



## Panic Button

## Objectives

- To analyse and model a problem in an object-oriented way, up to the point where it would be possible to begin to design a solution.
- To consolidate the analysis method with the tools and techniques used within the framework provided by the Unified Modelling Language (UML).



This case study is structured to present you, the participant, with a series of exercise, together with some information if relevant. The session leader will play two roles: the **Customer**, who you can ask for any background domain knowledge, and the **UML Consultant**, who doesn't know anything about the domain, but is able to assist in representing information in a UML-compliant way. Make sure you address your questions to the right role!

Each exercise is followed by a *suggested* solution. Note the indefinite article and the italics – it will be amazing if your solution is the same as the one provided, or that of any neighbouring syndicate group. That doesn't mean a solution is wrong – a model can only represent knowledge, and you can only represent the knowledge that you have. A model is correct if it represents your knowledge – if that knowledge is incorrect or incomplete, then that's a different problem!

Two teams with exactly the same knowledge will normally arrive at broadly similar models, and most of the differences will be in the names given to particular modelling elements. Of course, it is also possible to arrive at two or more models that are correct, but totally different! The job of an analyst is to encourage different views so that a qualitative assessment of each can be made. They can then be compared, contrasted and discussed, and the best one chosen. Or a new model can be constructed from the best points of all of them.

You will be given the opportunity to discuss differing solutions after each exercise. You can then choose whether to advance your own solution to the next stage, or to use the suggested solution as a checkpoint, or to employ the most realistic approach: that of stealing the best bits from all the solutions you've seen and incorporating them into your own!

Enjoy the exercise.

## Exercise 1 – The Problem Statement

- ▶ Read the Problem Statement on the following page.
- ▶ What is the problem?
- ▶ What is your critical assessment of this as a 'problem statement'?
- ▶ What guidelines could be given to anyone writing a Problem Statement?

10 mins



*Exercise 1 – Read this...*



**Panic Button – Problem Statement**

A device to assist the problem-stricken traveller.

If a car or one of its occupants suffers an incapacitating problem such as engine failure, accident or medical problem, it can sometimes be difficult or unsafe, if not impossible, to summon help from the appropriate emergency service. **Panic Button** is a proposed solution to this problem.

Panic Button will be a subscription-based service offered to motorists through one of the established motorists' organisations, such as the AA or RAC, who may possibly market the service under a differentiating brand name. Licensed operators, such as Halfords, will undertake installation of the necessary hardware. The franchising options will be explored in the second half of the feasibility study currently being produced.

The hardware will include a dashboard console, a central control module (CCM), a Global Positioning Satellite (GPS) receiver, and a communications unit (CommU) capable of, in the first instance, accessing a cellular telephone network. In the event of hardware failure due to vehicle damage, fallback will be to an Emergency Position Indicating Radio Beacon (EPIRB) operating on the now-superseded 121.5MHz emergency frequency.

The dashboard console will feature a numeric keypad, the red 'panic button', a cancel/reset button and a LCD scrollable display capable of displaying five lines of 20 characters. All will be backlit when the car's lighting is used or when any button is pressed. The LCD will display the current date and time when the console is not in use. A 'beeper' will provide audible feedback and prompts. The console is connected to the CCM. The CCM will, in turn, be connected to the GPS, CommU, the vehicle's own Vehicle Management System (VMS), and the EPIRB.

The following examples indicate how the system could be used.

1. Mr Driver is on the M4 when he feels pains in his chest. His medical history leads him to believe assistance is required. He pulls over onto the hard shoulder and keys a code into the console. The console displays the code and the text 'Medical emergency – confirm'. Mr Driver presses the panic button to confirm. The CCM dials the pre-programmed medical emergency number and, on contact, transmits a message containing the emergency type, current position, and driver and vehicle details. An acknowledgement is received and the text 'Help is coming – ETA <time>. Current time <time>' is displayed.
2. Mrs Driver is waiting at traffic lights when a car fails to stop behind her. The collision shunts Mrs Driver into the car in front of hers, triggering the airbag. The VMS registers the deployment of the airbag, and informs the CCM. The CCM displays a message on the console 'Airbag deployed – call help?'. If no button is pressed within 30 seconds, or if confirmed within that time by pressing the red panic button, a message is formatted and transmitted to the appropriate emergency services. Acknowledgement is displayed as above.
3. The panic button is pressed in an area with poor cellular signal strength. With no code, the message 'Confirm call help' is answered by pressing the button again. On receiving 'No signal' from the CommU, the EPIRB will transmit the vehicle callsign and position. No acknowledgement is possible in this case.

Subscribers will be able to sign up for Gold, Silver or Bronze Star membership, which will allow them to vary the range of services available to their system. The Gold Star scheme will include an anti-theft option. Once theft of the car has been discovered, a phone call will activate the vehicle's CCM and the car's position will be transmitted, allowing for rapid recovery of the vehicle and perhaps the capture of the thieves. A 'Gold Star' sticker displayed in the car's window will therefore serve as a deterrent. Another 'Gold' service will enable subscribers who have locked their keys in the car to call the Panic Button service provider who will, on the satisfactory completion of a security check, send a signal to the car to unlock the doors.

In the future, Low Earth Orbit (LEO) satellites will provide an alternative or augmentative means of communication, allowing for the Panic Button network to cover the globe. This will make it practicable and attractive to owners of other forms of transport such as boats and planes, for whom land-based coverage is less useful. Smaller, portable versions will also be offered to those involved in outdoor pursuits.

## Exercise 2 – Business Context Diagram

- ▶ Draw a diagram showing the Panic Button business and the entities (actors) with which it has *direct* interaction.
- ▶ Identify each functional component of the business and the interfaces between them
- ▶ Begin your glossary by writing a brief description of each domain-specific term as it is discovered.

15 mins



## Exercise 3 – Activity Diagrams

- ▶ Draw an activity diagram showing the end-to-end business process of handling an emergency call.
- ▶ Use swimlanes to allocate different parts of the process to the appropriate entities.

15 mins



## Exercise 4 - Actors

- ▶ Assume the scope will be restricted to the *on-board* system – that is, the system as installed *in the car*.
- ▶ Draw a diagram showing the on-board system and identify the entities (actors) with which it has *direct* interaction.
- ▶ Add the actor definitions to your glossary.

10 mins



## Exercise 5 – Use Cases

- ▶ Draw a diagram showing at least five use cases offered by the on-board system and their actors.
- ▶ Identify each actor as *primary* or *secondary* with respect to the use case.
- ▶ Write a full description of one use case, including pre- and postconditions and at least one alternate flow.

15 mins



## Exercise 6 – Advanced Use Cases

- Organise your Use Case diagram into Packages by primary actor.
- Identify opportunities for using «*includes*», «*extends*» and *generalisation* dependencies within each package.

