Car Navigation System

Assignment for Software Architecting course at Technische Universiteit Eindhoven

Radim Čebiš and Pavel Černohorský

Requirements

Extra functional

Performance

1. It (re)computes route to the desired destination in 3 sec.
2. When information about some traffic problems is received through RDS, it will be checked within 3 seconds whether this new information is relevant for current route and when yes, re-computation will be initialized.
3. It recomputes route in case of diversion from computed route in 3 seconds after diversion is bigger than minimal accuracy (see legend).
4. While entering destination by address, device suggests streets etc. which starts with the entered letters within at most 0.3 seconds.

Usability

5. User can always set the new destination in at most 4 touches (from any system state) not counting touches used for writing.
6. At least 80% of the screen is used for showing a map while system navigates user to some destination.
7. Device is controllable by finger touches of 80% of population.
8. User will be informed about successful screen touch.

Availability

9. While operating using batteries, it will be able to be fully operational for at least 4 hours in the worst case (no matter what functions user uses).
10. After turned on, device has to be operational (except for the navigation capabilities which require GPS module) within at most 5 seconds.
11. After turned on, device has to be fully operational within at most 3 minutes while on open view or 5 minutes while in the city while GPS satellites are working and in range.

Responsiveness

12. Map is refreshed while navigating to destination at 20 frames per second including all interactive input like zoom and changing between top view and 3D view.

Interoperability

13. Maps and points of interest can be updated by standard PC (not older than 5 years) with appropriate operating system without any additional software.
14. Statistics can be downloaded to standard PC (not older than 5 years) with appropriate

operating system without any additional software.

Functional

15. When the route is recomputed because of diversion, the current direction of the car is taken into account and device tries not to force the user to turn around when it is not allowed.
16. User can ask for alternative route. Alternative route is that one where at least 30% of the route will be different (will lead on different roads).
17. Car is kept on the road until its position is more than minimal accuracy far from any road – then the car is placed on the received position.
18. Car (its representation) will be shown every time on the same place (horizontally centred and vertically in the lower one third of the map) while navigating so that user will not have look for it on the map.
19. Voice tells the user about the change of direction right after every change, 1 min. before the change (when there is enough time for it) and then at least 5 seconds before the change according to current speed.
20. Voice instruction contains: distance to the next action, type of action (after 100 meters turn left, turn left, stay in the left lane, use second exit at the roundabout, slow down and similar with different distances and directions).
21. While showing the map, there is a button to replay last voice instruction.
22. User has possibility to switch the system into the night mode when it's luminance will be low enough (not higher than luminance of recent car's control panel with speed meters) not to disturb the user from driving when the system is placed in the car on the place described in the user's manual.
23. User can select which info (time, distance from the target, time to target, distance to the next direction change, next direction change, types of POIs to show on the map, velocity) should be presented all the time together with the map.
24. User can turn on/off the speed warning (by beep sound – the fastest way) which warns of speeding according to limits saved in map.
25. After switching on the last planned route is displayed– so the user can plan the route in advance.
26. User can enter the destination by address, coordinates, touch or selecting from saved positions.
27. User can save any position into the one of 20 save positions slots.
28. User can create at most 10 profiles which will be used for saving trip statistics information (distance, average speed, exact time of the trip). Trip in this context is one travelling on the given destination address.
29. Device will be capable to store statistics of last 10 trips for each user profile.
30. Statistics are saved in human readable text format.

31. When the device is not connected to the electricity source, it shows battery status in the main view (map view).
32. It informs the user about GPS signal strength – and possible inaccuracy. If there is no signal it shows special indication on the same place.
33. There will be enough place to upload the maps of whole Europe in highest currently available accuracy.

Legend:
minimal accuracy = 15 m
maximal accuracy = 3 m
POI = Point of interest

Design decisions

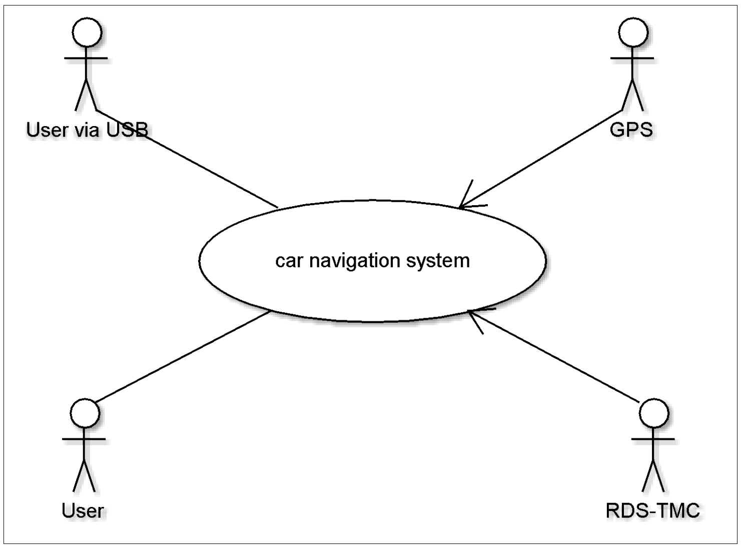
- Path finding algorithm with adaptive accuracy will be used. It will iteratively refine the route maximally until the given time limit is exceeded (in our case 3sec.). Supports requirement 1, 3. Rationale: this algorithm will return at least some result, which will be very close to optimal one.
- For saving the addresses the data structure with lexically ordered items and thus fast access for suggest functionality will be used. Supports requirement 4. Rationale: it is important to give the user fast suggest functionality while he is writing the address.
- In every screen there will be button to get back to main screen (with the map) and from the main screen it is possible to enter address in 3 clicks (not counting clicks used for input) – first click on button to enter the destination, second click to choose the type of the input of destination (i.e. address, coordinates...), third click to confirm the input (i.e. address). Supports requirement 5. Rationale: user must be able to drive a car, thus cannot be overwhelmed by controlling the device. He must have fast access to most often used functions.
- Screen buttons must be at least 1 cm x 1 cm big for the common finger to fit there. Supports requirement 7. Rationale: most of the population have fingers thin enough to be able to control the device with them.
- Device plays the press sound after each touch – confirming the successful input. Supports requirement 8. Rationale: user must be able to switch his focus on the route as soon as possible after the touch, so he cannot check the screen if the order was taken into account. Or sometimes he cannot pay attention to the screen and is pressing the buttons just by memory.
- High capacity batteries and the processor with low power consumption will be used. Supports requirement 9. Rationale: it must be possible to use the device without the charger for shorter journeys.
- Customized operating system will be used for the device (no controls and detection of HW on boot). Supports requirement 10. Rationale: it is not necessary to test or detect the HW, because the HW configuration will be fixed and this way we can provide better start-up times.
- High sensitivity GPS receiver will be used. For example SiRFstar III. Supports requirement 11. Rationale: our device should be as accurate as possible.
- Rendering the screen will be handled by main thread which will have high priority. Supports requirement 12. Rationale: it is important that the screen refresh will not be dependent on other functions.
- Screen refresh rate is lowered to 15 Hz while computing new route. Rationale: We believe that it is useless to have so much computing power to be able to refresh the screen as fast as

in navigating (20 Hz) when the computation of the route is done only few times per hour. We believe that this is better way than investing more money for HW performance which will be so rarely used and the 15 Hz refresh rate is still good enough. We suppose that this is very good trade off.

