UNIVERSITY OF Southampton

Michael Butler
mjb@ecs.soton.ac.uk
Dana Dghaym
D.Dghaym@ecs.soton.ac.uk
Thai Son Hoang
T.S.Hoang@ecs.soton.ac.uk
Yvonne Howard
ymh@ecs.soton.ac.uk

# COMP1216. Software Modelling and Design (2019-20)

## Solution Sheet 2: Functional Modelling

Issue date: 10 February 2020

This Problem Class relates to the following train ticket system. You can use **Visual Paradigm** to draw the Use Case Diagram. The intruction on installing *Visual Paradigm* can be found in the following link `https://secure.ecs.soton.ac.uk/noteswiki/w/COMP1216#Visual_Paradigm_Installation`.

## A Train Ticket System

A system must be specified for the automated purchase of train tickets from a ticket distributor. It is possible for the traveller to buy single or return tickets to available destinations, as well as weekly and monthly season tickets. The traveller will interact with the machine to specify ticket type, select destination, select payment mode (cash or credit card). A ticket purchase transaction may fail for various reasons: the distributor is out of change, out of ticket paper, credit card fails to validate, etc.

### Question 1. Use Case Diagrams

Draw a use case diagram for a Train Ticket system. The system includes two actors: a traveller, who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff.

Use cases should include: *BuyOneWayTicket*, *BuyWeeklyCard*, *BuyMonthlyCard*, *UpdateTariff*. Also include the following exceptional cases: *Time-Out* (i.e., the traveller took too long to insert the right amount), *TransactionAborted* (i.e., the traveller selected the cancel button

without completing the transaction), *DistributorOutOfChange* and *DistributorOutOfPaper*, and other exception cases that we have from the previous question.

NB: Identify the right abstraction in this example to avoid an unnecessarily complex diagram.

**Solution:**
This question can have several solutions. The following elements should be present:

- The relationship between an actor and a use case is a communication relationship (undirected solid line).
- The relationship between exceptional use cases and common use cases is an `<<extend>>` relationship.
- The exceptional use cases described in the exercise only apply to the use cases invoked by the traveler.

The following elements should be present in a *good* answer:

- All exceptions apply to all traveller use cases. Instead of drawing the relationships between these use cases and the exceptions, an abstract use case from which the exceptional use case inherit can be used, thus reducing the number of `<<extend>>` relationships at the cost of introducing generalisation relationships.
- There might be additional exceptional use case not specified in the exercise that apply to the CentralComputerSystem use cases.

**Note**: A common mistake in drawing a use case diagram is to include a "System" actor. Remember, the "System" is what you're specifying, it's the collection of all the use cases. For the ticket machine, the actors are the traveller and the central computer system (with the tariff database). If the machine accepted credit cards, the credit card bank would be another external actor. The system however is exactly the set of its use cases in the Use Case diagram.

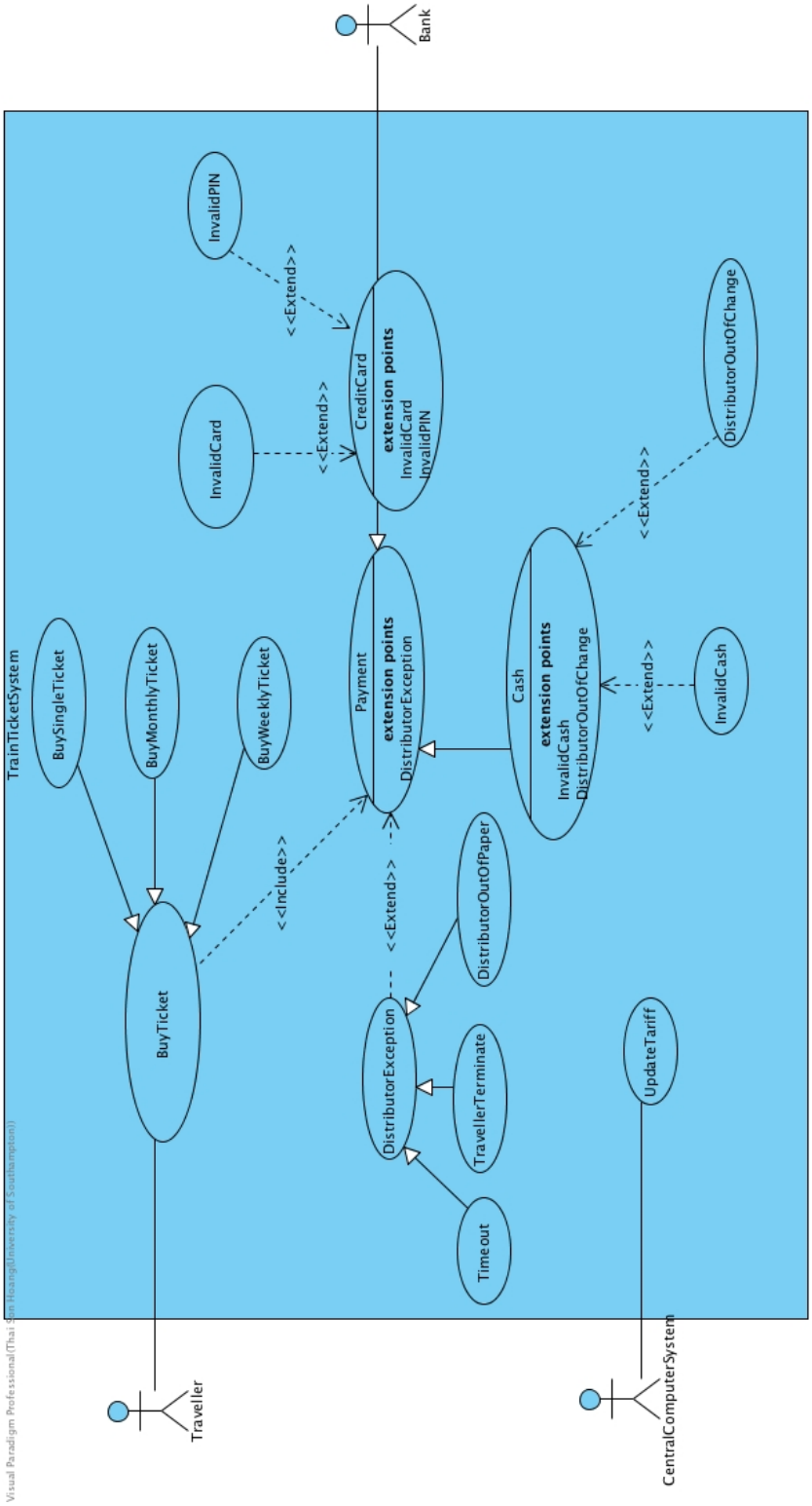


Figure 1: Use Case Diagram for Train Ticket System