20 mins



## Exercise 7 – ‘Candidate’ Classes

- ▶ Use the information from previous exercises to construct a list of key domain concepts that might make useful classes.
- ▶ For each candidate, try to assign a stereotype.
- ▶ Include each candidate in your glossary.

20 mins



## Exercise 8 - Scenarios

- Using nouns drawn from your list of candidate classes, construct a scenario describing the 'Mr Driver has a heart attack' example given in the problem statement.
- Remember – One '*Subject-Verb-Object*' clause per line.
- If time – do the same for the other examples.

20 mins



Excerpt from Problem Statement:

Mr Driver is on the M4 when he feels pains in his chest. His medical history leads him to believe assistance is required. He pulls over onto the hard shoulder and keys a code into the console. The console displays the code and the text 'Medical emergency – confirm'. Mr Driver presses the panic button to confirm. The CCM dials the pre-programmed medical emergency number and, on contact, transmits a message containing the emergency type, current position, and driver and vehicle details. An acknowledgement is received and the text 'Help is coming – ETA <time>. Current time <time>' is displayed.

## Exercise 9 – Sequence Diagrams

- Convert your scenario from the previous exercise to a sequence diagram.
- Use notation to distinguish between *message* events and *response* events.
- If time – experiment with other solutions, using additional objects and/or other distributions of behaviours.

15 mins



## Exercise 10 – The Class Diagram

- Use your sequence diagram(s) to construct a first-cut class diagram.
- Show only that which can be supported by the information from previous diagrams – likely to be classes, unadorned associations and operations.

15 mins



## Exercise 11 – Refining Associations

➤ Add detail to your class diagram from the previous exercise:

- Association names
- Role names
- Multiplicity
- Aggregations
- Anything else – ask your domain expert for more information!

20 mins



## Some Suggested Solutions

- These 'solutions' are NOT definitive!
- The best solutions probably lie somewhere between these and those produced by you.
- Use these to promote discussion.



***Problem Statement - Discussion***

Mmmm. The question is – is this really a statement describing a problem? As it stands, there seems to be quite a bit of text describing a solution – ‘A subscription-based service offered to motorists through one of the established motorists’ organisations’? There’s a fair bit of detail on the implementation technology, too. The console is described right down to the type of display, and the EPIRB’s broadcasting frequency is not only out of place here, it’s also wrong! (It’s 121.5KHz, since you asked...)

There are several ways to look at this example, all of them debatable. If all that is wanted is a bald statement of what the problem is, then the first paragraph might well be enough:

‘If a car or one of its occupants suffers an incapacitating problem such as engine failure, accident or medical problem, it can sometimes be difficult or unsafe, if not impossible, to summon help from the appropriate emergency service. Panic Button is a proposed solution to this problem...’

There is also some useful additional information regarding the problem of tracing the vehicle if stolen, or dealing with being locked out of the car, which could be included.

Including implementation details such as the use of GPS or cellular technology may serve more to describe the solution than the problem, but can be justified if the problem is re-framed thus:

‘The problem is to use a combination of existing, established technology such as GPS, cellular telephony and a vehicle’s on-board diagnostic facility to facilitate the reporting of problems of a medical or mechanical nature...’

... in which case, the three examples of usage and the information on future directions becomes useful.

There may be other constraints on writing a problem statement. It might be not be politically expedient to describe the *real* problem:

‘The problem is to conduct routine and covert surveillance on individuals on a regular or ad-hoc basis, while deferring the operating costs to these individuals... it may be necessary to trace a vehicle’s movements, immobilise it, and even gain entry to the vehicle...’

Of course, this could never happen in a democracy...

**Summary:**

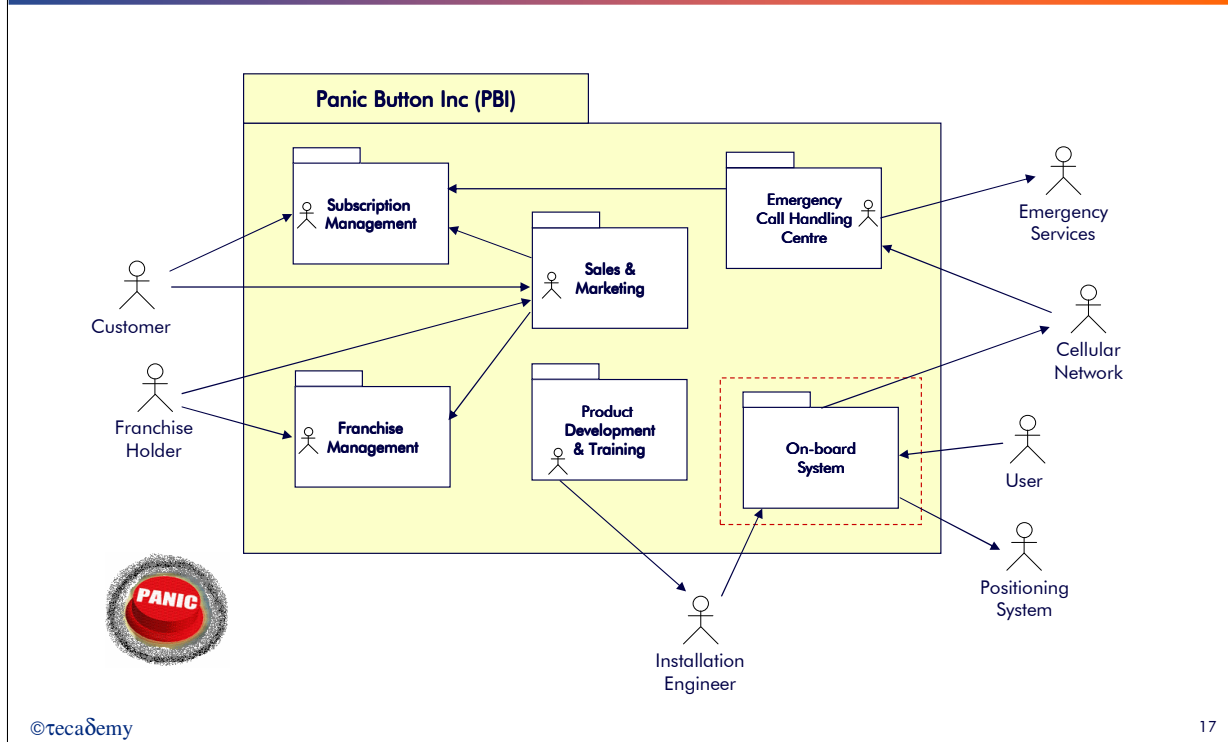
Problem Statements should:

1. identify the owner of the problem.
2. be Concise. The longer the document, the less likely it is to be read. One side of A4 is ideal...
3. contain **no** implementation detail. State *what* the problem is - not *how* the solution will look...
4. be written by, and in the language of, the owner of the problem rather than the solution provider...

