

UML

System Behaviour

UML Modelling

- Architectural
 - System structure
- Behavioural
 - System Behaviour

What sort of things need to be looked at ?

- What actions / processes a software system performs
- Why it performs such actions
- How it performs such actions
- Outcome of those actions
- Special Cases

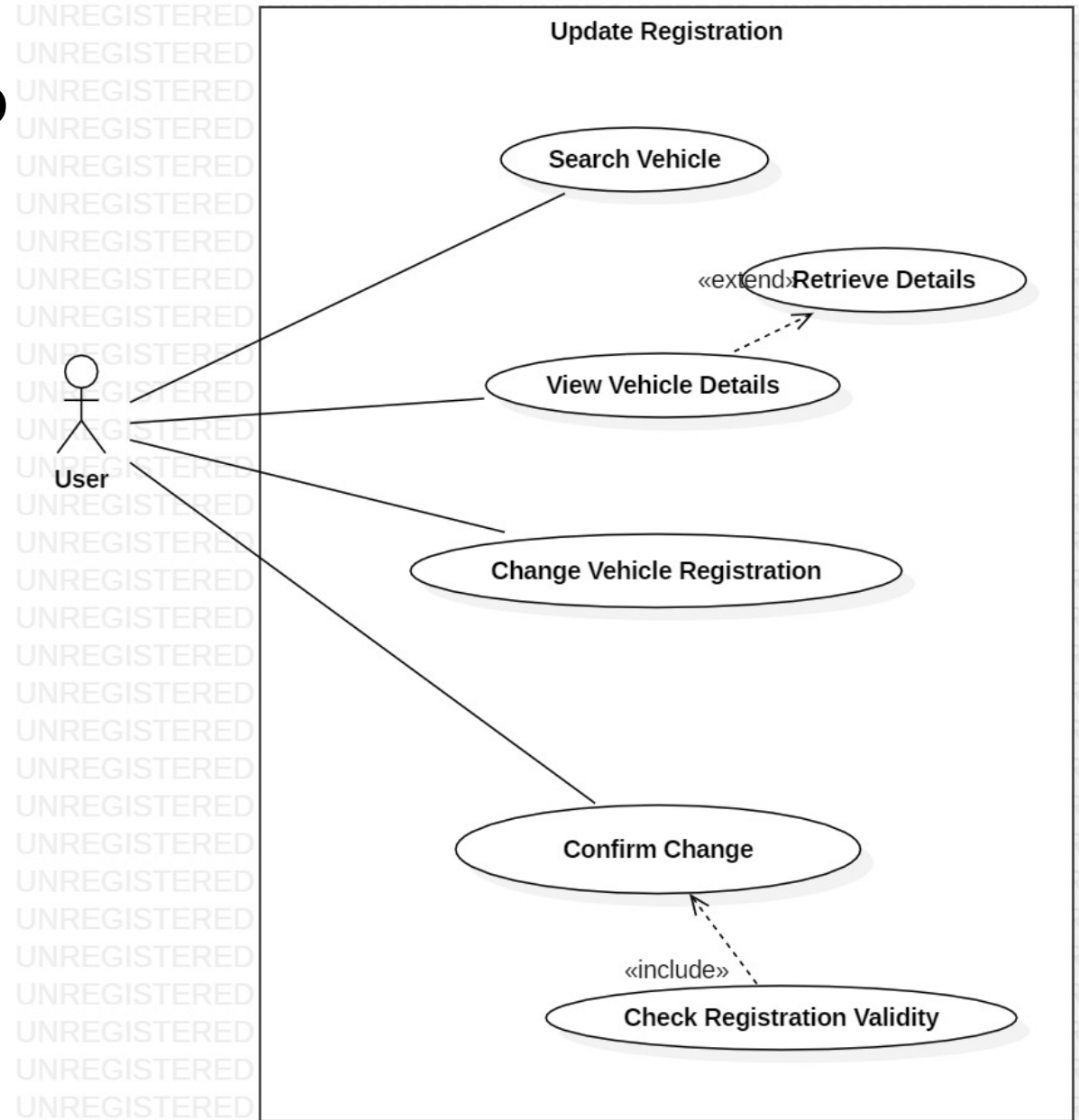
Where is the bounding box ?

- How low can you go ? – (No, we are not doing the limbo)
 - This all depends on your perspective and that of your colleagues

So what do we need ?

- Isolate each of the functionalities of a system
- Categorise or couple similar or dependent behaviours within a system
- Note the interactivity between dependent modules
- Draw some pretty pictures.....

Somewhat confusing ?

