use cases:

E-Payment

3.3. Global Functional Constraints

Functional constraints are constraints on the interactions between the user and the system.

3.3.1. Multimodality

Type:

Functional Constraint

Description:

The user interacts with the wearable system using different modalities, e.g. voice recognition and synthesis, three-dimensional maps on a head-mounted display, etc.

3.3.2. Nonfunctional requirements

Type:

Functional Constraint

Description:

Non-functional requirement describes the system organization including user interface, performance, and so forth.If we talk about system, we must think about what we can offer to satisfy the demands of the customers. The main function of non functional requirements is to find out how the system should be organized and which exspectations it should meet.

Following aspects will be discussed:

User :

There are at least two kinds of users of the system :

The first one is the person who has a problem with his or her car and lastly have to fill a form with this problem and in this way to cause automatical assignment of this problem to the next available A:Mechanic.

The second one is A:Mechanic who gets a call for help and who finally will repair the car with a help of this system (eventually with the expert) . So it is neccessary to have two different user-inteface . One for the person sitting in the center can just use a conventional PC . One for the A:Mechanic who have to take a portable machine which should be rather small ,as well the hardware as the Software.The software should be to intuitive,because the A:Mechanic have little time to understand the introduce of the software , he must concentrate on the repairing.The instruction given by the system should be as small as possible,so that the A:Mechanic know what he should do.

System :

The system , at least that part of it which will accompany A:Mechanic , should work in a wireless way. It must provide the possibility to request explicit information , such as wheels from a data base.So that the part of the system, at least that part of it which will accompany mecanics, should work in a wireless way. It should provide the possibility to request precise information e.g. about wheels from a data base. So that the part of system which will be used by mecanic will have only very small memory, but thanks to requesting ability mecanic will be able to have the information he/she needs within few seconds, in the worst case few minutes.And the useinterface should be multimodal, so that the A:Mechanic can use different modalities like voice recognition ,three dimensional maps, etc.The system must be very robust,because it is used by the A:Mechanics in the bad condition and for long time .It will be very stressed.

Documentation :

There should be a technical documenttation for maintainer and even the documentation of the development process.

Language :

The language prefer to English.

Error handling and extreme conditions :

Like any other system it must be errorfree.The sent informations between the A:Mechanic und the center must be exactly correct, because small mistake will cause to great catastrophes.

Security :

Because the system is only used by the A:Mechanic,it should be protected with the password against malicious user.

Resource :

The battery should work as long as possible,
because the A:Mechanic can not leave his work all minutes to
change the battery.

There are some more suggestions:

What level of document?
Interaction with other hardware systems?
How should the system handle exceptions and which?
What is the worse environment in which the system is
expected to perfom?
Should the system be protected agains external intrusions?

3.3.3. Reconfigurability

Type:

Functional Constraint

Description:

The user should be able to reconfigure his wearable system on the fly to accomodate different operating environments.

3.4. Quality Constraints on Use Cases

Constraints the use cases have to meet.

3.4.1. Performance of AR-related tasks

Type:

Quality Constraint on Use Case

Description:

The system should be able to calculate and display AR-information (e.g. in the HMD) in realtime.

Constrained Use Cases:

Find customer at parking spot

3.4.2. available service resources must be known

Type:

Quality Constraint on Use Case

Description:

it must be possible to find out quick, if there is an parking spot and a maechanic available.
If not it should be possible to axplore alternatives
in the worst case, another garage must be recomended
to the A:Customer

Constrained Use Cases:

Guide customer to parking spot

3.4.3. calibration duration

Type:

Quality Constraint on Use Case

Description:

the callibration should not take up to much time and should be easy to handle. It is not the main task of the user to calibrate his/her wearable.

Constrained Use Cases:

Calibrate system

3.4.4. calibration necessity

Type:

Quality Constraint on Use Case

Description:

calibration should only be necessary in extreme cases:
-changed user with other requirements
-changed environment conditions (light, temperature,...)

-very long and extensive usage since the last calibration

Constrained Use Cases:

- Calibrate system

3.4.5. didactic quality of route discription

Type:

- Quality Constraint on Use Case

Description:

if the electronic transfer to the custumers navigation system fails, the information must be passed to himself.
This could be realized by a verbal discription, or a printed map. This also depends on complexity of the route the user has to take.
Since the user is not familiar with the location, and is not used to the computersystems used by the garage employes, the informations must be presented to him in a commonly understandable, detailed, but not too overloaded manner.

Constrained Use Cases:

- Guide customer to parking spot

3.4.6. transfer method must be based on common standards

Type:

- Quality Constraint on Use Case

Description:

the method used to transfer the parking spot location to the A:Customer's car computer or navigation system should be available to as many A:Customers as possible.
Therefore it is necessary to inplement a commonly used standard, or if not available to support as many different systems as possible.

Constrained Use Cases:

- Guide customer to parking spot
- Request assistance

3.5. Quality Constraints on Services

Constraints the services have to meet.

3.5.1. tracking speed

Type:

- Quality Constraint on Service

Description:

The position of the user should always be up to date. This means several different position data per second must be available. The lag between the real movement and the position data may not be to big.

Constrained Services:

- Get User Position

4 . Examples

This section provides examples of the system in use in terms of actor instances and scenarios. These serve an illustrative purpose and support the discussion of details related to actors, user tasks, and use cases described in previous sections.

4.1. Actor Instances

Examples for the identified actors.

4.1.1. Anton

Actor:

- Customer

Description:

- *No description specified.*

4.1.2. Brandon

Actor:

- Mechanic

Description:

- *No description specified.*

4.1.3. John

Actor:

- Customer

Description:

- *No description specified.*

4.1.4. Manfred

Actor:

- Mechanic

Description:

- *No description specified.*

4.1.5. Toni

Actor:

- Customer Representative

Description:

- *No description specified.*

4.2. Scenarios

- Two possible scenarios for the use of the system.

4.2.1. Car Maintenance Scenario

Instantiated Use Case:

- *No use case specified.*

Initiating Actor Instance:

- *No actor instance specified.*

Flow of events:

A:John's head light turn signal does not work anymore, so he decides to go to a nearby garage.

When he arrives there, A:Toni, the A:Customer Representative at the reception enters the problem into his wearable computer, Spot. A:Toni wears a head mounted display and uses speech recognition and Inmedius' Wheel/Pointer to interact with his wearable computer.

A:Toni is advised by his wearable to reproduce the failure. A:Toni lets A:John sit in his car and activate the turn signal. It does not work.

Spot displays the following advice: "Let the A:Customer drive the car to parking lot 235 where the A:Customer should meet a A:Mechanic." The A:Customer drives to lot 235.

Meanwhile A:Brandon, a A:Mechanic who is inside the garage, gets a notification (via wireless ethernet): "Show up at lot 235". A:Brandon also receives repair instructions as an IETM (interactive electronic technical manual).

A:Brandon puts necessary spare parts into his toolbox and goes to the parking-lot guided by navigation information displayed in his HMD. At the parking lot A:John is already

waiting.

A:Brandon first checks the fuse-box following the steps automatically displayed inside his HMD. When he opens the fuse box - which is automatically detected by the optical tracker in his wearable system, the next instruction is displayed: "Check the fuse number 3123". The fuse is OK, so a new set of instructions starts to check the signal

A:Brandon then checks the lamp of the turn signal. He finds out that the lamp is blown, so he replaces it and checks whether the new one works (it does).

A:Brandon enters the payment information of A:John into his wearable (speech) and transmits the information via UMTS.

4.2.2. UMTS Scenario

Instantiated Use Case:

No use case specified.

Initiating Actor Instance:

No actor instance specified.

Flow of events:

Car owner A:Anton is driving on the Autobahn. His car breaks down with a problem that he cannot fix alone. With his UMTS phone, he calls the hotline of his car manufacturer, which notifies the closest workshop.

The workshop sends the A:Mechanic A:Manfred to check A:Anton's car. Some information and data which A:Manfred needs about the car are automatically sent by the car to A:Anton's UMTS phone, which in turn sends them to the hotline and from there it reaches A:Manfred.

From the data A:Manfred concludes that the problem could be caused by the distributor. He therefore decides to get a new distributor from the spare parts room and puts into his repair-bag.

A:Manfred then determines the location of A:Anton's car and finds the shortest route to get there. After his arrival he checks the car and finds that there is indeed a problem with the distributor.

Replacing a distributor is quite a complex operation, and A:Manfred uses his access to UMTS to download the necessary data - drawings, cable labelings, and remove-and repair instructions - to do the replacement.

The sequence of instructions is shown on his head-mounted display.
After the repair is finished, A:Manfred sends the total time and parts used to the workshop which in return sends a bill to A:Manfred via UMTS.

A:Manfred prints out the bill and gives it to A:Anton, who then pays with his credit-card.

5. Analysis

This section contains a model of the TRAMP system, which formalizes the specification produced in the chapters before.

- The first part of the model is the object model, you can find this in the diagram below.
  The sequence and use cases diagrams belong to analysis, too, although you already found them in chapter 3 (specification) due to readability reasons.

5.1. TRAMP Object Model

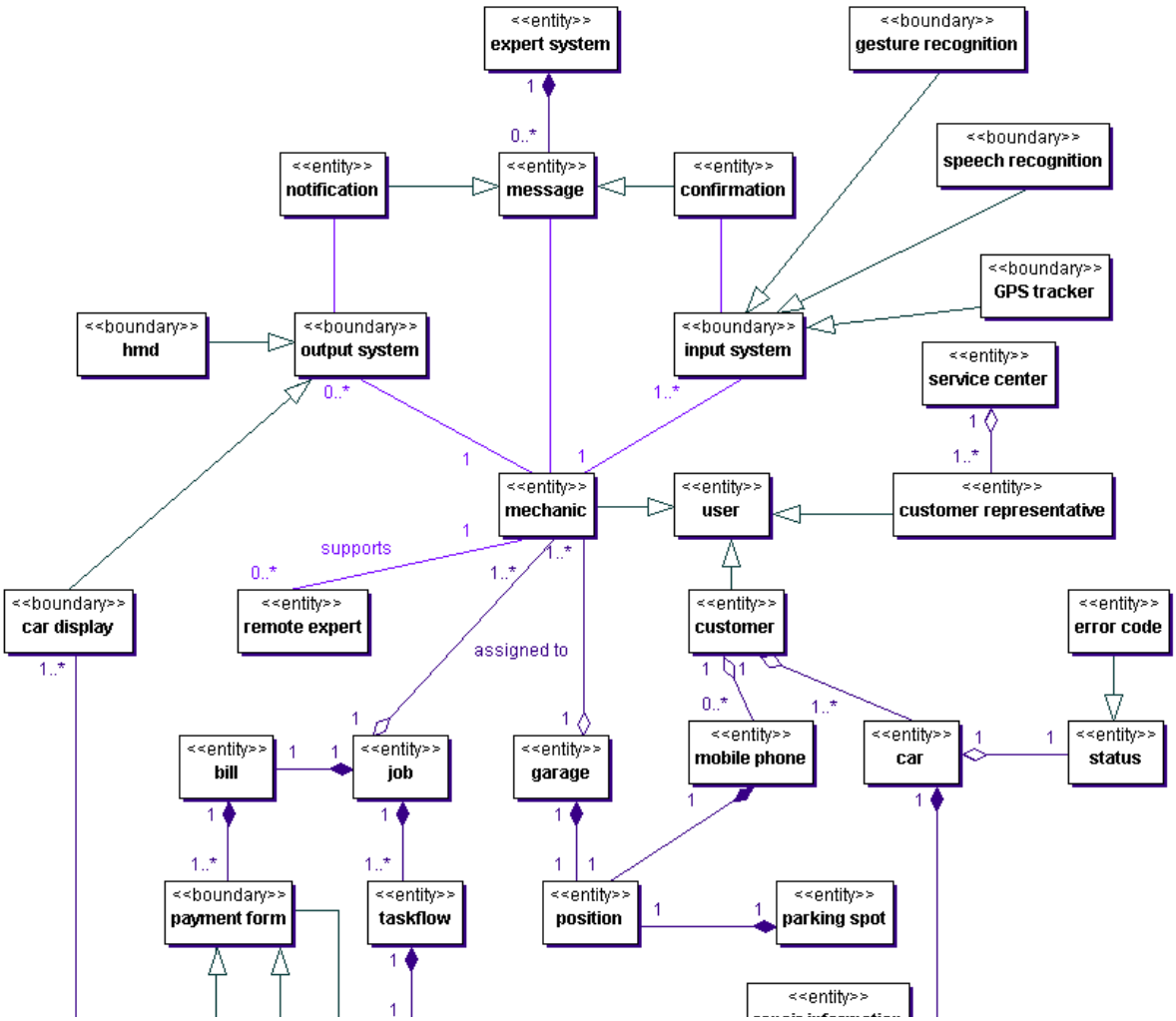
This diagram represents the TRAMP Object Model. It contains all objects identified by examining the system specification and the relationships between them.

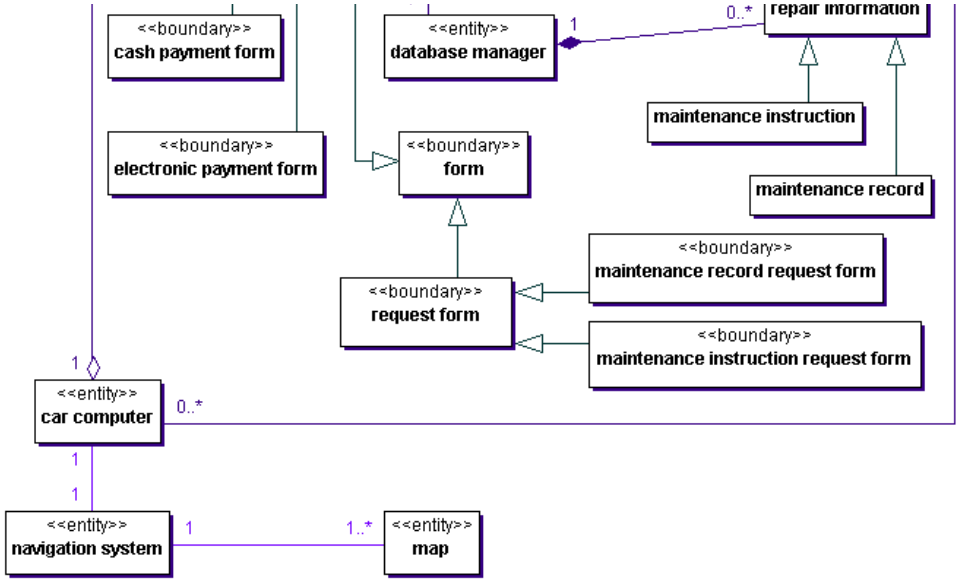
Two different kinds of objects can be found in the model:

- 1. Entity objects represent the persistent information tracked by the system, like real-world entities or activities (e.g. the mechanic, the customer, a single job or a confirmation).

2. Boundary objects represent the interactions between the actors and the system, like input-forms or user interfaces (e.g. the Head Mounted Display, speech recognition or the payment form).

The description of the relationships between the objects includes generalization and multiplicity.





6. Prototypes

In this section, decriptions and pictures of some of the prototypes we created for the Hardware- and User-Interface-Environment can be found.

6.1. Hardware Prototypes

Wearable team 3d prototype specification

Mounted on helmet:

- **HMD:** head mounted all-purpose display (IETMs, GPS maps, room layout, calibration patterns for the mechanic, to notify the mechanics, display status messages)
- **GPS/UMTS-Receiver:** for tracking the user's current position outdoor.□
- **Camera:** for tracking the user's current position indoor.
- **Microphone:** to serve as an input device for the mechanic during his repair activity; to enlarge the map.
- **Inertial tracker:** for gesture recognition for the mechanic during repairing, in loud environments, map navigation

6.2. User Interface Prototypes

UI-Navigation

Step 1: A short information will be sent to the Mechanic who is free at this time.

The short information will include : a) the probably task

b)the Location of the customer

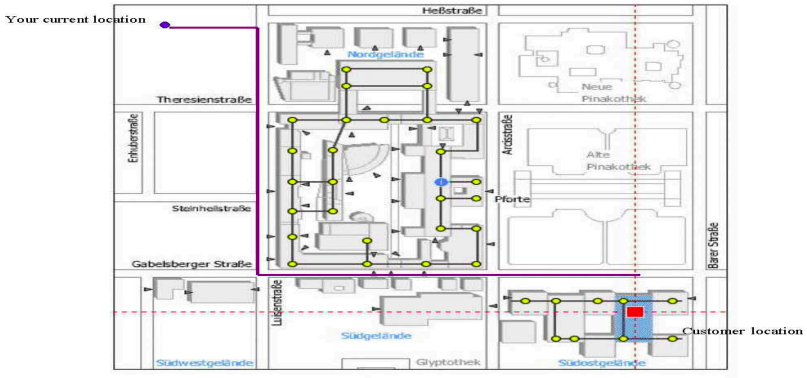
Screenshot : referring to Request Maintenance.

And on the screen of the mechanic's laptop will be shown something like this.

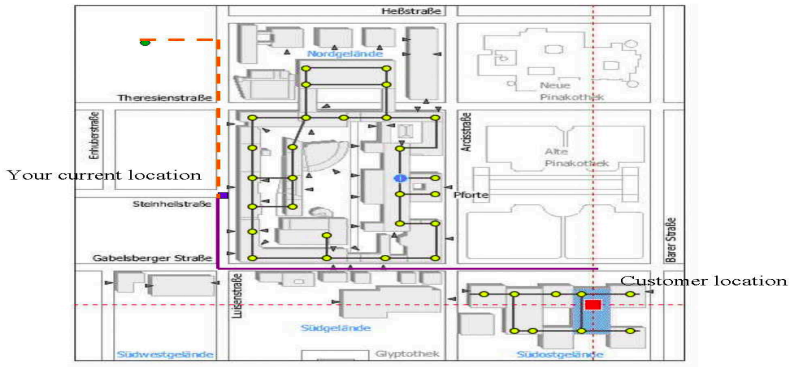
Service Center: Hi, Peter! There is a job for u!

Service Center: Maybe something wrong about engine, it is located in the south of TU-München , please confirm if u are willing to take this task.

Step 2: After Brandon's confirming (including estimated arrival time), the server will guide Brandon to the location by using laptop (HMD).



Step 3: during the way to customer , the screen will be shown like this



Step 4: Arrival

Repair Mock-up

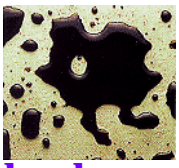
Overall car check

Check Oil
Check Tires
Check Engine
Check Exhaust

Done! - Hit 'Escape' or say "I'm done!"

Oil check

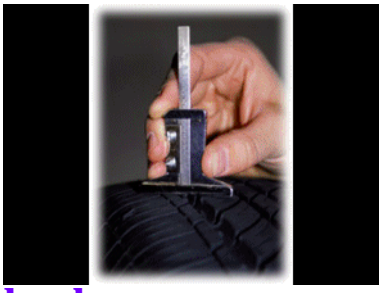
Step 1: (...)



back

Tires check

Step 1: (...)



back

Engine check

Step 1: (...)

back

Exhaust check

Step 1: (...)

back

Repair completed

Off to the cashier!

back

Payment-Prototype

Hardware: iPaq

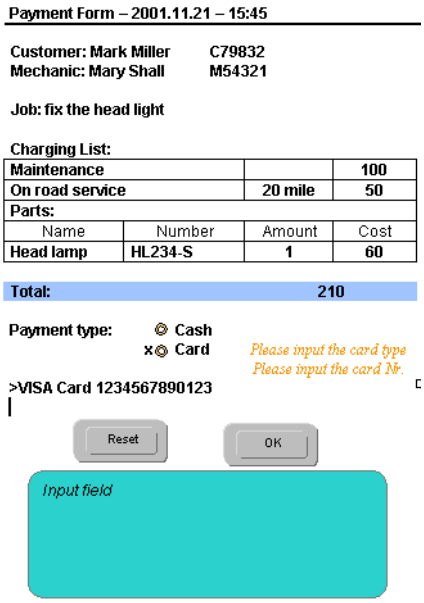
Flow of Events:

1. The mechanic gets a payment form (see the attached Mock- ups: "payment.html")
2. The mechanic checks if everything is right and ask the customer how he would pay for it.
3. The mechanic selects either cash or card
4. If "Card" is selected, the display asks him to write down the card type and number (VISA, MASTER, AE, or Diner?)

=> he writes down the type and the number.

5. The mechanic shows it to the customer to make sure everything is right and then confirms it.
6. The customer checks it and signs it.
7. The system certifies the card, sends a sign request back.
8. The customer signs. The mechanic confirms it and sends it to the system

Payment Complete!



Registration Prototype