An example or two describing an occurrence of the problem can be useful.



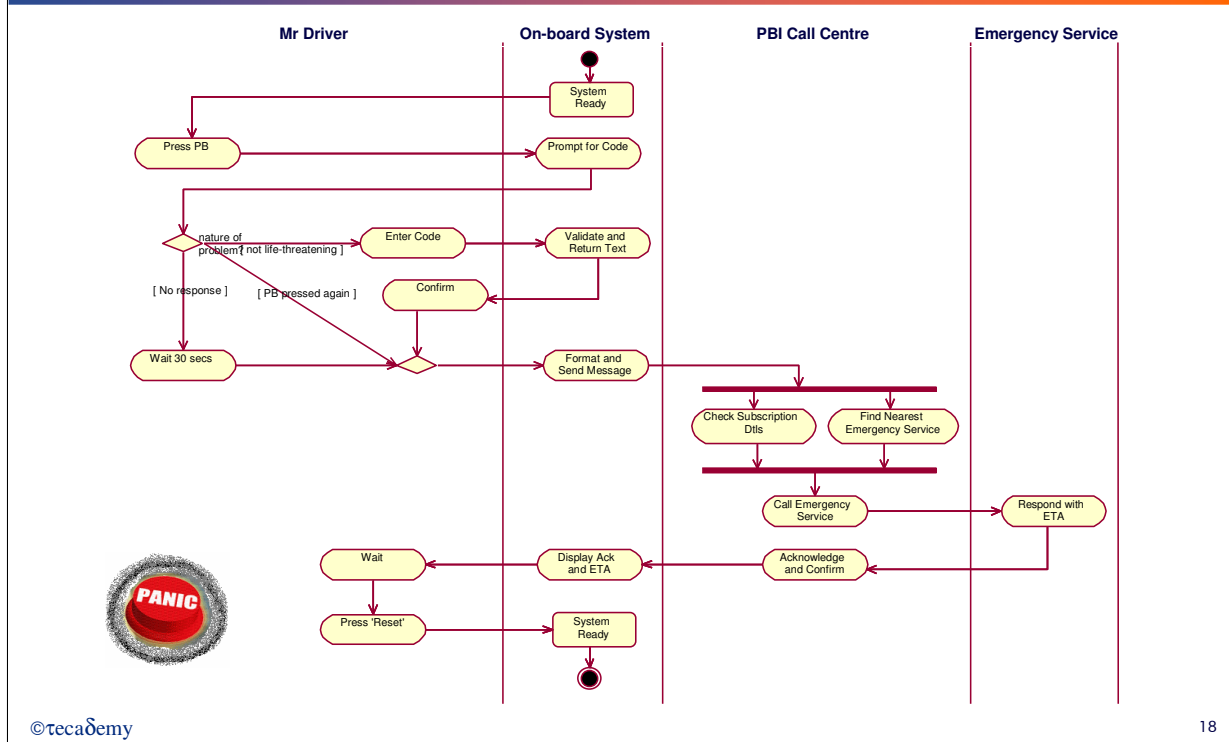
## Solution 2 – Business Context Diagram

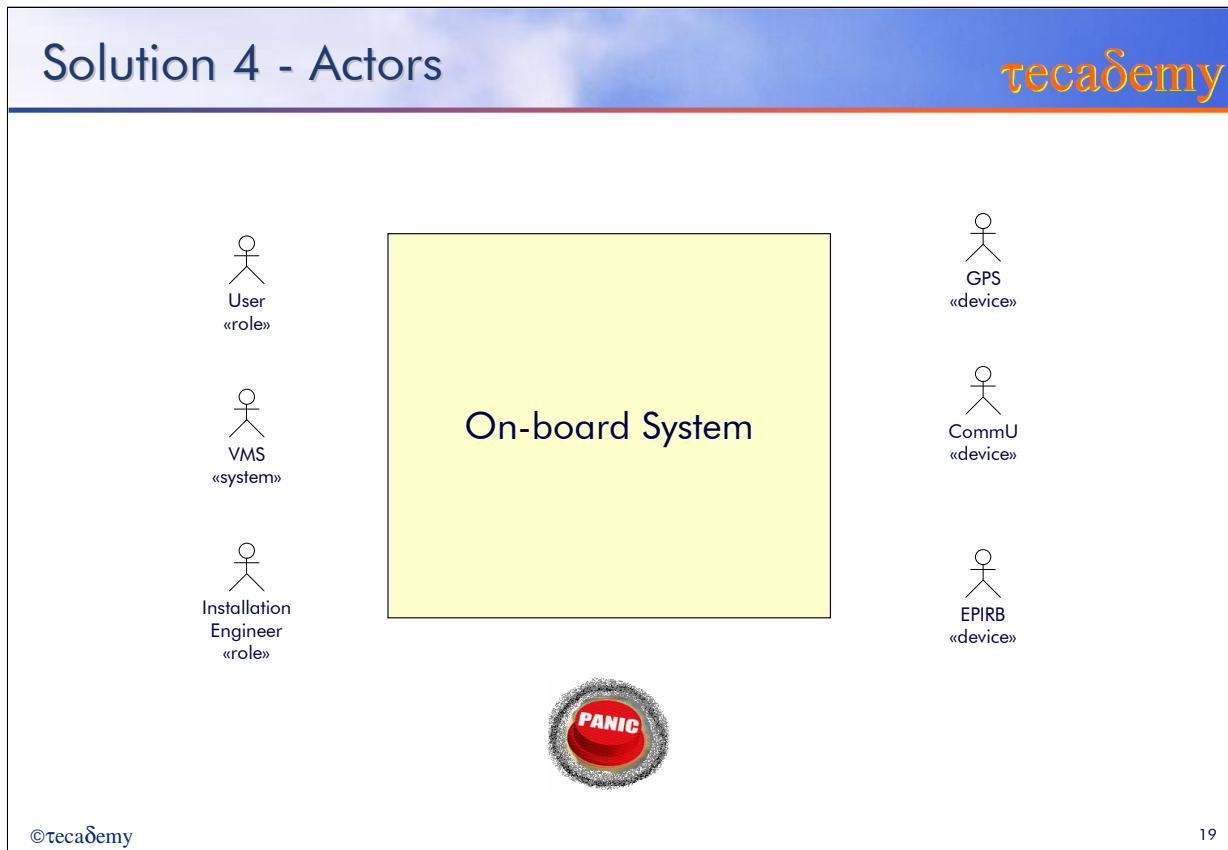


A customer first contacts Sales, subsequent enquiries direct to Subs Mgt.  
 Franchise Holders (e.g. Halfords) first contact Sales, subsequent enquiries to Franchise Mgt.  
 Installation Engineers receive training from PBI, then install/configure system in Vehicle.  
 User calls for help using system in vehicle.  
 On-board system uses GPS to fix current position.  
 On-board system uses cellular network to pass emergency message to PBI call centre.  
 PBI call centre checks subs info and contacts appropriate emergency services.