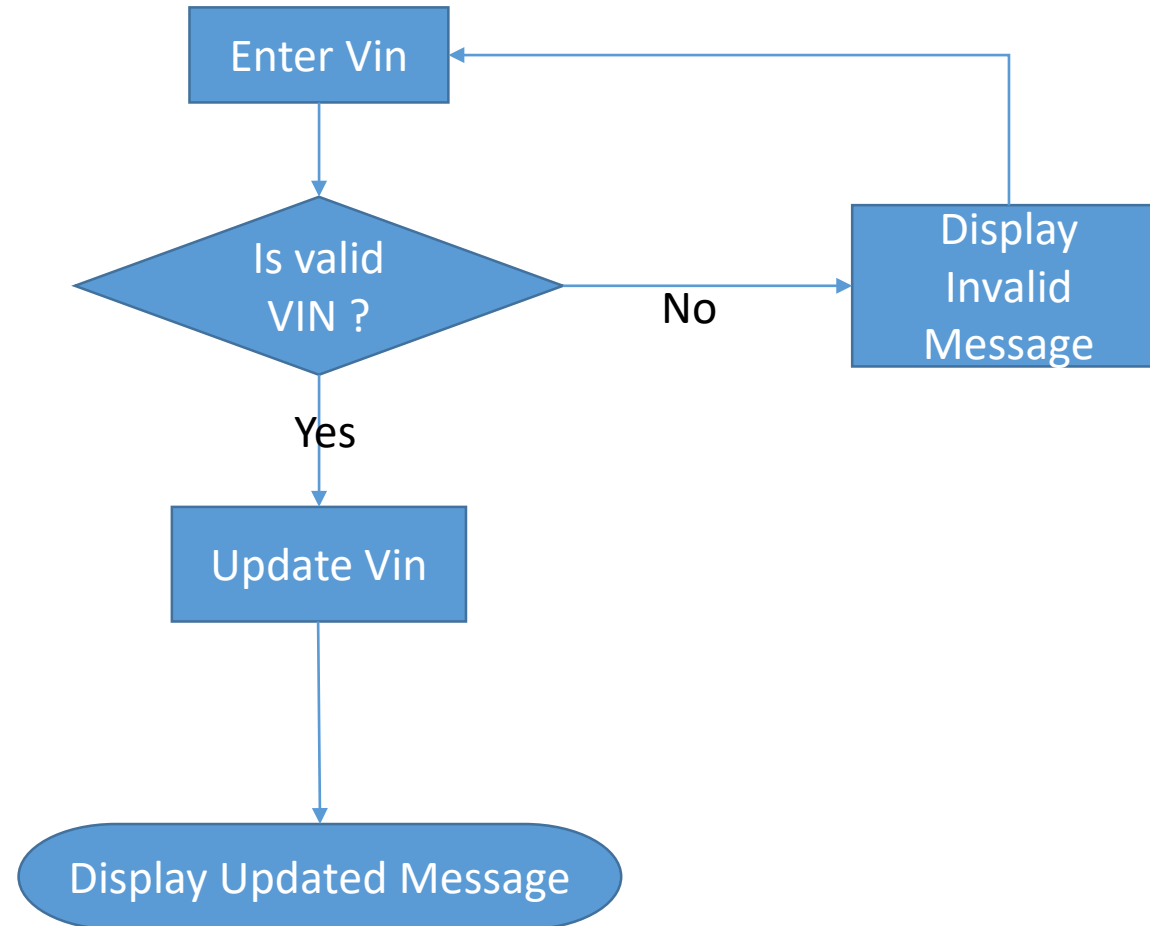


System behaviour

- How the system does what it does
- There are some examples of ways of representing this

How have we represented system behaviour ?

- Flowcharts



Behavioural documentation

- UML is an architectural description language, as such it can contain
 - Graphical depictions
 - Text