- Maps and POIs will be updated via USB mass storage standard. Statistics will be downloaded to PC via USB mass storage standard. Supports requirement 13, 14. Rationale: this way we can achieve even better interoperability than stated in requirements and also it is more standardized solution which should be easier to implement. Also no drivers are needed in modern operating systems so our system will not need any additional installations.
- Direction of the car will be remembered and used in path finding algorithm. Supports 15. Rationale: it will allow us to keep the car representation in right direction and the algorithm will not force the driver to turn when he deflects from the route.
- We are using incremental path finding algorithm – the alternative route is picked up from other not so well rated routes. Supports requirement 16. Rationale: alternative route was already computed when the optimal route had been searched and it will always give us different route.
- Voice module will run as an independent thread, which is asking for route changes every second. Supports requirement 19. Rationale: The system will not be overwhelmed with questions from this module and still it will be fast enough for a driver.
- Internal implementation of GUI will have customizable colour schemes. Supports requirement 22. Rationale: this way we can achieve variable user interface and in future we can make it more attractive. Also it allows tuning of day/night modes.
- The actual route will be saved in persistent memory. Supports requirement 25.
- There will be screen which will offer the user possibility to select the type of input for destination. Types will be (address, coordinates, selection from previously saved position, touch on the map). Supports requirement 26. Rationale: it is better to offer the user a crossing in the interface then overwhelming the interface with too many input graphics components.
- There will be 20MB of internal persistent memory reserved for statistics, last planned route, configuration and saved positions. Supports requirements 23, 24, 25, 27, 28, 29. Rationale: “640kB ought to be enough for everybody”, but we believe that for persistent use nowadays 20MB should be enough.
- Statistics will be stored in the form of .csv tables. Supports requirement 30. Rationale: simple file format, which is supported in many applications (MS Excel, OpenOffice, etc.).
- There will be 2GB of place in the device for the maps and POI data files. So there will not be problems to load accurate maps of whole Europe. Supports requirement 33. Rationale: this amount of memory is enough for holding all the relevant data in appropriate radius.

UML Basic Diagrams

Context diagram



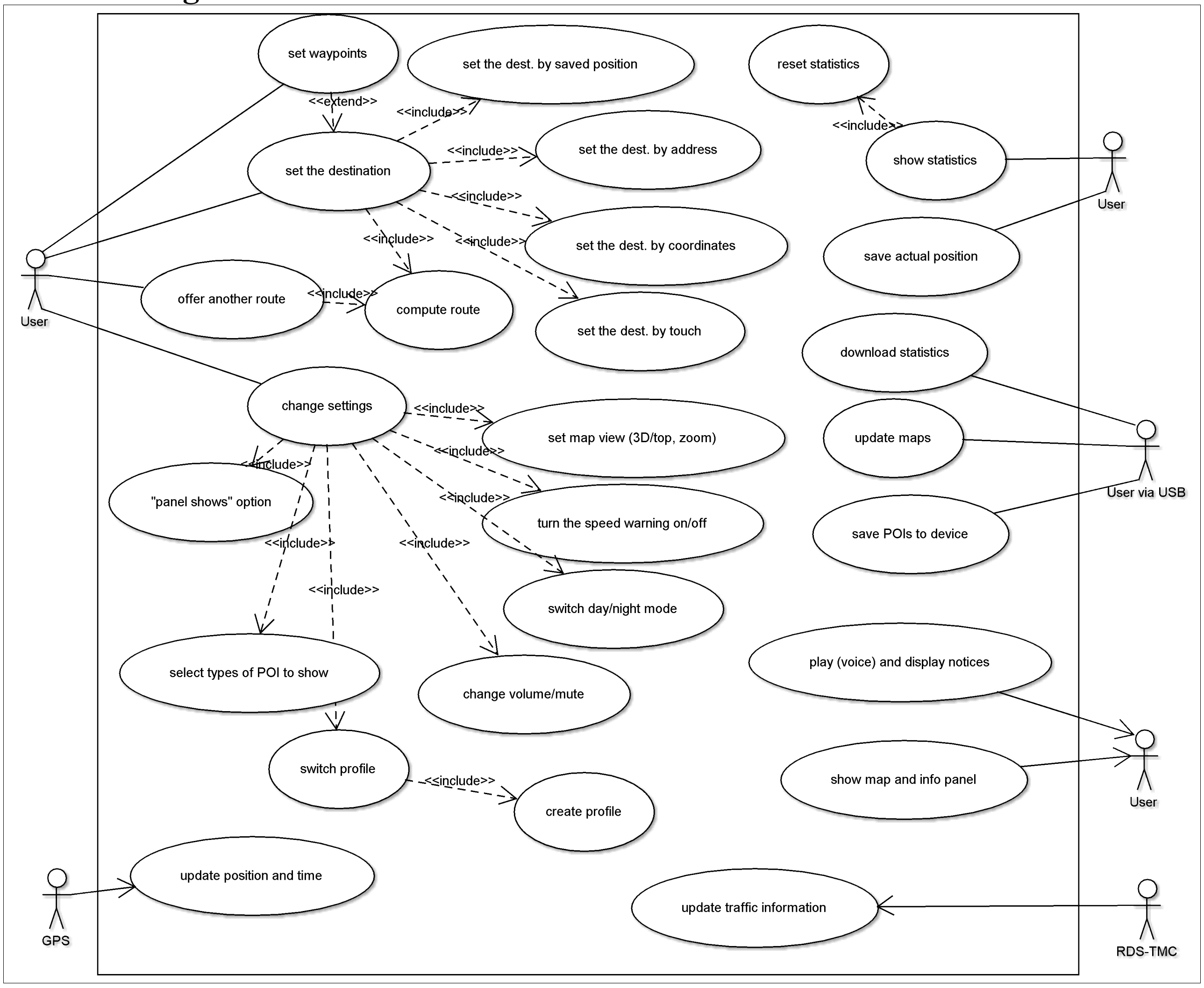
User – uses the device in a car.

User via USB – uses the device while uploading new maps, POIs and downloading statistics.

GPS – input from satellite system

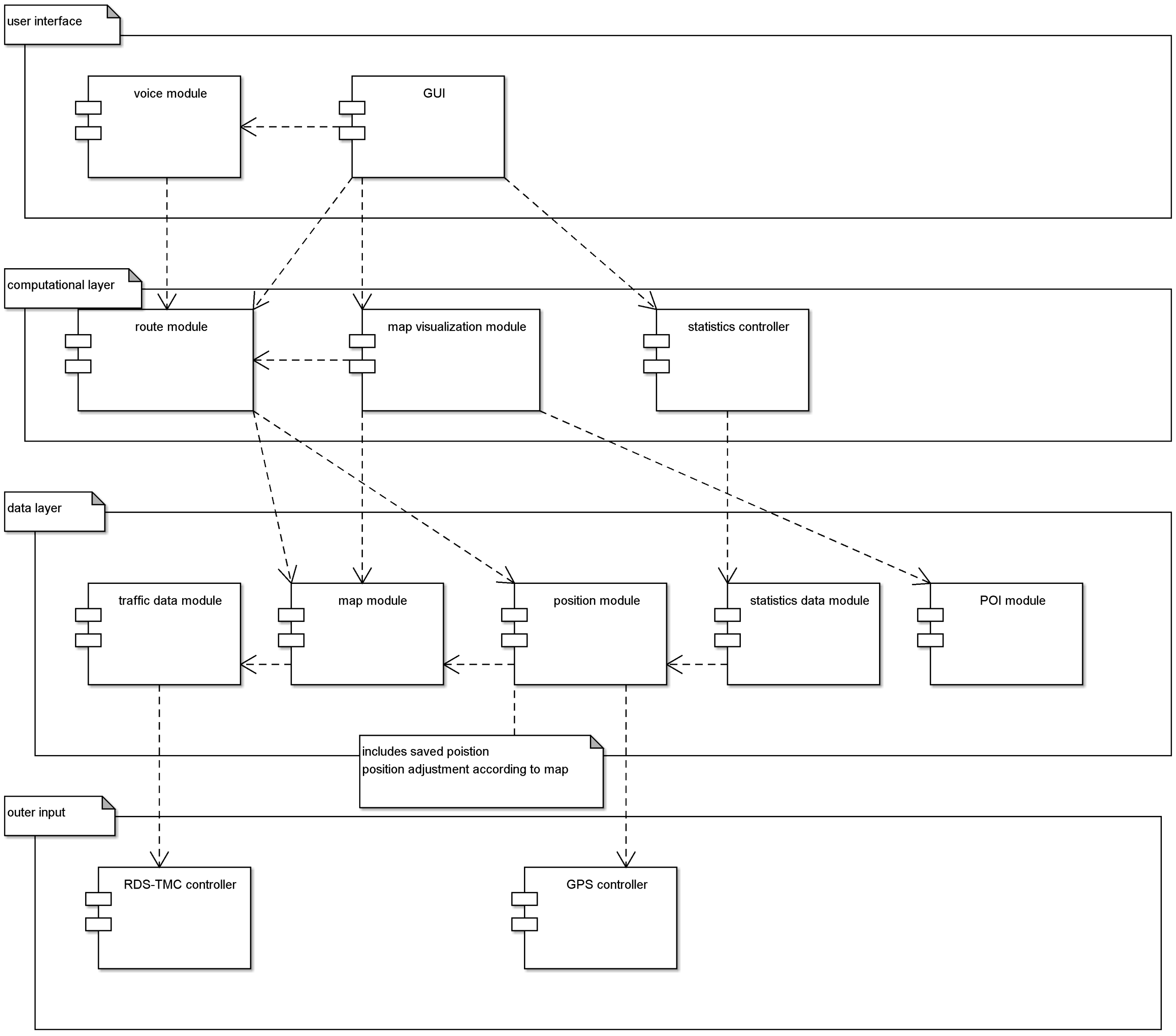
RDS-TMC – input from radio transmitters

Use case diagram



This diagram describes all the possible uses of the system. Use cases “Play voice and display notices” and “Show maps and info panel” are here to warn of the need to process these use cases even without the user's input. Some of the use cases are described via their sequence diagrams with commentary.

Component diagram



We have decided to use 4 layers. A layer provides services to the layer above it and receives services from the layer below it. Components inside one layer can also communicate between themselves. This way we can achieve good separation of responsibilities and the components are not dependent on many others.

Layers of architecture

Outer input layer

Layer which provides input from outer systems (GPS, RDS-TMC). This way the components will be independent from the particular system and usable in other possible projects.

Data layer

Provides all data needed for system activities. This way we can separate application logic from data handling and the handling is centralized in one layer providing better control over the data. All the components provide interface for their data.

Computational layer
Includes higher level functions. Usually demanding computational algorithms which use data layer. The components in this layer can be replaced for other or better functionality. For example route module can be replaced with updated route module providing better path finding algorithm. The layer separates application's logic. Components provide high level access to data stored in lower layer.

User interface layer
This layer provides user interface – including voice output and graphical input + user input. This way we can easily change the user interface. For example localization or change of voice behaviour.

Module responsibilities

GPS controller:

- provides interface for GPS HW
- gives position, direction, velocity and time according to received GPS data

RDS-TMC controller

- provides interface for RDS-TMC HW
- gives actual traffic data it receives

Map module

- provides interface to saved maps with supplemented
- provides map uploads/actualizations via USB
- handles map data
- provides interface to save the actual route into persistent memory and to load it

Position module

- saves desired positions (saved ones, destination position, way points)
- adjusts position according to the map (puts the car on the road) – error correction
- gives actual position (it remembers it)
- handles appropriate data

Statistics data module

- handles all statistics data including user profiles

- updates statistics according to changes in position
- provides interface for statistics data and user profiles

Traffic data module

- handles traffic data received via RDS-TMC
- keeps inner database of traffic data
- provides interface for traffic data

POI (points of interest) module

- handles all saved POIs
- provides POIs uploads/actualizations via USB
- provides interface for POIs data
- keeps list of desired types of POIs to show

Route module

- computes route according to map, traffic, way points, actual position and destination
- keeps track of upcoming changes in direction and provides interface for voice and GUI module to ask for them
- watches over the speed warnings

Map visualization module

- makes visualization of the vector map
- puts desired POIs on the map
- puts traffic information on the map
- puts route on the map
- handles types of view and view port (zoom, 3D/top view)
- translates screen position to position on the map (for user input)

Statistics controller

- switches statistics profiles
- provides reset of the statistics
- provides higher level interface for statistics data
- provides export of the statistics via USB
- is responsible for updating statistics data

Voice module

- provides audio output
- gives orders to users according to route

- offers volume change/mute
- plays click and beep sounds

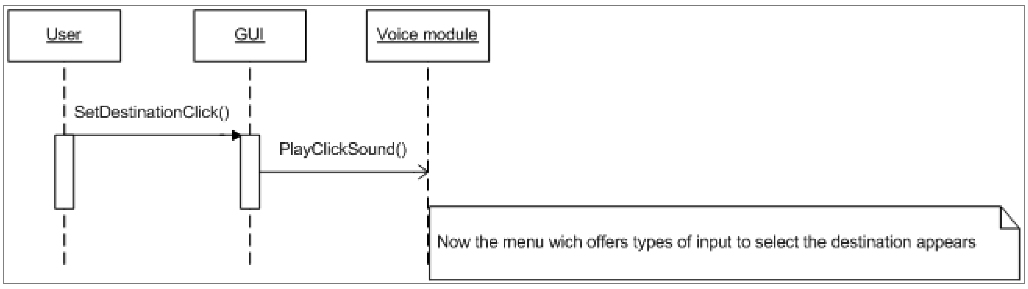
GUI

- provides user interface
- handles day/night mode
- propagates events to responsible modules
- remembers which fields to show on the information panel

UML Sequence Diagrams

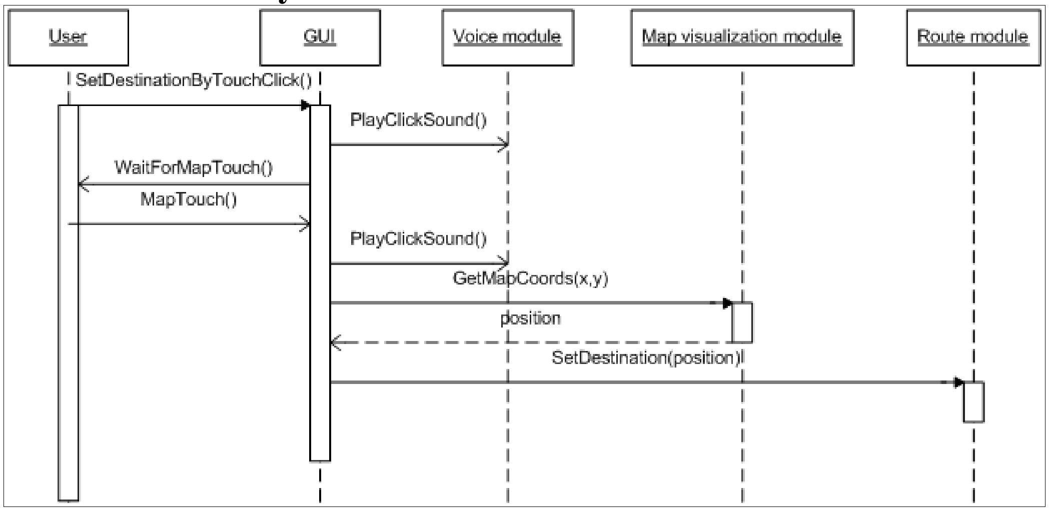
Only the representative ones are presented.

Set Destination


User
GUI
Voice module
SetDestinationClick()
PlayClickSound()
Now the menu wich offers types of input to select the destination appears

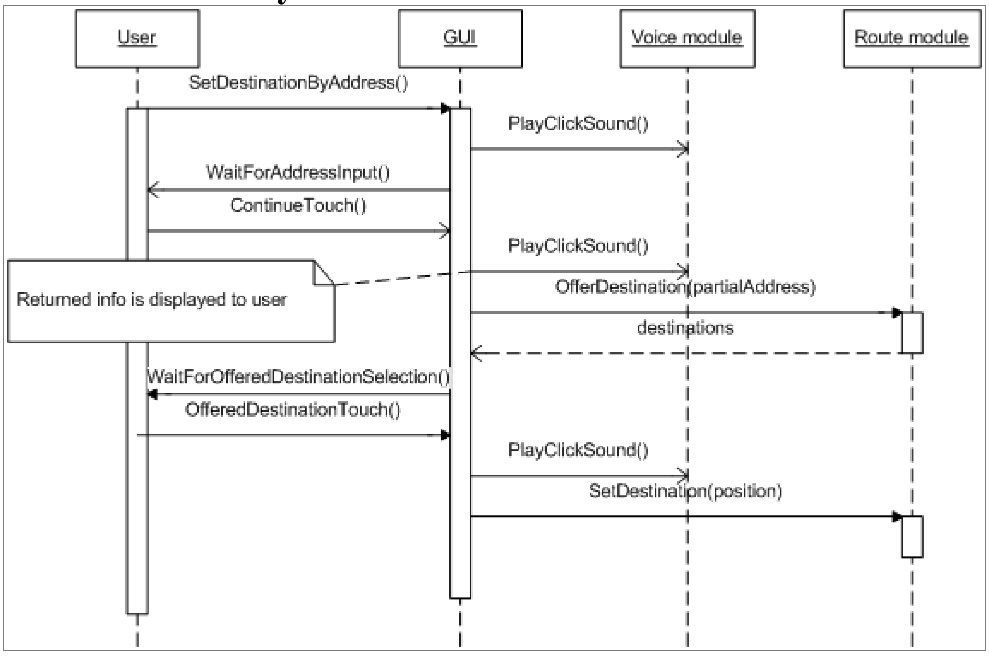
This use case is executed, when the user clicks on the icon Set destination to set a new destination in map view. It just shows the menu from which the user can select types of input (i.e. by address, coordinates, etc.).

Set Destination By Touch



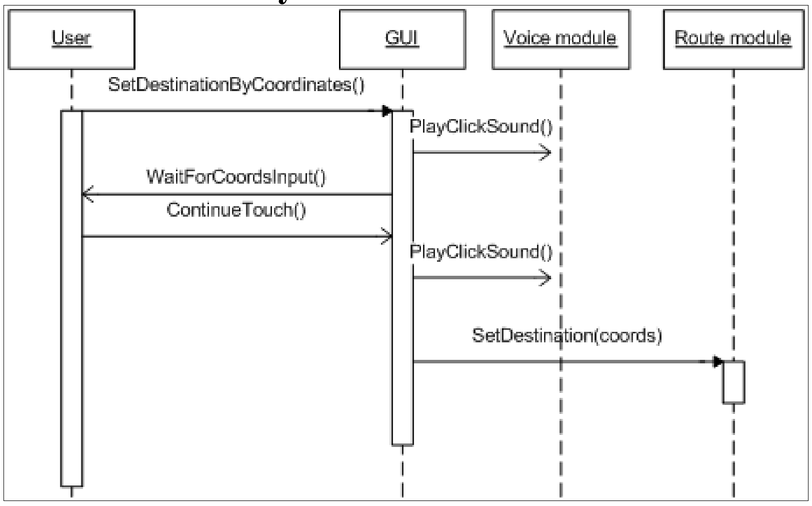
This use case begins when the user clicks in the input type menu, that he wants to enter the destination by touching the screen (pointing at the place on the map). After every call of SetDestination a new route is computed (executed a Compute route use case). It finishes when the destination was passed to the route module and the screen is back in map view.

Set Destination By Address


User
GUI
Voice module
Route module
SetDestinationByAddress()
PlayClickSound()
WaitForAddressInput()
ContinueTouch()
PlayClickSound()
Returned info is displayed to user
OfferDestination(partialAddress)
destinations
WaitForOfferedDestinationSelection()
OfferedDestinationTouch()
PlayClickSound()
SetDestination(position)

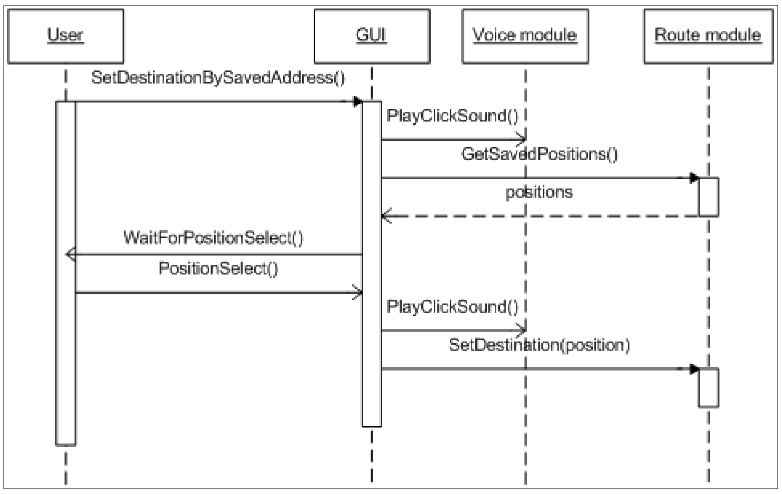
This use case begins when the user clicks in the input type menu, that he wants to enter the destination by entering the address. It includes the offering with selection of found addresses. It finishes when the destination was passed to the route module and the screen is back in map view.

Set Destination By Coordinates



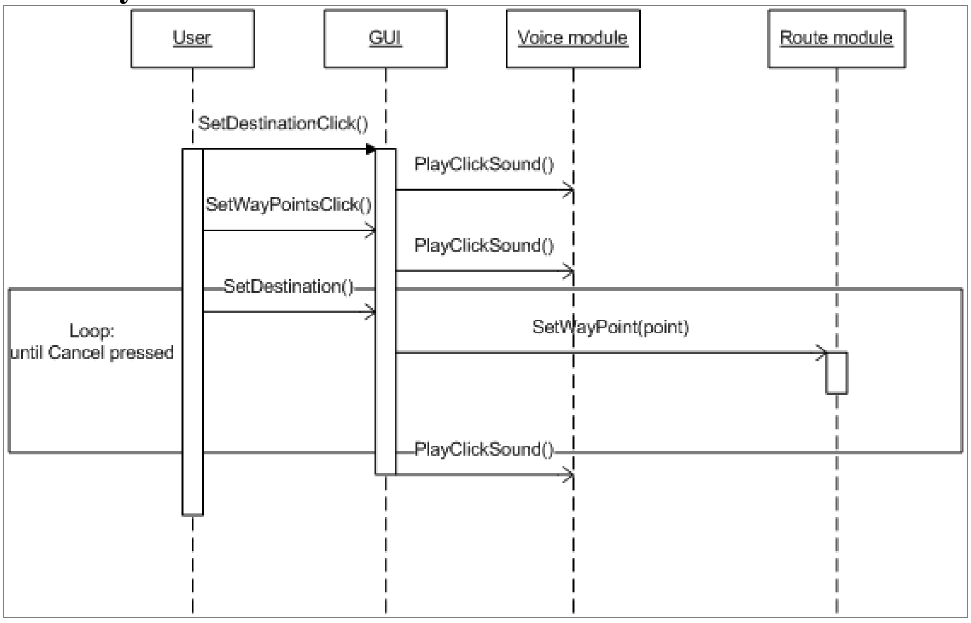
This use case begins when the user clicks in the input type menu, that he wants to enter the destination by entering the coordinates. It finishes when the destination was passed to the route module and the screen is back in map view.

Set Destination By Saved Position



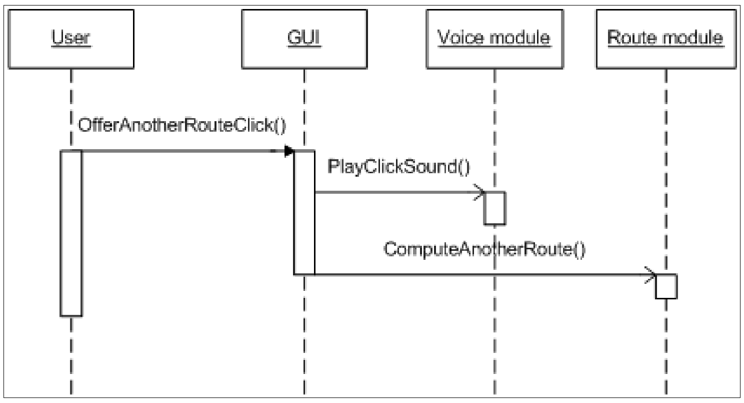
This use case begins when the user clicks in the input type menu, that he wants to enter the destination by selecting from previously saved positions. It includes the offering of list with saved positions. It finishes when the destination was passed to the route module and the screen is back in map view.

Set Way Points



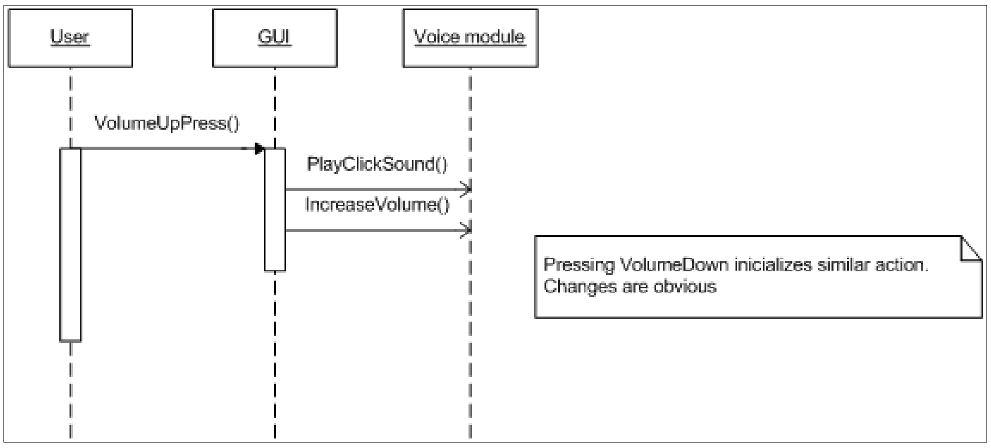
This use case begins when the user clicks in the input type menu, that he wants to enter the destinations of way points. Then as long as he does not press the Cancel button, the Set Destination use case is executed in the loop, but this time the way points are saved into route module instead of calling set destination. It finishes when the Cancel button is pressed. (Cancel button is shown in all the views as mentioned in design decisions) and the screen is in map view.

Offer Another Route



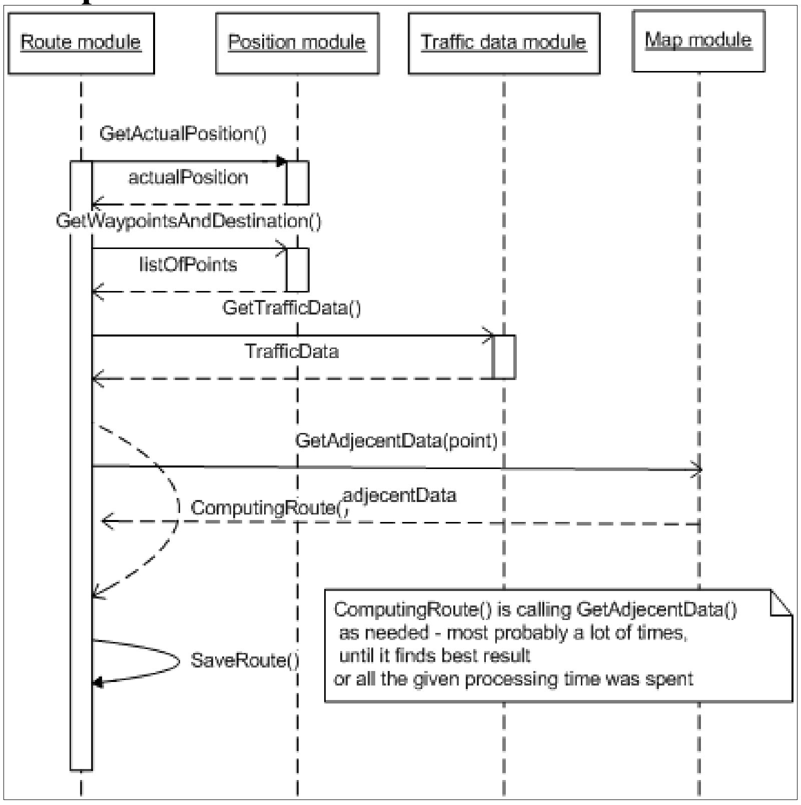
This use case begins when the user clicks on the Offer another route button, which will be in the menu (which is accessible by clicking Menu icon from the map view). It finishes when the request to compute another route was passed to the route module and the screen is back in map view.

Change Volume


User
GUI
Voice module
VolumeUpPress()
PlayClickSound()
IncreaseVolume()
Pressing VolumeDown inicializes similar action. Changes are obvious

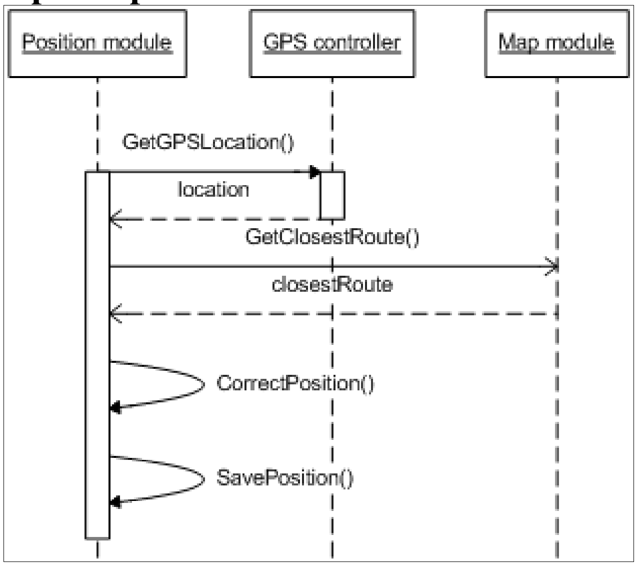
This use case begins when the user touches volume up or alternatively volume down button. It only passes the message to increase/decrease the volume to voice module.

Compute route



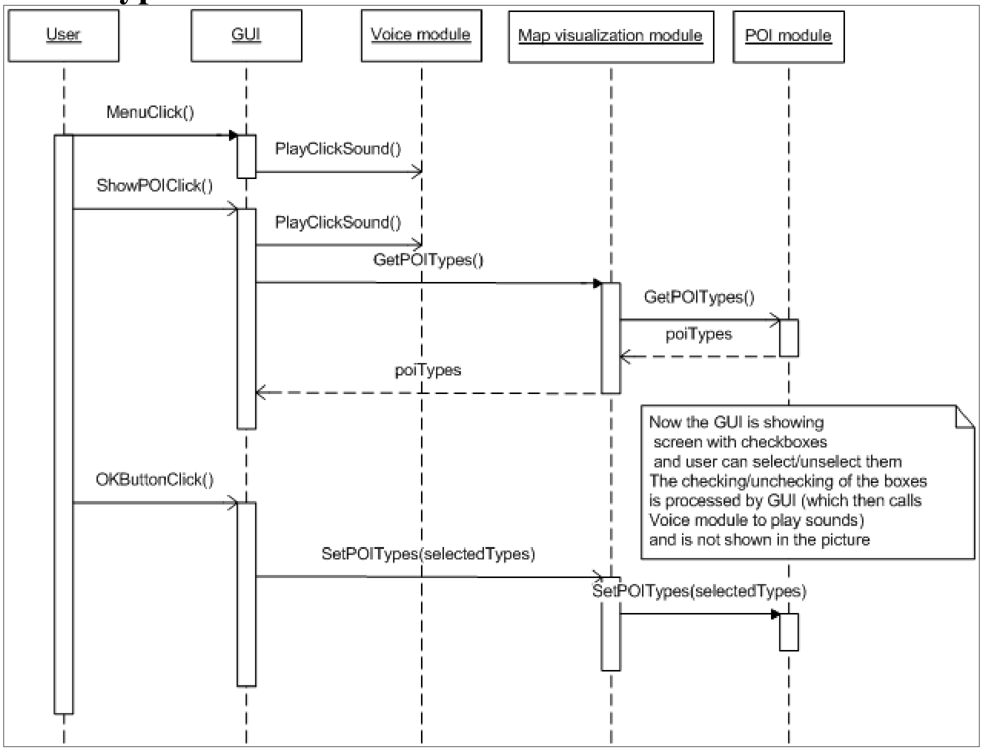
This use case begins when the SetDestination is invoked. It finishes when the new route is computed and saved.

Update position and time



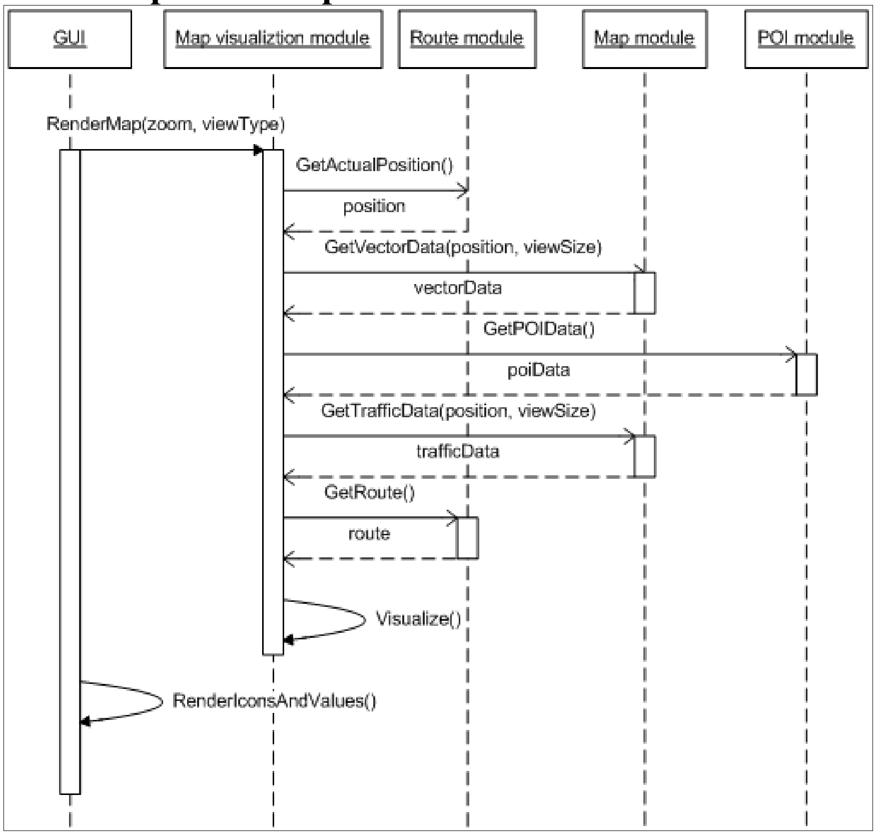
This use case is executed by periodical thread which updates position and time when the SetDestination is invoked. It finishes when the new corrected position is saved.

Select types of POI to show


User
GUI
Voice module
Map visualization module
POI module
MenuClick()
PlayClickSound()
ShowPOIClick()
PlayClickSound()
GetPOITypes()
GetPOITypes()
poiTypes
poiTypes
Now the GUI is showing screen with checkboxes and user can select/unselect them The checking/unchecking of the boxes is processed by GUI (which then calls Voice module to play sounds) and is not shown in the picture
OKButtonClick()
SetPOITypes(selectedTypes)
SetPOITypes(selectedTypes)

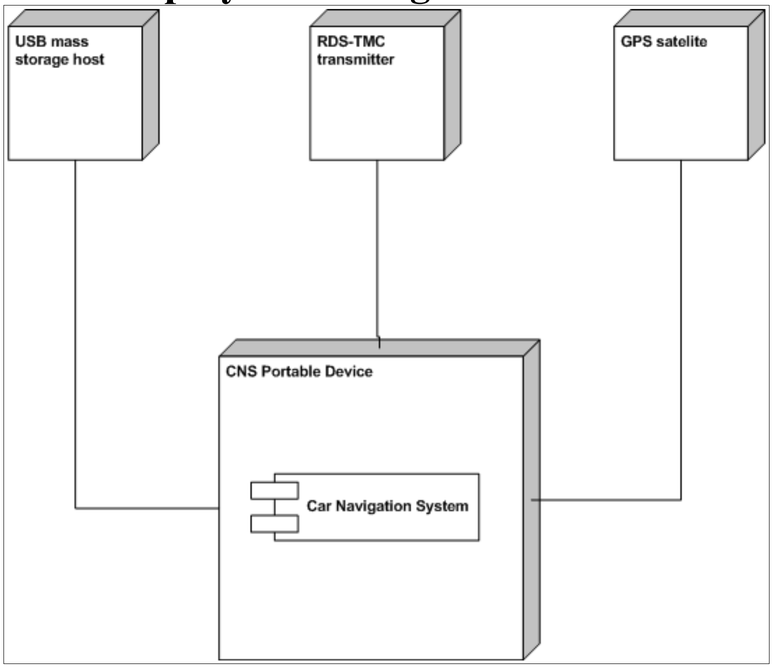
This use case is executed when user wants to select which types of POI to show. It finishes when user click OK and is back in map view.

Show map and info panel



This use case is executed by periodically by a thread when the user is in map view. It updates the screen. It finishes when the screen is refreshed, but then is executed almost immediately again.

UML Deployment Diagram



This diagram describes which other devices are used with the device. Only the CNS portable device is under control of developers and architects.

Performance analysis

Frequencies of events and accuracy

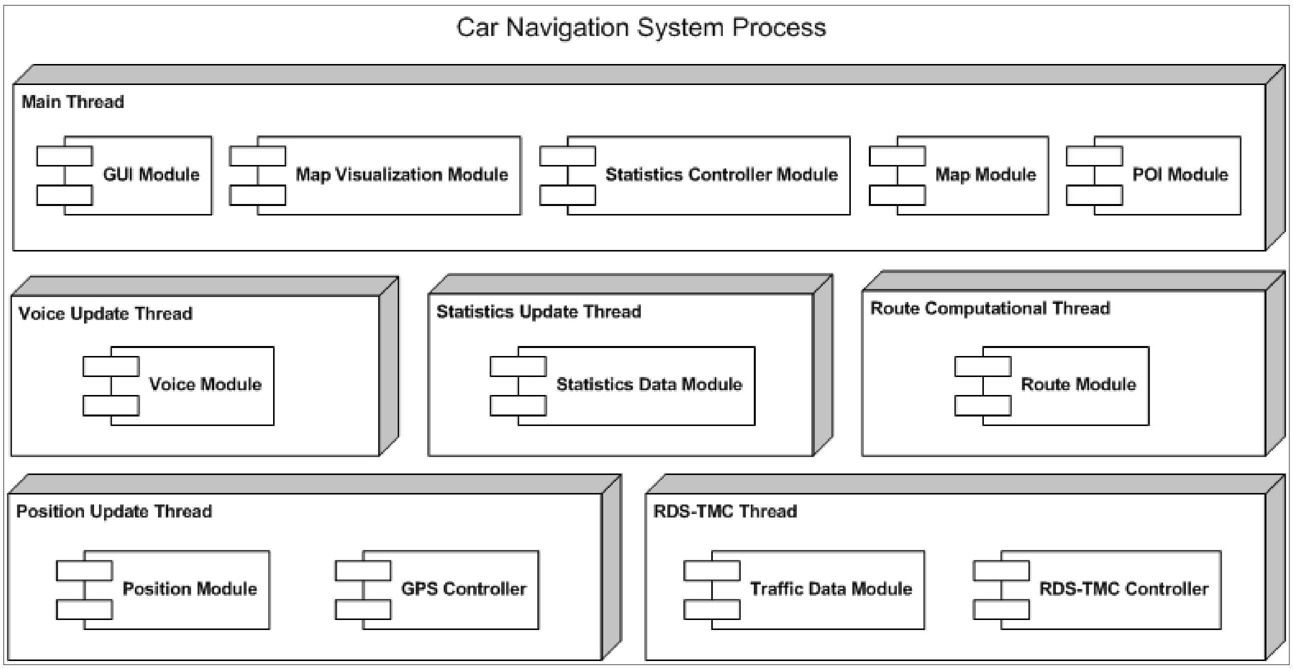
GPS's maximal accuracy for civilian use is about 3 m and minimal accuracy is about 15 m when at least 3 satellites are in the view.
The GPS HW can give a position (and velocity, direction, time data) every second (for example the SiRFstar III GPS chip).
Position controller will update the actual position in frequency of 28 Hz (provided that the maximal speed 300 kmph of the device and maximal GPS accuracy is 3 meters, it will take 10e-5 hours to drive 3 m at this speed and that is 0.036 sec, which gives us frequency of approx. 28 Hz). But it can ask for the data the GPS HW only once per second, so the position between the seconds is calculated by interpolating last known values.
If the RDS-TMC message is composed only from compulsory data, it takes 68 seconds to receive all the information from the message sent by the RDS-TMC transmitter (transmitting speed is 1187.5 bits per second, maximal number of events in the message is 2048 and one event is composed of minimally 39 bits). But the RDS-TMC controller will check the reception of new events every second so the database of traffic problems is kept actual.
Screen will be refreshed at 20 Hz and user input will be noticed at the same rate.
Computing route can take 3 seconds while the screen is refreshed at 15 Hz.
Statistics will be updated every second.
Voice module is retrieving information about direction changes every second.

Estimated resource consumption			
Name of the event	**CPU Usage in percent**	**Operating memory usage in percent**	**Frequency of occurrences**
Acquiring position from GPS HW	2%	1%	1 Hz
Update of the actual position (including interpolation and error correction)	4%	1%	28 Hz
Acquiring new data from RDS-TMC	2%	2%	1 Hz
Screen refresh and user input checking	35%	25%	20 Hz

Computing road while keeping the screen refreshing in 15 Hz	78%	80%	Dependant on user's needs, but at most once per 3 seconds
Statistics update	2%	3%	1 Hz
Voice module	10%	15%	1Hz

Performance tactics:
We are introducing concurrency so the most resources intensive tasks will have independent threads with appropriate priorities (this tactics also gives us graceful degradation property – for example if one of the not so important thread encounters some problem, remaining sub-system will continue working properly). There will be main thread which handles screen updates, user input and some components which do not need to run independently. Other threads will be: statistics update, position update, voice update, RDS-TMC update, route computational thread. The threads are given processor time according to their priority. We are using only frequencies of updates which are as low as possible for usability. We also bound the execution time for finding the best route to 3 sec. Also the frequency of the user's input is bounded to 20 Hz – as the GUI is running at that speed.

Thread diagram



This diagram describes in which thread which component is running.

3 most critical performance scenarios:

1. User wants to have the actual position and map with all the requested information available

while navigating and be able to control the device with low response time. We want to provide the screen refresh rate at 20 Hz and the same thread handles the input in that rate. Our load estimates show, that this should be possible even when all the other possible tasks are running in the same time.
2. User wants to re/compute the route in short time while keeping the screen updated. We will provide the route at 3 sec and will be refreshing the screen at 15 Hz while computing the route. We estimate that this is possible, but the CPU will be in the highest possible load.
3. User want to be informed by voice about the upcoming changes (of direction etc.) as soon as possible. Our voice module will check the change status every second, which should be enough in even the route densest areas. The requested computing load is low enough to be able to keep this frequency of checking the conditions.

Top 5 technical risks:

1. Bad selection of route planning algorithm: Overall usability of the device is influenced a lot by it's ability to give a driver really the shortest route that meets the given constraints. If wrong algorithm (slow, not holding given constraints or unsuitable in some other way) is chosen, whole device may become completely unusable although everything else is working according to the specification.
2. Wrong handled concurrency of threads: Working with threads always brings new risks to the system and handling them in the wrong way may introduce various performance problems that may cause violation of some of the requirements or in worse case, wrong handling of the concurrent access to the data may lead to the severe data corruption.
3. Bad HW selection: Big care needs to be taken also while selecting the proper hardware. There has to be proper balance between the speed and the power consumption of the HW. Using too fast HW may lead to big energy consumption and to violating some availability requirements. On the other hand HW with the low power consumption might not be strong enough and some performance requirements may be violated
4. Slow implementation of map rendering: Wrong map rendering mechanisms may lead to big slow-down of the main thread and to violating of some requirements. For example GUI response time may be higher than required etc.
5. Bad underlying OS selection: Choice of the right underlying OS affects not only the cost of the system but also influences simplicity of implementing desired functionality (programming and debugging). OS also has to have a good and if possible highly customisable or controllable thread scheduling algorithms.

Evaluation overview tables.

VALUE: How much value does this use case contribute to the user of the system?

name of reviewer:	Radim Cebis and Pavel Cernohorsky
name of application	Car Navigation System

	criticality to system operation 0=not valueable; 7=extremely valuable								brief motivation
Name of Use Case	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
Set destination by address								X	This is the function is important for all the users.
Set destination by coordinates			X						This is useful just for small group of users.
Compute route								X	This is the function which is why the system could be sold.
Update position and time								X	Necessary for navigating.
Show map and info panel								X	Necessary for navigating.
Save position					X				Increases usability of the system, but it is not so important for its main function
Update maps						X			Increases modifiability of the system, but when good maps are provided out of the box, it is not so needed, but the routes change in time.

CRITICALITY : how critical is this use case to the overall operation of the system?

name of reviewer:	Radim Cebis and Pavel Cernohorsky
name of application	Car Navigation System

	criticality to system operation 0=not important, 7=essential								brief motivation
Name of Use Case	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
Set destination by address						X			It is important to be able to give input.
Set destination by coordinates						X			It is important to be able to give input.
Compute route								X	System would be useless without this function
Update position and time								X	Necessary for the previous function to work
Show map and info panel								X	It is necessary to give the user the information
Save position	X								This is not needed for navigating.
Update maps	X								This is not needed for navigating.

EFFORT: how much effort will be needed for implementing this use case?

name of reviewer:	Radim Cebis and Pavel Cernohorsky
name of application	Car Navigation System

	implementation effort required: 0=very little, 7=very much								brief motivation
Name of Use Case	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
Set destination by address				X					This only needs some structures for address lookup. It is just an input.
Set destination by coordinates		X							This just needs a small part of GUI. Points will be internally saved in coordinates, so nothing else special is needed.
Compute route								X	It will be hard to choose and develop fast algorithm with best results which must take into account traffic information.
Update position and time					X				This requires the developer to learn GPS HW interface and to implement some error correction.
Show map and info panel								X	Good implementation is essential for good performance.
Save position		X							Easy to do – just save the data to the persistent memory.
Update maps		X							Easy to do – just load new data.

VALUE: How much value does this component contribute to the user of the system?

name of reviewer:	Radim Cebis and Pavel Cernohorsky	phase 1
name of application	Car Navigation System	

	criticality to system operation 0=not valueable; 7=extremely valuable								brief motivation
Name of Component	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
GPS controller								X	This component is necessary for system operations.
Position module								X	This component is necessary for system operations.
Route module								X	This component is necessary for system operations.
Map visualization module						X			Necessary module, but can be sufficient even when is not perfect
GUI						X			Necessary module, but can be sufficient even when is not perfect
Map module								X	This component is necessary for system operations.
Voice module						X			Valuable for user, but can live with only graphic representation.

CRITICALITY: how critical is this component to the overall operation of the system?

name of reviewer: Radim Cebis and Pavel Cernohorsky phase 1
name of application Car Navigation System

	criticality to system operation 0=not important, 7=essential								brief motivation
Name of Component	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	
GPS controller								X	This component is necessary for system operations.
Position module								X	This component is necessary for system operations.
Route module								X	This component is necessary for system operations.
Map visualization module								X	This component is necessary for system operations.
GUI							X		This component is necessary for system operations, but might be slower or not so pretty looking
Map module								X	This component is necessary for system operations.
Voice module	X								This function might be turned of by the user – he can live without it .

EFFORT: how much effort will be needed for implementing this component?

name of reviewer: Radim Cebis and Pavel Cernohorsky phase 1
name of application Car Navigation System

	implementation effort required: 0=very little, 7=very much								brief motivation
Name of Component	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	
GPS controller			X						Just reads values provided by GPS HW as specified by the HW.
Position module				X					Needs to perform interpolation of position and error corrections.
Route module								X	It will be hard to choose and develop fast algorithm with best results which must take into account traffic information.
Map visualization module						X			Depends on graphic interface and hardware. Needs fast implementation.
GUI			X						Depends on the OS API, but should not be so hard to implement alone.
Map module					X				Needs good designed data structures with fast spatial search.
Voice module				X					Depends on underlying sound HW and chosen libraries. But should not be so hard.