The dashed line indicates the portion we are interested in – our *Scope*.

# Solution 3 – Activity Diagrams





### Glossary

**User** - Anyone initiating a call to the emergency services. Need not be a **Subscriber** (i.e. subscriber could be company supplying company car)

**VMS** - The Vehicle Management System (VMS) is the vehicle's on-board diagnostic facility. It monitors the state of the engine and major systems such as brakes, electronics and safety. An interface is provided to allow service engineers to access the information provided by the VMS. This interface is used by the PanicButton system between services.

**Installation Engineer** - Responsible for the installation, configuration, maintenance and repair of the Panic Button system.

**GPS** - The Global Positioning System (GPS) receiver uses the difference in time signals (transmitted from a constellation of 3 or more satellites in Earth orbit) to calculate the latitude, longitude and altitude of the receiver. GPS satellites also transmit the time, derived from the on-board atomic clock which is accurate to one ten-millionth of a second.

**CommU** - The Communications Unit (CommU) handles communications with the cellular network.

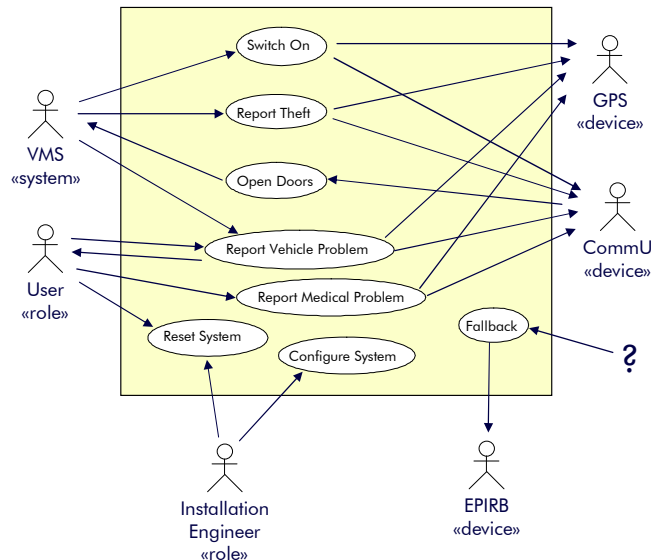
**EPIRB** - The Electronic Position Indicating Radio Beacon (EPIRB) is a device that transmits a pre-programmed ID on the 121.5MHz band. This band is monitored by aircraft and the emergency services, which can home in on the transmission to its source.

This suggested solution shows that a decision has been made to restrict the system boundary to that which defines the on-board component of the system, that is to say the bit that hosts the Panic Button 'intelligence'. Things like the GPS and the VMS are likely to observe protocols that are outside of our control and are likely to be available more cheaply as off-the-shelf components – so why include them as our responsibility?

The diagram also implies issues such as subscription maintenance and call management will not be included in our domain. This doesn't mean they are not important – just that we are not going to assume responsibility for their development. The diagram above is a subset of the Business Context Diagram from the previous exercise, which details the organisational infrastructure of the whole 'Panic Button' operation, and in which entities like 'Subscriptions Management' and 'Emergency Services' figure prominently.

As far as we are concerned, in our corner of the Panic Button universe, we send messages out to, and receive responses from, the CommU – we leave any dialogue between the CommU and the Emergency Services to somebody else to worry about. This nicely decouples us from any implementation decisions regarding the communications technology, which will make it easier if in the future we decide to move from terrestrial comms systems to, say, a satellite-based system as envisaged in the problem statement. As long as we can talk to our CommU and understand its responses, we'll be happy.

## Solution 5 – Use Cases

**Use Case Name** - Report Theft**Actor(s)** (primary [secondary]) - VMS [CommU, GPS]**Preconditions** - ((Panic Button system is live) AND (VMS.authorised\_start = FALSE))OR((Panic Button system is live) AND (CCM.authorised\_user = FALSE))

**Main flow** - The use case begins when the vehicle is started using an unauthorised procedure (e.g. door forced, ignition switch by-passed). If five minutes elapse without userID clearance, the system retrieves the vehicle ID and start position and formats a '**Vehicle** <vehicleID> **stolen from** <position>' message. The PanicButton service provider is dialled, and on connection, the message is transmitted. When acknowledgement is received from the service provider, the vehicle plots its position at 30-second intervals, and the results are transmitted via the CommU as they are received. The console continues to display the date and time as normal. The use case ends when the system is reset by an authorised user.

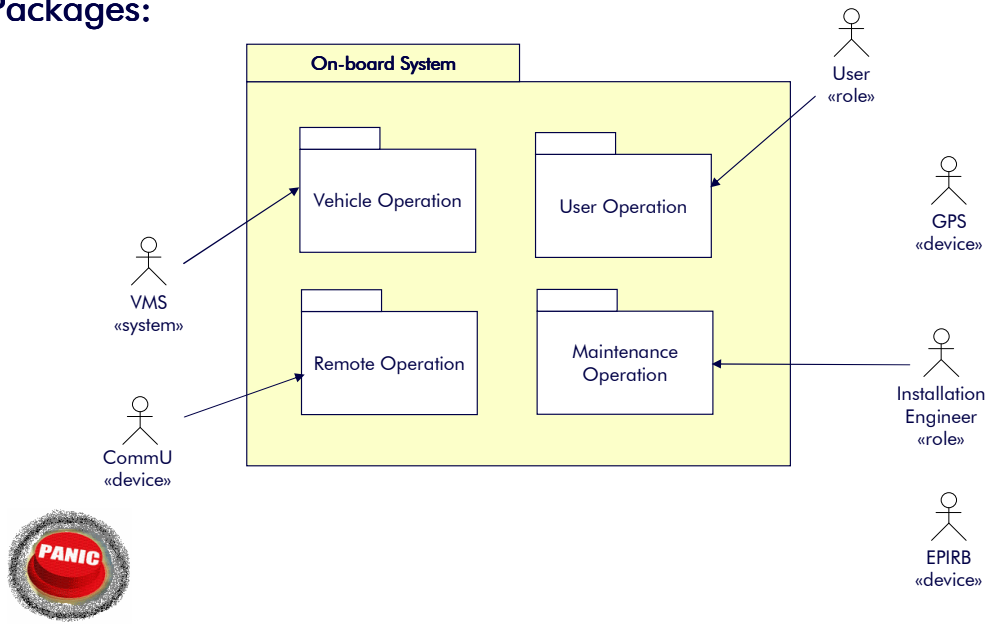
**Alternate flow** - Cellular signal is lost / **Action** - Restart use case**Alternate flow** - System is disabled / **Action** - Initiate use case 'Fallback'**Postconditions** - (Panic Button system is live) AND (CCM.authorised\_user = TRUE)

Note: The use case 'Fallback' appears to have no primary Actor. All this means is that one has not yet been identified – it can be useful to use question marks on a diagram to highlight areas where questions need to be asked.