Textual representation of system

- Descriptions
- Pseudocode cards
- Naming conventions within diagrams

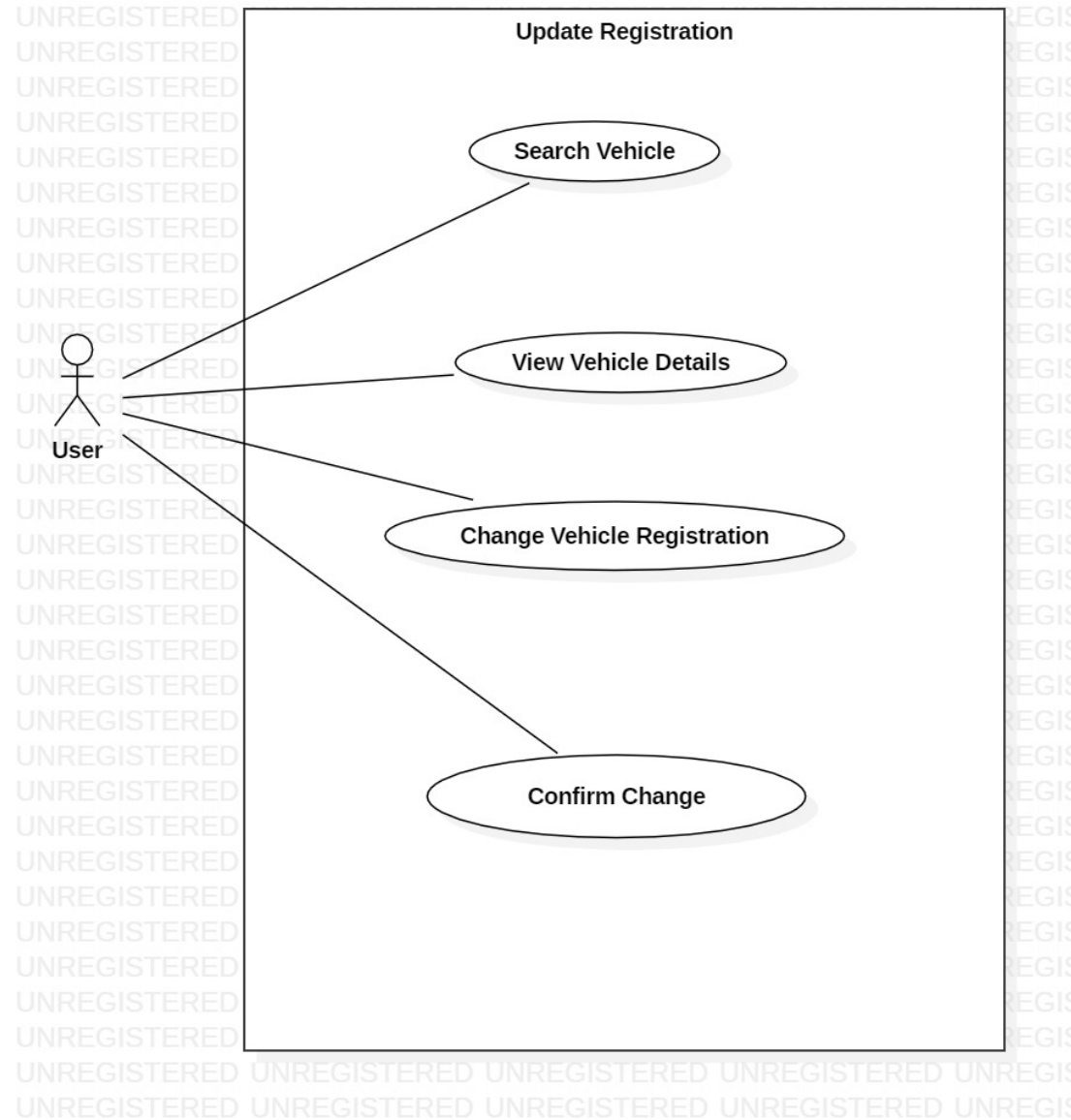
What is my system being used for ?

- What is the purpose of your system ?
- What challenge are you trying to overcome ?
- What problem are you solving ?
- All of these are first answered in the requirements analysis stages by....
 - User stories

A User Story ?

- A user story is
 - Outline of a process in a system
 - Technically a requirement
 - A users perspective on what the system is required to achieve