A full use case template is given in the appendix.

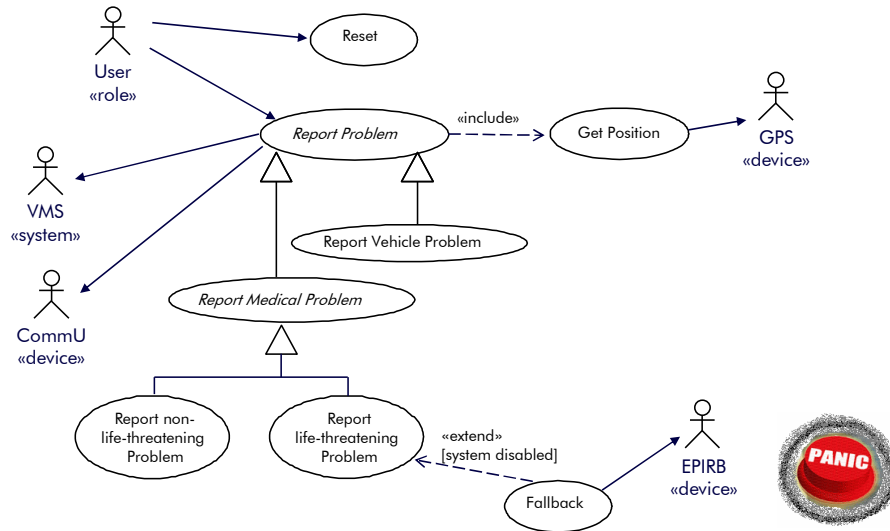
# Solution 6 – Advanced Use Cases

## Packages:



## Solution 6 – Advanced Use Cases

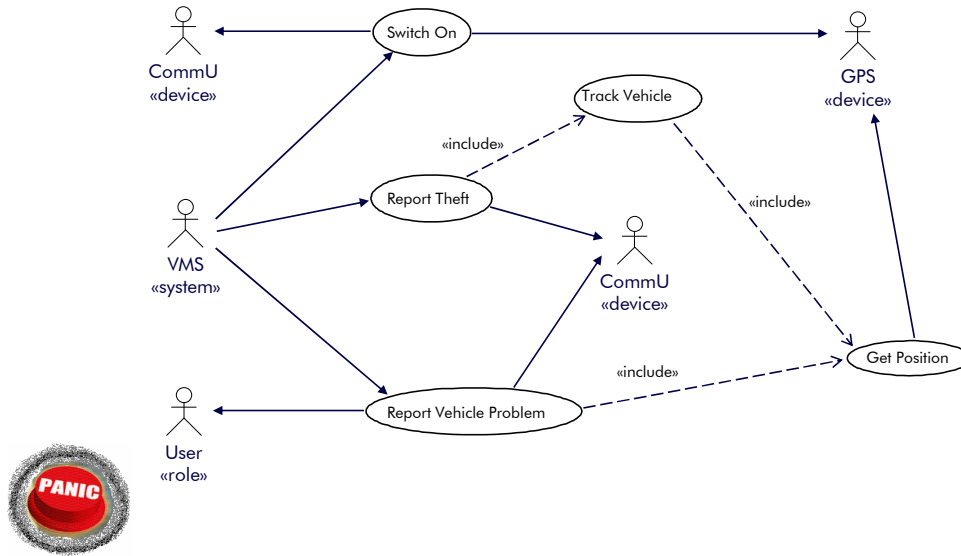
### User Operation:



Some of the use cases have some common behaviour. For example, Report Medical Problem and Report Vehicle Problem need to determine the vehicle's position. A 'Get Position' use case could be created and used by all use cases that need a position fix. Examples of *include*, *extend* and *generalise/specialise* are given on this and the next page for the User Operation and Vehicle Operation packages.

## Solution 6 – Advanced Use Cases

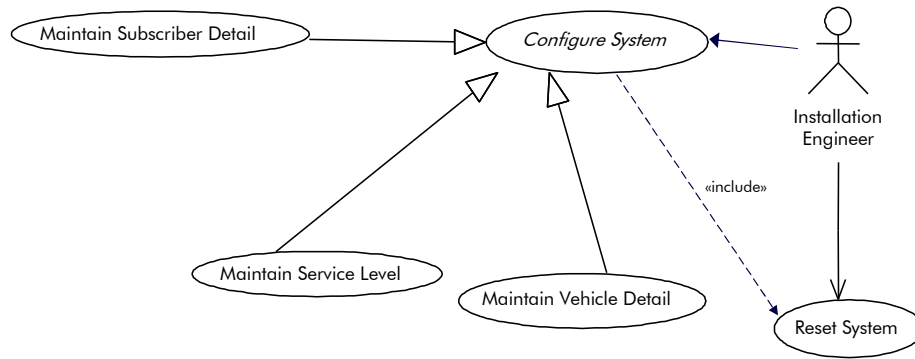
### Vehicle Operation:



Repeating an Actor (e.g. CommU, above) is allowed and can help to minimise crossing lines. The only rule is that if two or more actors have the same name, then they are the same Actor.

## Solution 6 – Advanced Use Cases

### 🎯 Maintenance Operation:



## Solution 7 – ‘Candidate’ Classes

<b>ConsoleIF</b> «boundary»	Accepts input from the console keys and sends output to the console’s display, lights, buzzer.
<b>VMSIF</b> «boundary»	Polls the Vehicle Management System.
<b>EPIRBIF</b> «boundary»	Activates the Electronic Position Indicating Radio Beacon.
<b>GPSIF</b> «boundary»	Polls the Global Positioning System receiver.
<b>CommUIF</b> «boundary»	Communicates with the Communications Unit.
<b>CallManager</b> «control»	High-level supervisor.
<b>SubscriberDetail</b> «entity»	Includes the subscription number and service level.
<b>VehicleDetail</b> «entity»	Includes registration number, make, model and colour of vehicle.
<b>VehicleState</b> «entity»	The current operating status of the vehicle, as determined by the VMS. See VMS protocol manual.
<b>AuthorisedUserDetail</b> «entity»	Includes personal identification number (PIN)
<b>ProblemCode</b> «entity»	Matches 4-digit problem code with descriptive text. E.g. 3232 = “I am being hi-jacked”.
<b>ProblemCategory</b> «entity»	Matches ProblemCategory, e.g. Medical, Vehicle, Security, with appropriate phone number.
<b>EmergencyMessage</b> «entity»	What gets sent to the service provider. Contains subscriber, user and vehicle details, vehicle state, nature of emergency, current position.
<b>Acknowledgement</b> «entity»	Text returned from service provider on receipt of message.
<b>Position</b> «entity»	Geographic location of vehicle. E.g. Latitude and Longitude, 6-figure Ordnance Survey grid reference.
<b>Clock</b> «utility»	Gives current time.