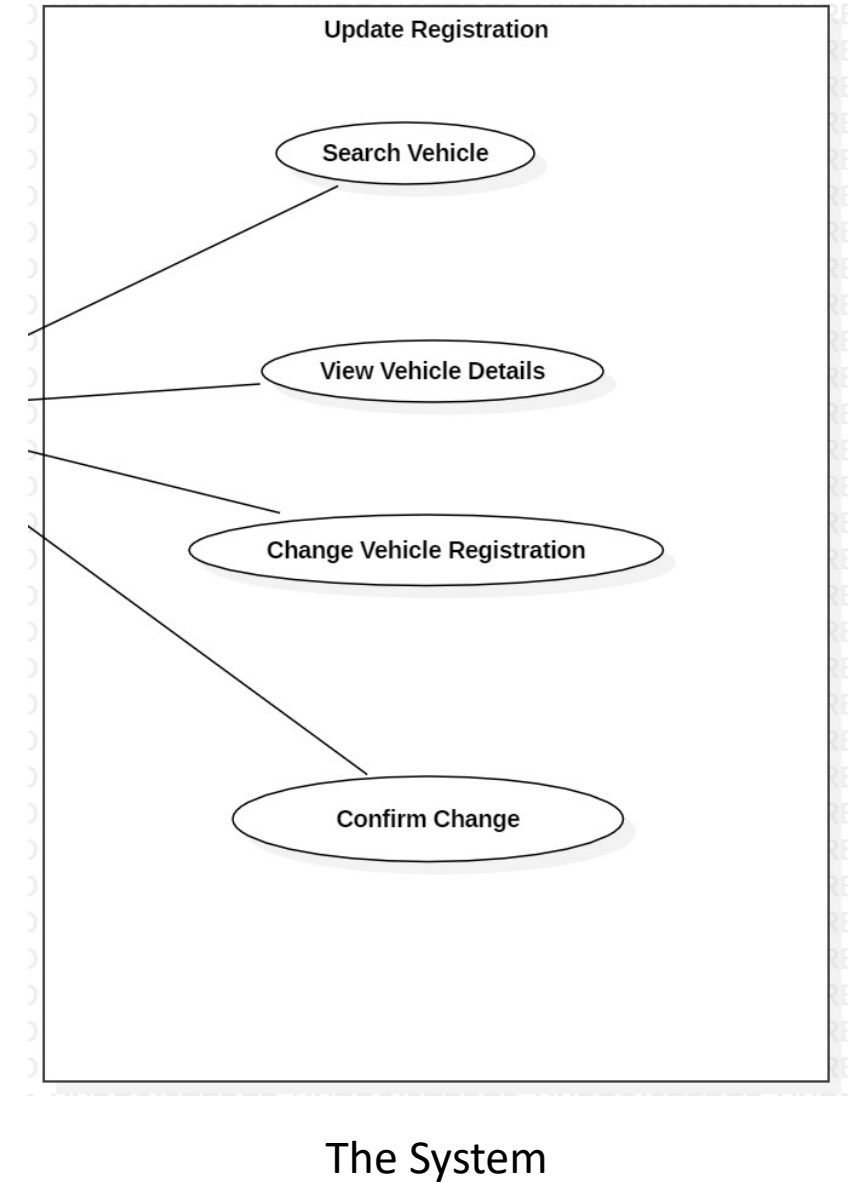
How can I paint a pretty picture of a user story?



What do all those stick men blowing bubbles mean ?

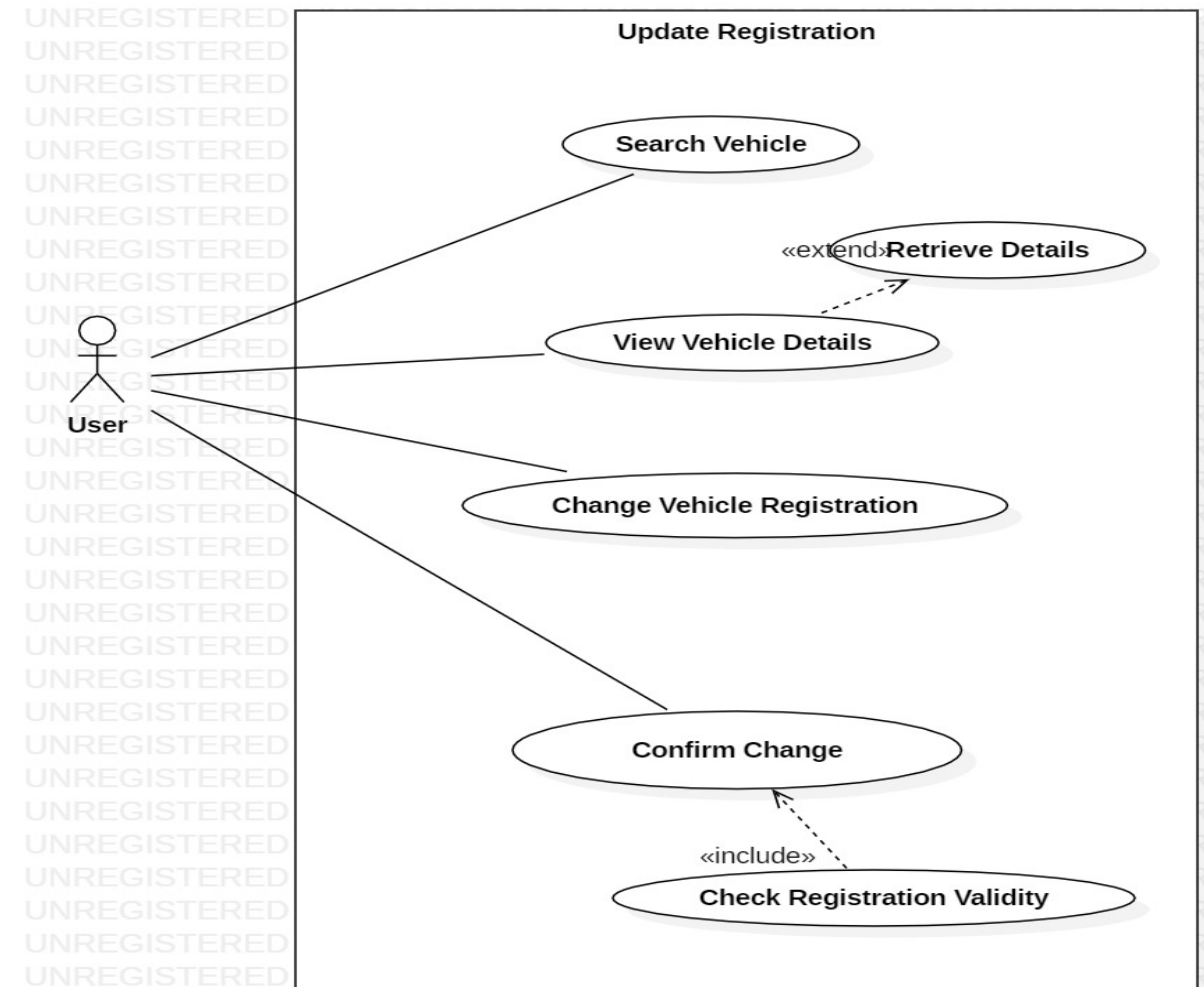
- Each element represents an entity within the system
- Use-case bubbles are the process
- The system is bounded by a box
- Association is evidenced by the arrows and connections utilised

Use-Case Notation



What about dependencies ?

- Extends – Optional external Functionality use.
- Include – Mandatory external Functionality use.



Remember that UML also includes text ?

- Use-cases can also be represented in text form
- Each of the use-cases can be thought of as individual processes and described in prose
- Text representations make it far easier to describe the events path

Use-case Descriptions

Use Case Name	Update Registration	Use Case Code	UC1
Actors	User, Customer		
Process Description			
User searches for the customer’s vehicle, selects the correct vehicle from returned results and enters the new registration for the vehicle. The new registration is checked for validity and then the data store is updated.			
Process Tasks			
1.Search for customer vehicle 2.Select correct vehicle 3.Enter new vehicle registration 4.Select save details option			
Extended Tasks			
1. System fails to locate vehicle a. Append details to the data store 2. System fails to validate registration details a. Check details with customer			
Pre-Conditions			
Customer has a vehicle User has authorisation to change details			
Post-Conditions			
Customer vehicle registered with new registration			

Process Progression

- I have outlined my processes, now what ?
 - Show the method of activation of each of those processes
 - Show the chronology of activation of each of those processes
 - Show the dependency of each of those processes

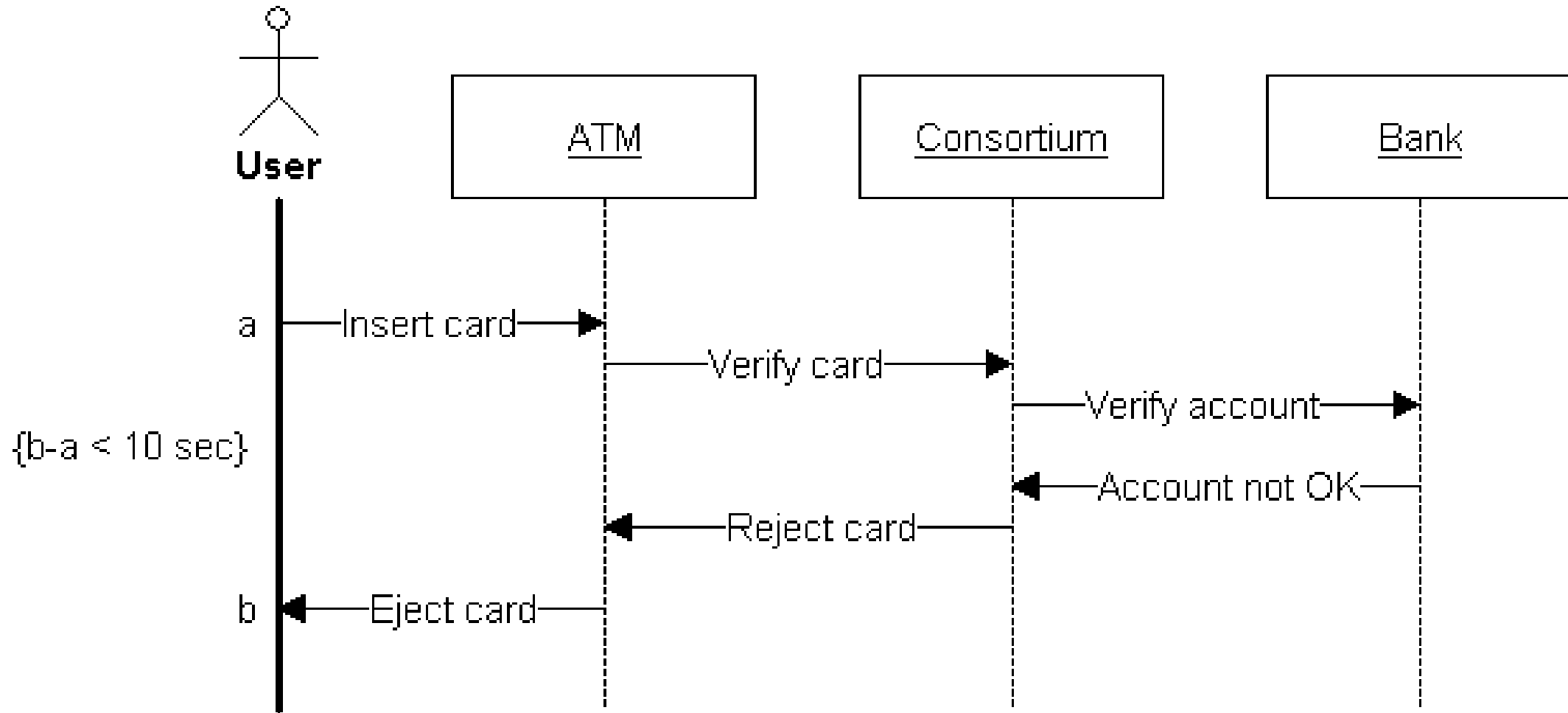
Can we show these with the use-cases ?

- No – use cases are a way of representing the functionalities of the system, not when they occur.
- What about the fact that they appear in a linear progression down the diagram ?
 - This does not show chronology, rather a means of placing the functionality on the diagram in a way that will fit

So how do we show the SEQUENCE of events ?

- With another diagram – the SEQUENCE diagram.
- The sequence diagram represents the order in which calls are made to and from each entity within the system.

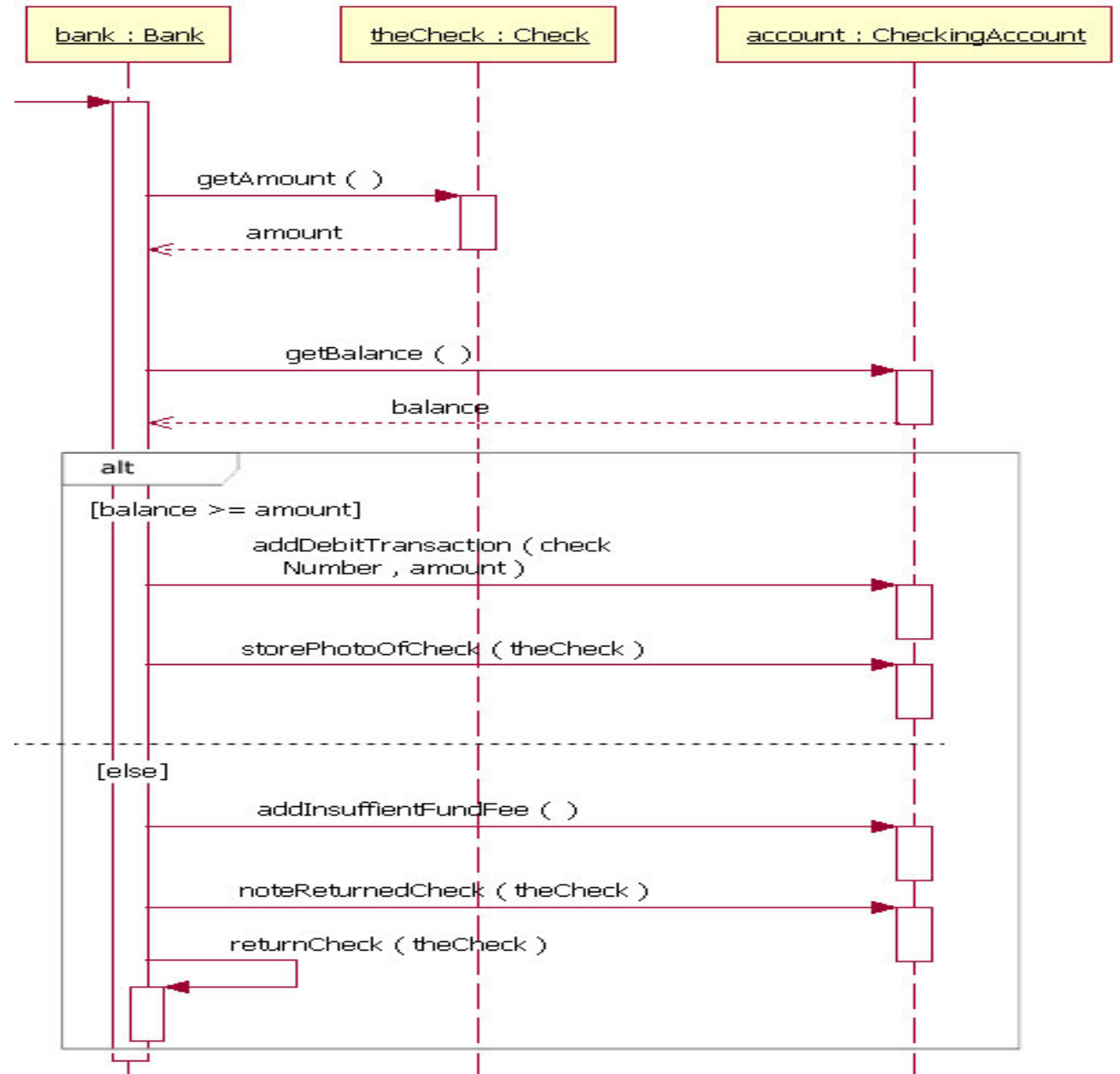
Sequence Diagram depiction



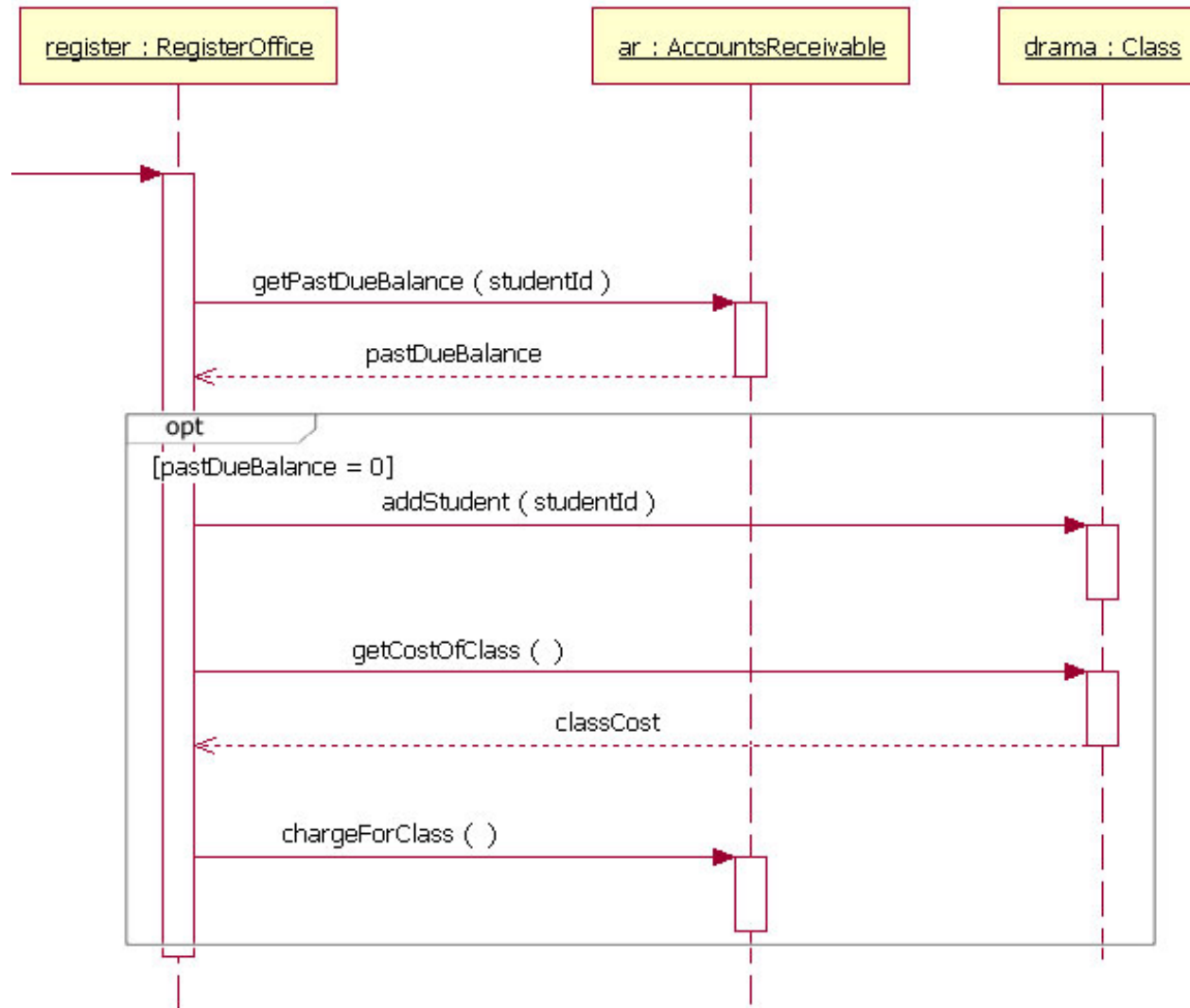
Speed-Bumps

- Sometimes things don't quite go to plan
- Sequence diagrams are equipped to deal with this utilising:
 - Alternative flows
 - Optional flows
 - Guard Conditions
- Sometimes operations are concurrent

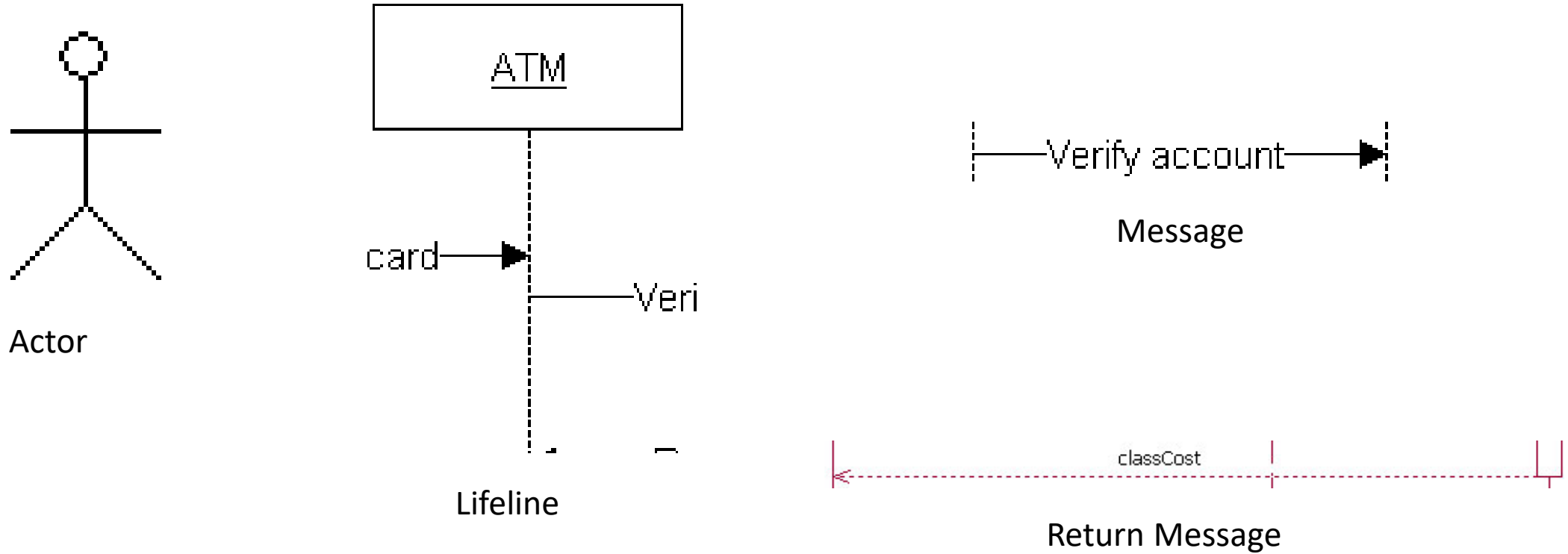
Alternative Flows



Optional Flows



Sequence diagram notation



Framing the picture

- Frames are important in annotating the level of functionality and the conditions of its execution.
- There are a number of frame types, mostly we are concerned with only two:
 - Alt – Provides view for alternative program flow situations
 - Opt- Provides view for optional program flow situations