These are some suggested candidates, drawn from a reading of the Problem statement, the Use Cases and discussions from the customer. (What do you mean, you didn’t talk to the customer?) You may find the most difficult thing about this exercise was agreeing on a name for a class – getting good names that properly reflect the semantics might be tricky. Be prepared to refine names as your understanding of the domain improves. If the names you have chosen don’t appear below, don’t worry. Read the glossary entries – the chances are you’ve got it, but named it differently. You may have some that don’t appear here, or you may have fewer. This is inevitable whenever different groups are working on the same problem. The solution for this is *not* to confine the exercise to one group! Use the differences to generate questions.

**TIP:** When modelling ‘for real’, split your team into two or three groups, each supplied with a whiteboard, pens, Post-It notes and a literate person. Post-Its allow for classes to be added and grouped together easily. Each group works on the problem for one hour, then presents their solution to the other groups. Criticize, challenge, debate. Where your solutions are similar is not the problem. It’s where they don’t overlap that the debate needs to be focused. Differences should be resolved before moving on.

Remember – these are *not* classes, only *candidates* for class status. To qualify for class-dom, we must demonstrate that instances of the above contribute to the behaviour required from the system. It is not to be considered a complete list – some candidates may turn out to be redundant, and we may need to add new ones in order to get things done at all. *Scenarios* allow us to ‘audition’ the candidate classes and decide whether or not they are useful.



## Solution 8 – Scenarios (version 1 of 3)

***USE CASE – Report Problem***

***SCENARIO – User reports medical emergency (system level)***

The User enters an emergency code into the System  
The System converts the code into a text string  
The System displays the text to the User  
The User tells the System to confirm the message  
The System asks the GPS for the current position  
The GPS sends the position to the System  
The System asks the VMS for the Vehicle State  
The VMS sends the Vehicle State to the System  
The System creates the emergency message  
The System sends the message to the CommU  
The CommU sends the message to PBI call centre  
PBI sends acknowledgement to CommU  
The CommU returns acknowledgement to the System  
The System displays the acknowledgement and time to the User.



This scenario describes the interaction between the system as a whole and the actors, from the primary actor's viewpoint. It can be useful to focus on the external view at first in order to establish a structure for the dialogue. Once such a structure has been settled, we will want to explore further the interactions between the objects inside the system boundary.

A useful first refinement is to replace actors with the system objects that communicate with them, i.e. the «boundary» objects. It may aid readability to retain some interaction with actors, particularly primary actors. These could be distinguished from internal interactions by rendering them in italics. An example refinement follows on the next page...

## Solution 8 – Scenarios (version 2 of 3)

### **USE CASE – Report Problem**

#### **SCENARIO – User reports medical emergency (first refinement)**

*The User enters an emergency code into the ConsoleIF*  
The ConsoleIF passes the code to the CallManager  
The CallManager converts the code into a text string  
The CallManager passes the text string to the ConsoleIF  
*The ConsoleIF displays the text to the User*  
*The User presses the ConsoleIF's Panic Button*  
The ConsoleIF tells the CallManager to send the message  
The CallManager asks the GPSIF for the current Position  
The GPSIF sends the Position to the CallManager  
The CallManager asks the VMSIF for the VehicleState  
The VMSIF sends the VehicleState to the CallManager  
The CallManager formats the EmergencyMessage  
The CallManager sends the EmergencyMessage to the CommUIF  
*CommUIF sends message to PBI call centre*  
*PBI sends acknowledgement to CommUIF*  
The CommUIF returns the Acknowledgement to the CallManager  
The CallManager passes the Acknowledgement to the ConsoleIF  
*The ConsoleIF displays the Acknowledgement to the User.*  
The CallManager passes the time to the ConsoleIF  
*The ConsoleIF displays the time to the User.*



This treatment has de-coupled the system from the actors and has decomposed it into several boundary-type objects and a control-type object. This object, the CallManager, attracts attention – very busy, very mysterious! What is going on inside? How is it doing its job, and what other objects is it using? Where does it get the information it needs to, say, 'convert the code into a text string'??

***Solution 8 – Scenarios (version 3 of 3)***

***USE CASE – Report Problem***

***SCENARIO – User reports medical emergency (another refinement)***

*The User enters an emergency code into the ConsoleIF*

The ConsoleIF passes the code to the CallManager

The CallManager asks the ProblemCode to convert the code into a text string

The ProblemCode passes the text string to the CallManager

The CallManager passes the text string to the ConsoleIF

*The ConsoleIF displays the text to the User*

*The User presses the ConsoleIF's Panic Button*

The ConsoleIF tells the CallManager to send the message

The CallManager asks the ProblemCode for the ProblemCategory

The ProblemCode passes the ProblemCategory to the CallManager

The CallManager asks the ProblemCategory for the phone number

The ProblemCategory passes the phone number to the CallManager

The CallManager asks the SubscriberDetail for subscription number

The SubscriberDetail passes the subscription number to the CallManager

The CallManager asks the VehicleDetail for vehicle details

The VehicleDetail passes the details to the CallManager

The CallManager asks the VMSIF for the VehicleState

The VMSIF sends the VehicleState to the CallManager

The CallManager asks the GPSIF for the current Position

The GPSIF sends the Position to the CallManager

The CallManager formats the EmergencyMessage

The CallManager sends the EmergencyMessage and phone number to the CommUIF

*The CommUIF dials the number and waits for contact*

*The CommUIF sends the message and waits for an Acknowledgement*

The CommUIF returns the Acknowledgement to the CallManager

The CallManager passes the Acknowledgement to the ConsoleIF

*The ConsoleIF displays the Acknowledgement to the User.*

The CallManager asks the Clock for the time

The Clock passes the Time to the CallManager

The CallManager passes the time to the ConsoleIF

*The ConsoleIF displays the time to the User.*

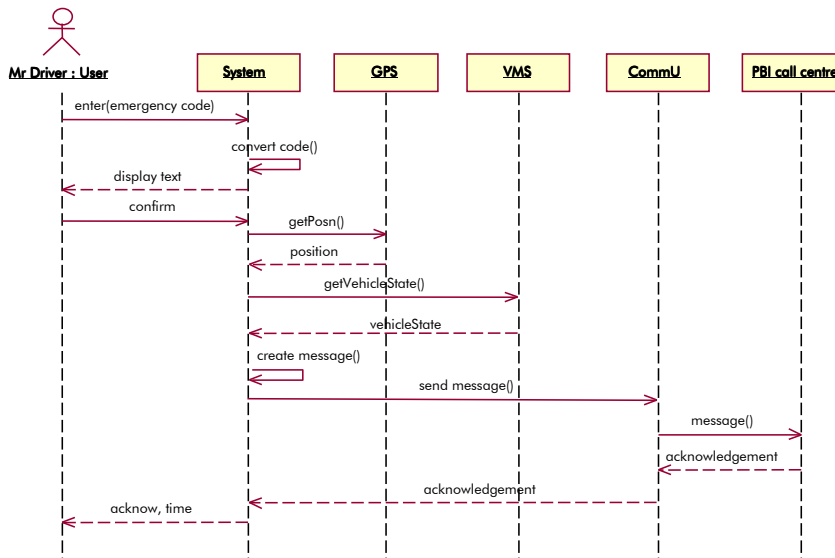


Phew! Well, this works. But just because it *works*, doesn't mean that it's the *best* way to make it work - questions need to be asked. Is this the only configuration of objects and messages that will perform this scenario? That CallManager object is still looking very busy, and seems to need to know about every other object. What are the consequences of this for maintainability or extensibility? What if the ProblemCode owned and encapsulated the ProblemCategory? Couldn't the VehicleDetail object include the mechanism for accessing the VehicleState? What if the EmergencyMessage object was given the responsibility to format and send itself? You can probably think of other challenges to this solution. Good. That's what Analysts do.

These issues would need to be explored within the wider context of other scenarios, other use cases. Consideration should also be given to the overall object architecture we are working within (especially if reusing existing objects) or working towards (how can we design in maximum reuse potential?)

Scenarios can be a cumbersome tool for exploring alternatives, but are essential in the early stages of requirements analysis. Once confidence has developed in our choice of objects, *Sequence Diagrams*, which are easier to read, can be constructed directly.

# Solution 9 – Sequence Diagrams (1 of 3)

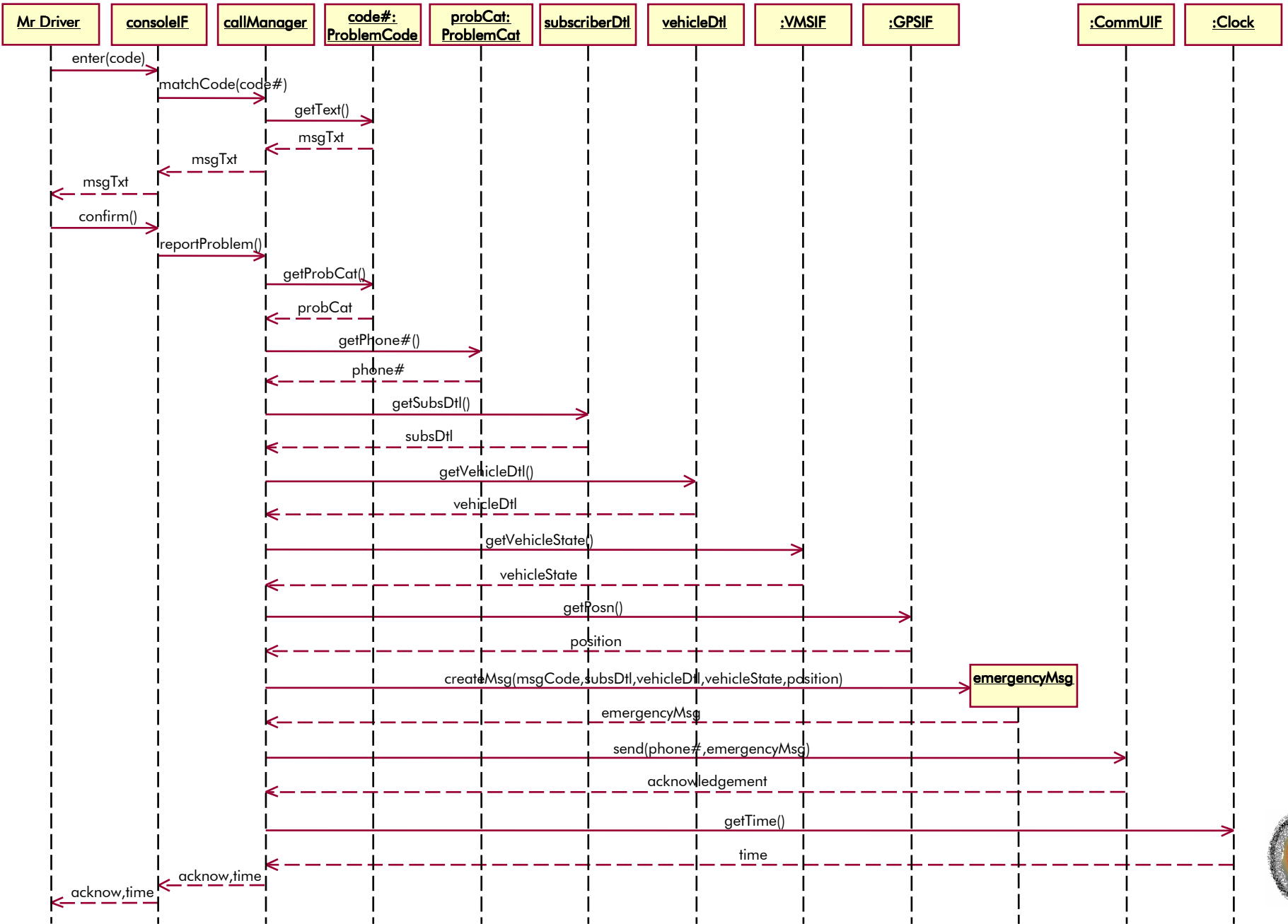


## 6.1 Additional Refinements - Sequence Diagrams 2 and 3

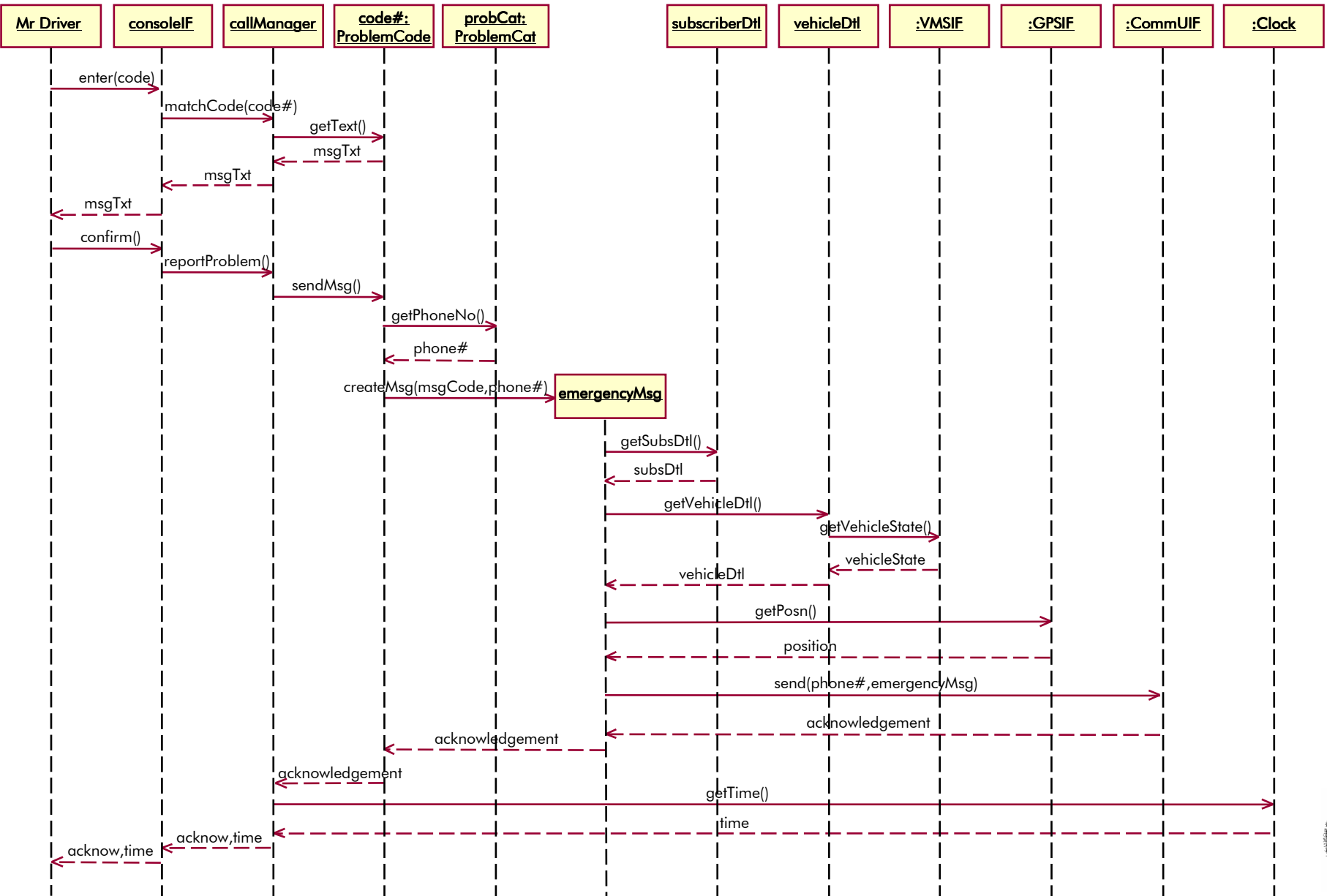
Sequence Diagram (SD) 2 is a refinement of the suggested solution, based on the more detailed scenarios from the previous exercise. This shows lower-level objects and more interactions. Because of the added detail, it has been necessary to rotate it to landscape format. (See following pages.)

Sequence Diagram 3 is an alternative to SD2, and is shown on the page following SD2. It does exactly the same thing, at least from the User's perspective, but the internal mechanisms have changed. Where SD2 was very CallManager-centric, in this version responsibility has been devolved to the EmergencyMessage object. The EmergencyMessage, which previously played a very passive role and was created by the CallManager, now creates and sends itself, managing its own collaborations to do so.

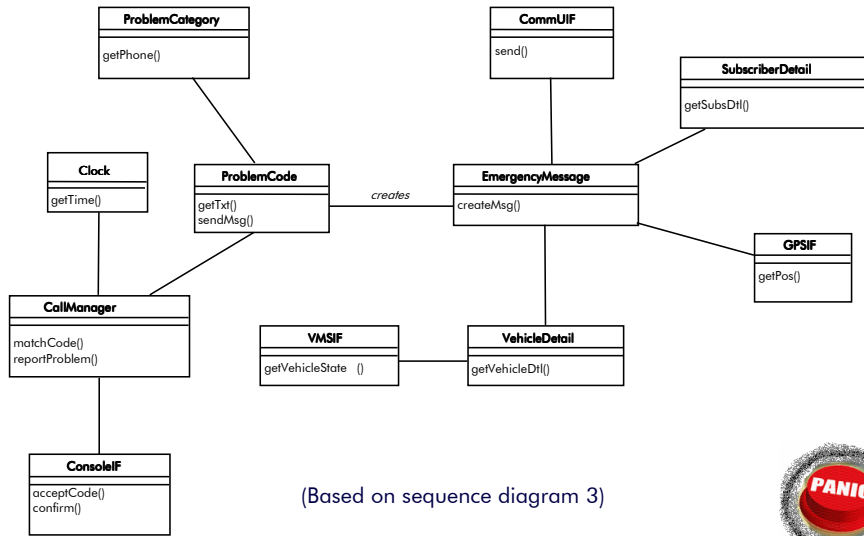
Sequence Diagram 2



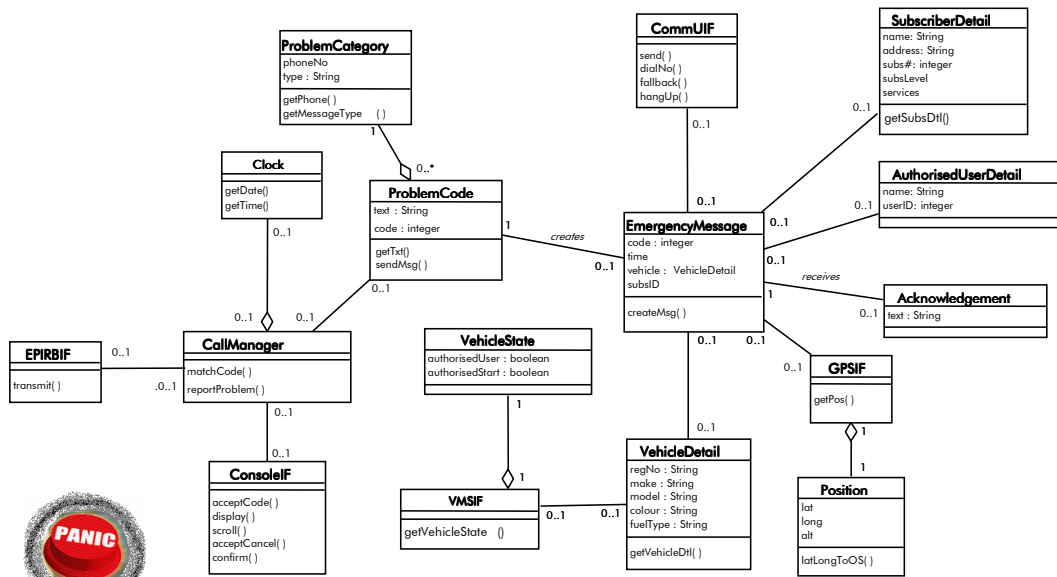
Sequence Diagram 3



# Solution 10 – The Class Diagram



# Solution 11 – Refining Associations



 Notes: