Software Engineering

UML Report

Service Station

Mariano Etchart & Liam O'Neil

Table of Contents

1.0 Introduction

This report provides a detailed description of the Service Station which has been captured in the UML diagrams below, it aims to aid understanding and provide clarity to design decisions. UML Version 2.5 has been used throughout the report [1]. Visual Paradigm V14 has been used as the UML design tool.
The main features of the Service Station include:

- Fueling Bay
- Security System
- ATM
- Retail Shop
- ANPR
- Car Wash

Underlining all local systems is a single point of control, responsible for monitoring the Service Station's operations. A staff interface provides direct communication between the local Operation System and an off-site database, of which monitors stock levels.

A major focus of the design is the utilisation of a single Operation System used to monitor concurrent ancillary systems. **Concurrency** is particularly important within the Service Station as multiple fuel bays are required to be monitored and accessed simultaneously. With each Pump Station featuring a Pay at Pump system, simultaneous payments are crucial to the business model.

Each of the features were included in the main software control regardless of how decentralized or centralised they were. The only ancillary system which was completely excluded from the software control was the ATM. This was a design decision as it involved access to software which is most likely managed by a 3rd party provider.

2.0 Use Case Descriptions

The following descriptions accompany the system use case diagram shown in Figure 1.

2.1 Withdraw Fuel

A customer can initiate fuel withdrawal by picking up the nozzle, the pump detects this action and sends a fuel request to an employee in the shop [2]. Once this request is granted, the customer is allowed access to fuel and the pump is turned on. Whilst the nozzle is in use, the pump system displays the cost per litre, the litres withdrawn and the current cost. To pay for the fuel the customer must enter the shop to make a purchase.

2.2 Purchase From Shop

Purchase from shop is a general use case which encompasses any scenario where a customer requires to make a payment to the shop, be this any combination of fuel, in-shop goods or a car wash . When purchasing from the shop the user presents any in-shop items for purchase, specifies whether a fuel payment is required and can also purchase an on-site car wash token. The employee is responsible for adding or removing items to the customer's virtual basket, requesting a payment type and confirming the sale.

2.3 Ordering of Stock/Fuel

A part of the weekly duties of the Service Station manager is to monitor stock level [2]. To do this the Manager is able to query the Shop Database for low stock. Upon discovering all items of low stock the manager is able to submit an order report to the company head office. In the case of fuel, the Operation System monitors the reservoir tank level in real time and raises a flag if fuel level falls below a margin. The manager can then order fuel accordingly.

2.4 Pay at Pump

To pay at pump the customer must first insert a bankcard to the Pump Station before lifting the Nozzle. Once detected, the card will then need to be authenticated by the transaction system through receiving a correct PIN from the user. The Pump Station stores the card information and allows the user to operate the pump freely up to a defined limit. When the nozzle is put down, the transaction system calculates the total cost and charges the card.

2.5 Wash Car

During a shop purchase the customer can request a car wash, the employee can then add a 'car wash' item to the customer's basket. Once purchase is complete a wash token code will be generated and printed on a receipt. Upon printing the receipt the customer will have a time period of 7 days to enter the token to the car wash terminal, upon entering the code the customer will be able to drive inside the car wash in order to initiate it.

2.6 Authorise and Log Transaction

Whenever a payment is made within the petrol station, the customer will specify a payment type, if a card payment is made the transaction system must: validate the card, validate an entered pin and communicate with the bank in order to complete the payment transaction. If payment is made by cash, the system will allow the employee to operate the till. By legal requirement the system must provide an invoice of said transaction (receipt) [3]. Upon the successful completion of a transaction the system logs: date/time, payment method and items sold.

2.7 Record Licence Plate

When withdrawing fuel from the Pump Station the customer lifts the nozzle, triggering a fuel request to the Employee. When a request is granted the ANPR camera is triggered to capture an image of the vehicle requesting pump usage. Upon capturing the image the ANPR performs image processing in order to extract the plate information. The plate details are then stored until a successful payment validation, in which the plate information can then be discarded.

2.8 View/Withdraw Funds

A customer wanting to view or withdraw funds can do so by using the ATM within the service station. After inserting a card, an authentication and validation process will be carried out by contacting the bank. The ATM will then display the options: Show Balance or Withdraw Cash. If cash is withdrawn, the ATM will contact the bank and perform the required transaction. An in-built security feature constantly checks for tampering with the ATM and flags the operation system if it is detected.

2.9 Record CCTV & Monitor Alarm

The Service Station will have multiple CCTV cameras, all of which will be constantly recording and storing their feeds on a server, with a set retention period. The security system within the building and forecourt will be safety critical and therefore a fault tolerant electromechanical system will be responsible for burglary and fire alarms. However, the system software will monitor alarm status in order to display an alert in the case of a trigger.

2.10 Record/Report Theft

Upon withdrawing fuel and placing the Nozzle back in the holstered position a 30 minute fuel payment timer is initiated. If payment is not satisfied within this time frame the employee interface is notified, if the car has driven off the employee can then record a theft [2], the manager can later decide upon the action to be taken.

2.11 Publish Monthly Sales

Every month the manager publishes a monthly sales report, from this the company head office can calculate a revenue for the service station.

3.0 Class Diagram

The Class Diagram can be seen in Figure 2. Upon establishing use cases for the system a Class Diagram was formed comprising of 16 classes:

EmployeeInterface: Represents the control medium between the employee, the ancillary systems and the database.
ManagerInterface: Inherits from EmployeeInterface, adds additional privileges to the class, including database access and stock ordering.
Basket: Responsible for accumulating items to be purchased and their total cost prior to payment.
Tannoy: Allows the Employee to speak through a microphone system.
TransactionSystem: Responsible for authorising any payment made to the service station.
Till: Inherited from Transaction System, adds the functionality of cash payments for in-shop transactions.
OperationSystem: Control class, provides a layer of encapsulation between the Employee Interface and the ancillary systems, funneling the flow of information.
ShopDatabase: Represents the database which stores all stock information, prices and transaction history for the Station.
PumpStation: Represents each fuelbay, monitoring the usage of the nozzle, requesting fuel, turning on the pump and tracking the real time fuel cost.
Nozzle: Dedicated to monitoring signals triggered by user interaction with the nozzle.
Tank: Monitors and reports the level of fuel left in the reservoir.
ANPR: Standalone system which captures the image of a vehicle requesting fuel, processes image and stores plate for access in case of theft.
ATM: Independent from Service Station system, allows customers to withdraw/view money.
Company: Represents the Head Office of the business, able to track monthly sales figures of individual Service Stations.
CarWash: On-site automated car wash facility.
SecuritySystem: Monitors the electromechanical alarm systems and controls/ CCTV footage.

3.1 Class Relationships

To appropriately display system functionality the choice of relationship between classes and how they interact had to be considered. Before the Class Diagram was defined, the concept of control was considered. Certain aspects of the station required *centralized* control whilst other, safety critical systems were better fit for *decentralised* control.

Aggregation and composition were chosen to represent a number of ownership-focussed object relationships within the diagram. The PumpStation class has multiple Nozzles and a Transaction System, where the EmployeeInterface has a Tannoy, Basket and Till. In these instances the choice of ownership focussed relationships has allowed the software control of both the Withdraw Fuel and Purchase From Shop use cases to become centralized to a single class, PumpStation and EmployeeInterface.

In instances where a class's control is decentralised or is independent of other classes, an associative relationship has been made to the Operation System. This is in order to provide information or functionality to the EmployeeInterface. The Operation System class's primary purpose is to provide a common ground between a network of otherwise unrelated ancillary systems, funnelling only necessary information to the display of the Employee Interface.

Within the system design two opportunities occurred where additional functionality was required to be given to an existing class. In the case of ManagerInterface, the ability to access more information and perform additional tasks to that required of an Employee brought the need for inheritance. This was similar in the case of Till which needed to be able to accept cash payments as well as card and hence inherited from TransactionSystem.

Multiplicity has been used to indicate the quantity of objects in scope between relationships. For example the Operation System is associated with multiple Pump Stations, although each Pump Station is only associated to a single Operation System.

4.0 Activity Diagrams

4.1 Withdraw Fuel

Depending on whether the customer wants to pay at shop or pay at pump, the activity diagram in Figure 3 illustrates the two different flows taken after the initial **decision node**. Within both, **guard conditions** have been utilized to either accept or reject a credit card or a fuel request. In a guard condition, the flow does not continue until the guard condition is met, i.e. the credit card is accepted. The pump is turned on in both cases, and the final node is reached after some form of payment.

In the case of paying at shop, an **interruptible region** is entered where there is a timer which raises a flag if allowed to finish. An interruptible region is a part of the activity flow whose processes can be terminated whenever there is an interruption raised. A payment is pending after the customer has selected to pay at shop and there is a 30 minute timer being run before a 'payment failed' flag is raised. Nevertheless, if a payment is received before the end of the timer, an interruption is caused and the timer terminates its process because a payment has been received and the flow reaches the final node.

The selection of paying at shop calls *RequestFuel()* which calls *DisplayFuelRequest()* to alert the EmployeeInterface that there is a fuel request pending. After the request is cleared *GrantFuelRequest()* and then *PumpEnabled()* is appropriately called. When the customer finishes refueling and places the nozzle back down, *DetectNozzleReturned()* is called and the OperationSystem calls *InitiatePaymentTimer()* which starts the FuelTimer. Unless that timer is interrupted, *DisplayTheftFlag()* is called and a flag is raised suggesting that fuel has been stolen.

4.2 Wash Car

A customer wanting to wash a car is shown car wash information from the EmployeeInterface. The employee then selects and adds the wash type to a basket and another guard condition is entered in order to offer the choice to add additional shop purchases to basket. This is held within guard condition until the basket is finalised. After selecting the payment method and authorizing the transaction, the control flow enters a **fork node** where the execution of two activities occurs simultaneously. The wash token is generated and printed by the OperationSystem whilst the TransactionSystem prints the receipt for the payment. This is allowed to occur in parallel because the OperationSystem is running in a completely separate execution environment to the TransactionSystem (see Deployment diagram). A **merger node** allows each of the processes to be **synchronized** before starting the 7 day timer and entering an interruptibility region. If the wash token code is entered in the keypad to initiate the wash, the timer is interrupted and the wash is started. If the timer expires, the virtual token is destroyed.

4.3 Order Stock/Fuel

Stock and fuel levels are monitored in different ways, outlined in Figure 5. In the diagram, each activity is represented with a separate initial node (multiple initial nodes are allowed in UML 2.5 [1]) as the ordering of stock and fuel happens simultaneously and independently of each other. Stock is monitored by the manager on a weekly routine. In this case, the manager uses the ManagerInterface to call *QueryStockLevel()* this displays the stock level of the desired items. The manager is then in a position to decide whether *OrderStock()* should be called. Fuel is regularly monitored by an embedded system (see Deployment diagram) which is in an infinite loop checking the fuel level with *MonitorFuelLevel()*. When a certain threshold is reached, the *DisplayLowFuel()* function is called from the OperationSystem, which alerts the ManagerInterface and the manager may accordingly use *OrderFuel()*.

5.0 Sequence Diagrams

5.1 ATM Withdrawal

The operation of the ATM is sequentially outlined in Figure 6 and consists of three lifelines: the Customer Interface (graphical user interface), the ATM backend, and the Bank. Whenever wanting to interact with the system, the customer must first insert his/her card which triggers the *DetectCardInsert()* function. An authentication process is carried out by the ATM backend by reading the chip, in order to confirm the card's legitimacy and to detect which type of bank the card is associated with. Once authenticated, the front end prompts the user for a pin. A loop is entered for three iterations where the pin is sent off to the bank for validation through a secure network with the *ValidatePin()* function. If the pin is correct, the loop is broken (break fragment) out of and account information is returned. The Customer Interface would then call *AcceptPin()* to inform the user that

the pin is valid. In the case where the pin has been rejected from the bank, *DenyPin()* is called and *PromptUser()* prompts the user again and the loop is continued. Once outside the loop, the value of the boolean *pin* is checked and the card is swallowed if it is still false. The ATM then prompts the user with the options Withdraw Cash and View Funds and then acts according to what the user chooses (alternative fragment). When wanting to withdraw cash, the user is prompted for the amount and transaction information is sent to the bank. *EjectCard()* and *EjectCash()* are appropriately called. If the user just wants to view funds, the bank is contacted and the amount is returned to the ATM backend which is then displayed by the Interface with *DisplayFunds()*.

5.2 Purchase From Shop

The use case Purchase From Shop involves many interfaces and collaborates with several ancillary systems which is outlined in a sequence diagram in Figure 7. When wanting to make a purchase from the shop, the EmployeeInterface enters a loop whilst scanning items where *AddToBasket()* is called until the boolean *BasketLoading* is set to false. The loop queries the central database for prices and then receives the prices to be calculated. An optional fragment allows the EmployeeInterface to include fuel to the basket in case the user wants to pay for fuel at the shop. The EmployeeInterface object calls the *GetFuelCost(PumpID)* function from the OperationSystem which then calls *ShowFuelCost()* from the PumpStation class to eventually return the cost to the Employee Interface. After the cost has been received, *AddFuelToBasket()* is accordingly called to add the cost of the fuel to the basket. The Basket object then calculates the basket price and displays the accumulated total for the customer to see. An optional fragment is included to allow the employee to remove items and recalculate the basket accordingly. At the end of the purchasing process, the basket total is sent to the Till class where the transaction is authorized (see state diagram) and an invoice is created by calling *PrintReceipt()*. After the customer has paid, the shop database is automatically updated with *AmendStockLevel()* where a Basket object is the argument.

6.0 State Diagram

To further capture and represent the behaviour of classes between multiple use cases a statechart has been used [4].

6.1 Withdraw Fuel

The state diagram for withdrawing fuel maps the transition between the Withdraw Fuel, Record Plate and Report Theft use cases. As can be seen from the diagram in Figure 8, the Pump Station remains in an idle state until customer interaction. To leave the idle state the customer is either required to lift the Nozzle requesting fuel and gain confirmation from an employee, or insert a card and enter a valid pin in order validate their payment details and enable pay at pump mode. If the employee grants a fuel request the ANPR camera is implicitly required to capture an image of the vehicle registration plate, this is not required for customers opting to pay at pump as their card information will be recorded. Upon returning the Nozzle and withdrawing fuel a FuelPaymentTimer is initiated, this allows the customer a 30 minute time frame to successfully pay for their fuel. If payment is not received within the 30 minute window a potential theft is flagged to the Employee, who will then have to check the CCTV and record the theft for later managerial review.

7.0 Deployment Diagram

As can be seen from Figure 10 the on-site system deployment consists almost entirely physical systems required to operate the Service Station. The physical devices consist of:

Alarm System: Electromechanical alarm system fitted within building, alarm is sounded in the event of a break in, fire or pressing the emergency stop button.
ANPR: Automatic Number Plate Recognition Camera, captures images of licence plates, processes the image and stores the registration as a string value.
DVR: Digital Video Recorder, responsible for recording, storing and playing back footage captured on the security cameras.
Security Camera: Provides forecourt and in-shop recorded surveillance.
Company Data Centre: Off-site data centre which holds the data base.
Shop Client: On-site computing systems which are heavily reliant on the external data centre to retrieve database queries.
Station Client: Runs the execution environment Operation System which performs real time monitoring of all local network systems.
ATM: Separate from the Service Station system but a facility available to customers.
Bank Server: ATM and Merchant contact bank server when authorising transactions or cash withdrawals.
Car Wash: Automated car washing facility.
Tank: Holds fuel, capable of monitoring fuel level and sending fuel low signal.
Pump Station: Interface which the customer interacts with when withdrawing fuel.

The Service Station is centrally controlled and monitored by staff through a client side [5] Employee Interface software package. Real time updates from **ancillary** systems are monitored and received through the separate Operation System environment which is ran on a Station client, allowing **concurrent** Shop Clients simultaneous access to the information. The head office data centre holds the shop database where data (i.e login credentials, fuel prices) is stored and stock levels are recorded.

The on-site 'station client' is responsible for real time monitoring of all local ancillary systems. This client will run a custom execution environment fit for system purpose. The Operation System environment will display flags alerting the employee interface of status changes to the ancillary systems alongside sending control signals to clear pump requests and trigger the ANPR.

A conscious design choice within software and hardware has been to avoid any reliance on software during a safety critical use case. As can be seen from the Deployment diagram the Alarm System relies upon fault tolerant electromechanical systems to sound an alarm or summon an emergency stop. This is to remove the chance of any potential computing performance [6], software bugs or updates hindering the effectiveness of safety critical applications. Instead the Alarm System is connected via a status signal which is triggered in the event of an emergency alerting the Operation System.

8.0 Design Limitations

The Service Station UML system design has predominantly been limited by a lack of information regarding correct or existing Service Station procedure. To build up a picture of how a Service Station system functions a former employee was interviewed prior to design. However, the information gathered was primarily oriented around an existing system and not a general model.

In industry, a task such as this would have defined requirements, time and cost constraints. There would be specified stakeholders and the task would perhaps be to improve an existing system rather than create an idealised system such as this.

A further lack of experience and familiarity with database structures and network protocols has lead to ambiguous functions such as *QueryStockLevel()* that would perhaps require more detail to fetch information from a SQL database and labeling network connections ambiguously such as 'secure network'.

9.0 References

1. UML 2.5 [Internet]. Omg.org. 2017 [cited 13 May 2017]. Available from: http://www.omg.org/spec/UML/2.5/

2. Todd A. Interview with former Service Station employee. Loughborough University; 2017.

3. Invoicing and taking payment from customers: Overview - GOV.UK [Internet]. Gov.uk. 2017 [cited 13 May 2017]. Available from: https://www.gov.uk/invoicing-and-taking-payment-from-customers/overview

4. Fowler M. UML Distilled. 1st ed. Boston: Addison-Wesley; 2004.

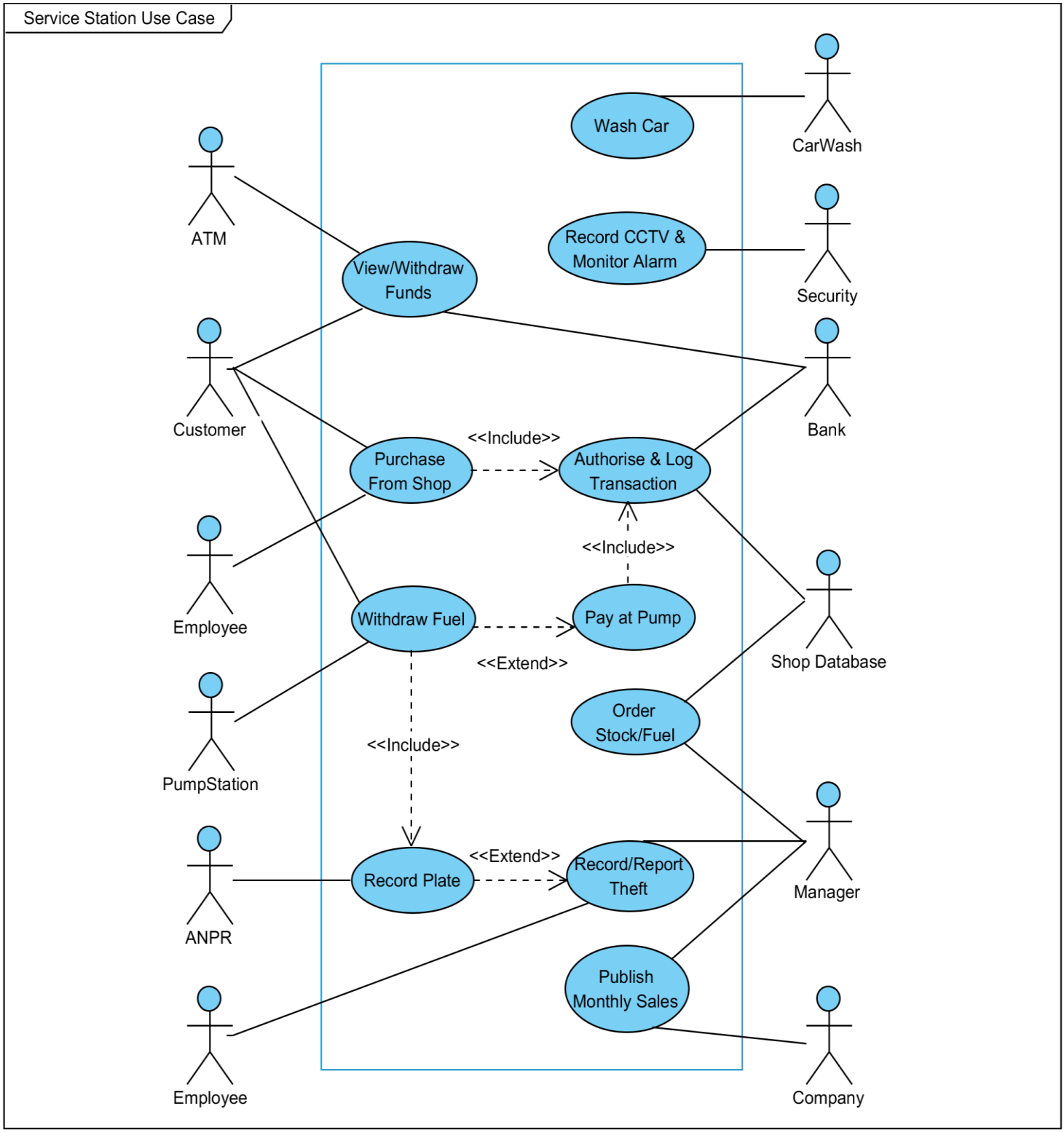
5. Rosenberg D, Stephens M. Use case driven object modeling with UML. 1st ed. Berkley, CA.:

Apress; 2007.

6. Hobbs C. Embedded software development for safety-critical systems.

Liam O'Neil - B513647, Mariano Etchart – B528854

Figure 1



Liam O'Neil - B513647, Mariano Etchart – B528854

Figure 2

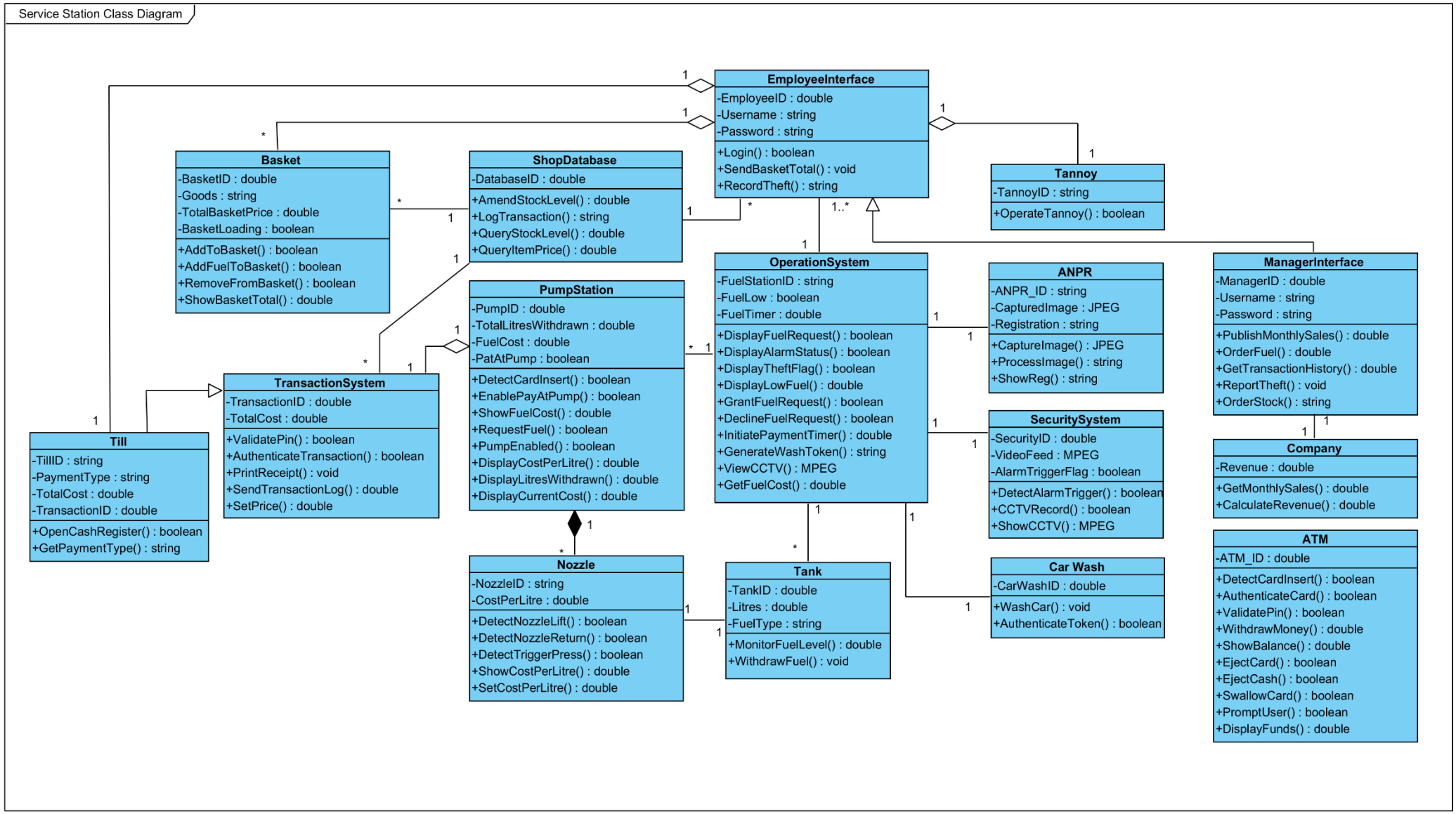


Figure 3

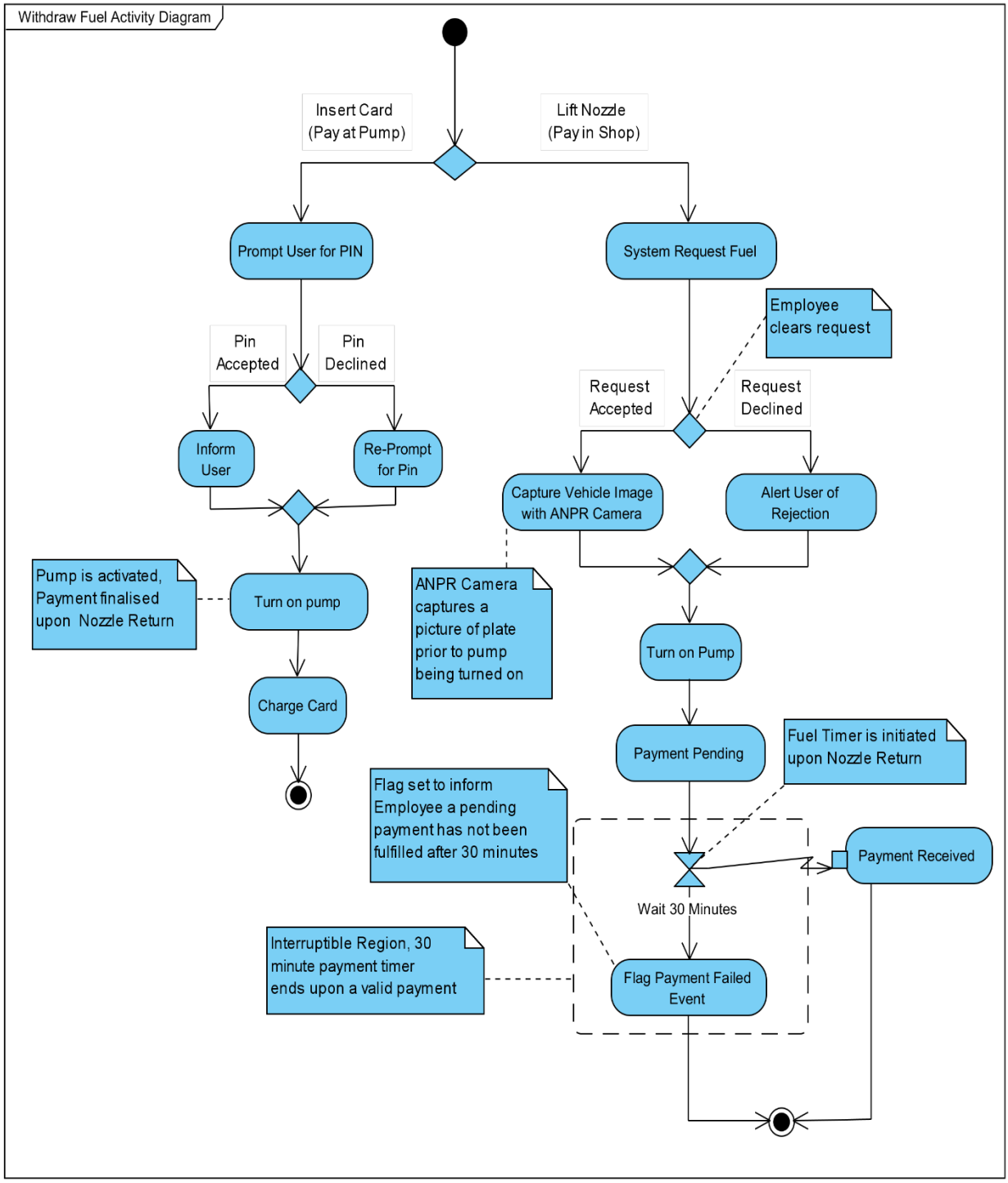

Withdraw Fuel Activity Diagram
Insert Card (Pay at Pump)
Lift Nozzle (Pay in Shop)
Prompt User for PIN
System Request Fuel
Employee clears request
Pin Accepted
Pin Declined
Request Accepted
Request Declined
Inform User
Re-Prompt for Pin
Capture Vehicle Image with ANPR Camera
Alert User of Rejection
Pump is activated, Payment finalised upon Nozzle Return
Turn on pump
ANPR Camera captures a picture of plate prior to pump being turned on
Turn on Pump
Charge Card
Payment Pending
Fuel Timer is initiated upon Nozzle Return
Flag set to inform Employee a pending payment has not been fulfilled after 30 minutes
Payment Received
Wait 30 Minutes
Interruptible Region, 30 minute payment timer ends upon a valid payment
Flag Payment Failed Event

Figure 4

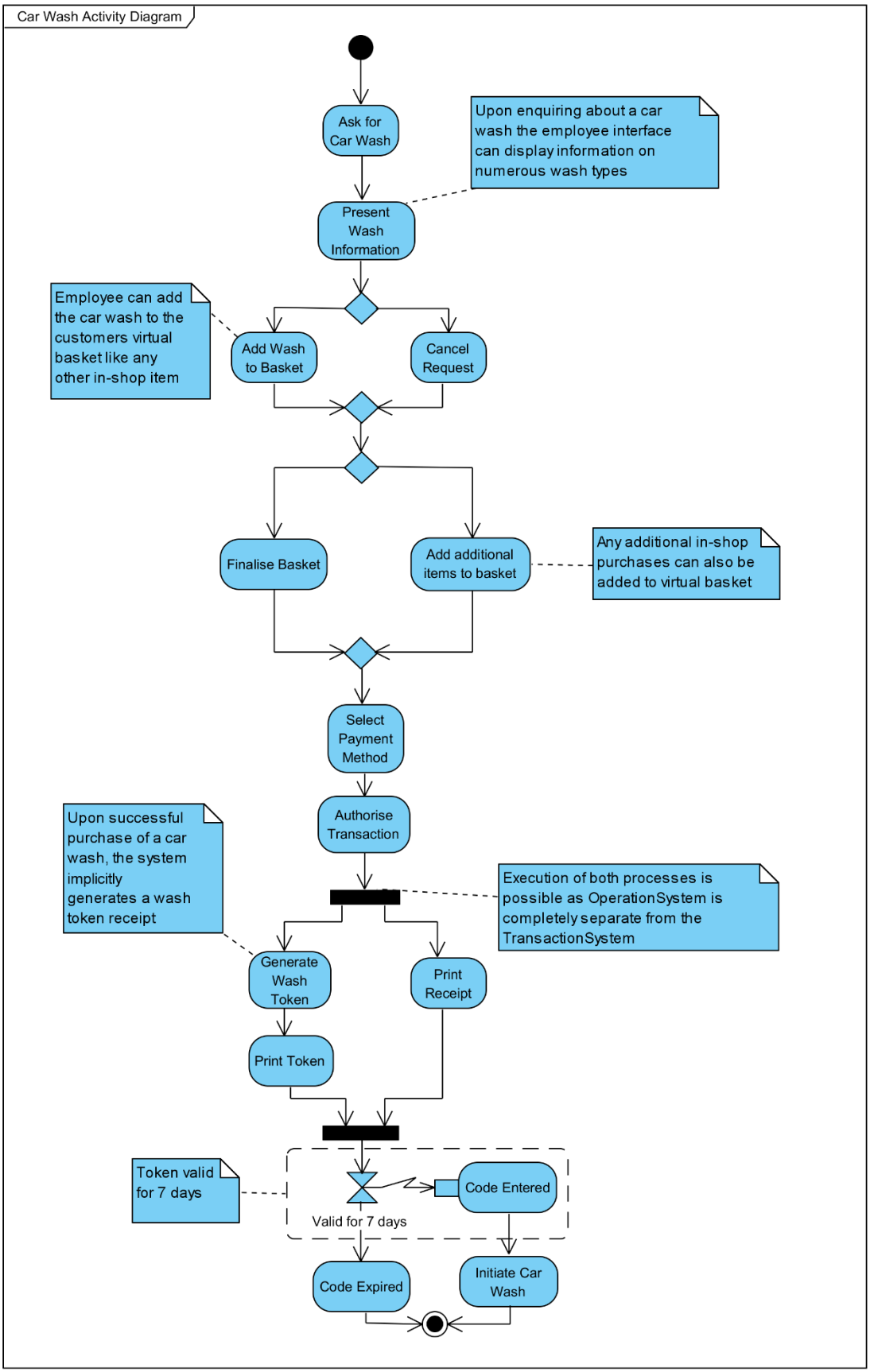
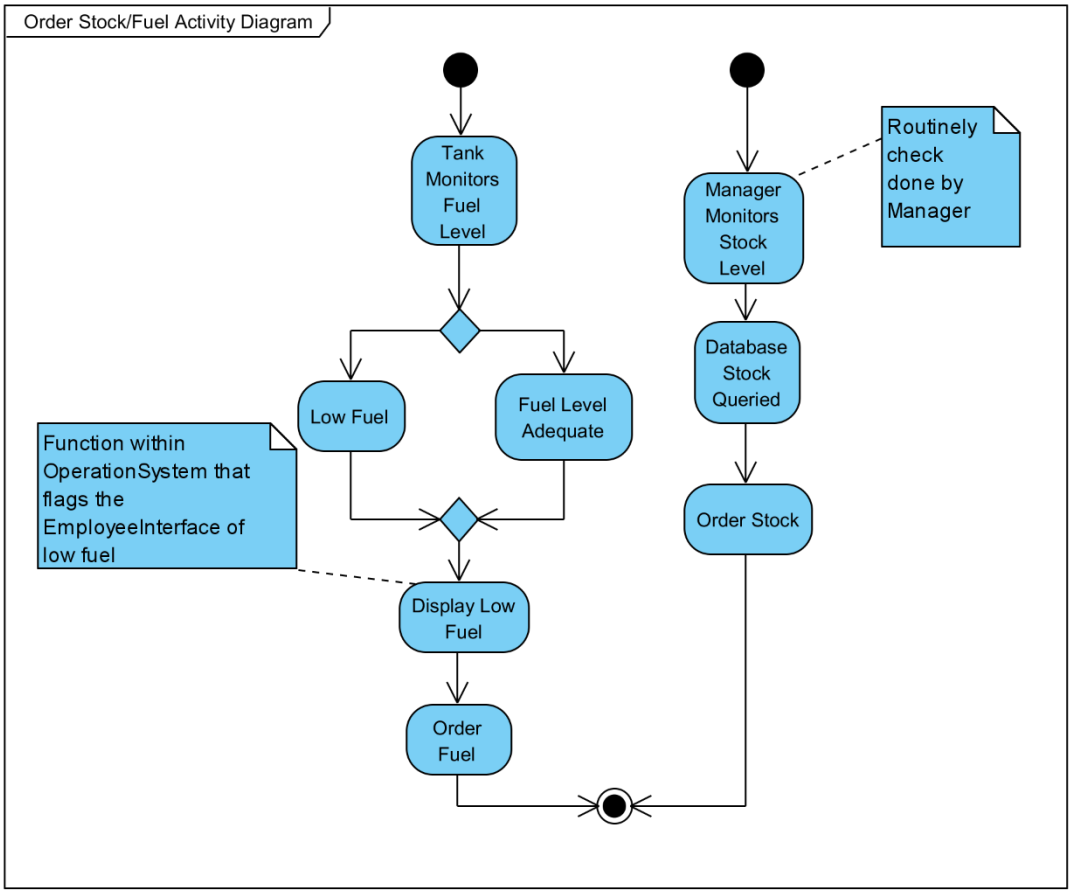

Car Wash Activity Diagram
Ask for Car Wash
Upon enquiring about a car wash the employee interface can display information on numerous wash types
Present Wash Information
Employee can add the car wash to the customers virtual basket like any other in-shop item
Add Wash to Basket
Cancel Request
Finalise Basket
Add additional items to basket
Any additional in-shop purchases can also be added to virtual basket
Select Payment Method
Authorise Transaction
Upon successful purchase of a car wash, the system implicitly generates a wash token receipt
Execution of both processes is possible as OperationSystem is completely separate from the TransactionSystem
Generate Wash Token
Print Receipt
Print Token
Token valid for 7 days
Code Entered
Valid for 7 days
Code Expired
Initiate Car Wash

Figure 5


Order Stock/Fuel Activity Diagram
Tank Monitors Fuel Level
Low Fuel
Fuel Level Adequate
Function within OperationSystem that flags the EmployeeInterface of low fuel
Display Low Fuel
Order Fuel
Manager Monitors Stock Level
Routinely check done by Manager
Database Stock Queried
Order Stock

Liam O'Neil - B513647, Mariano Etchart – B528854

Figure 6

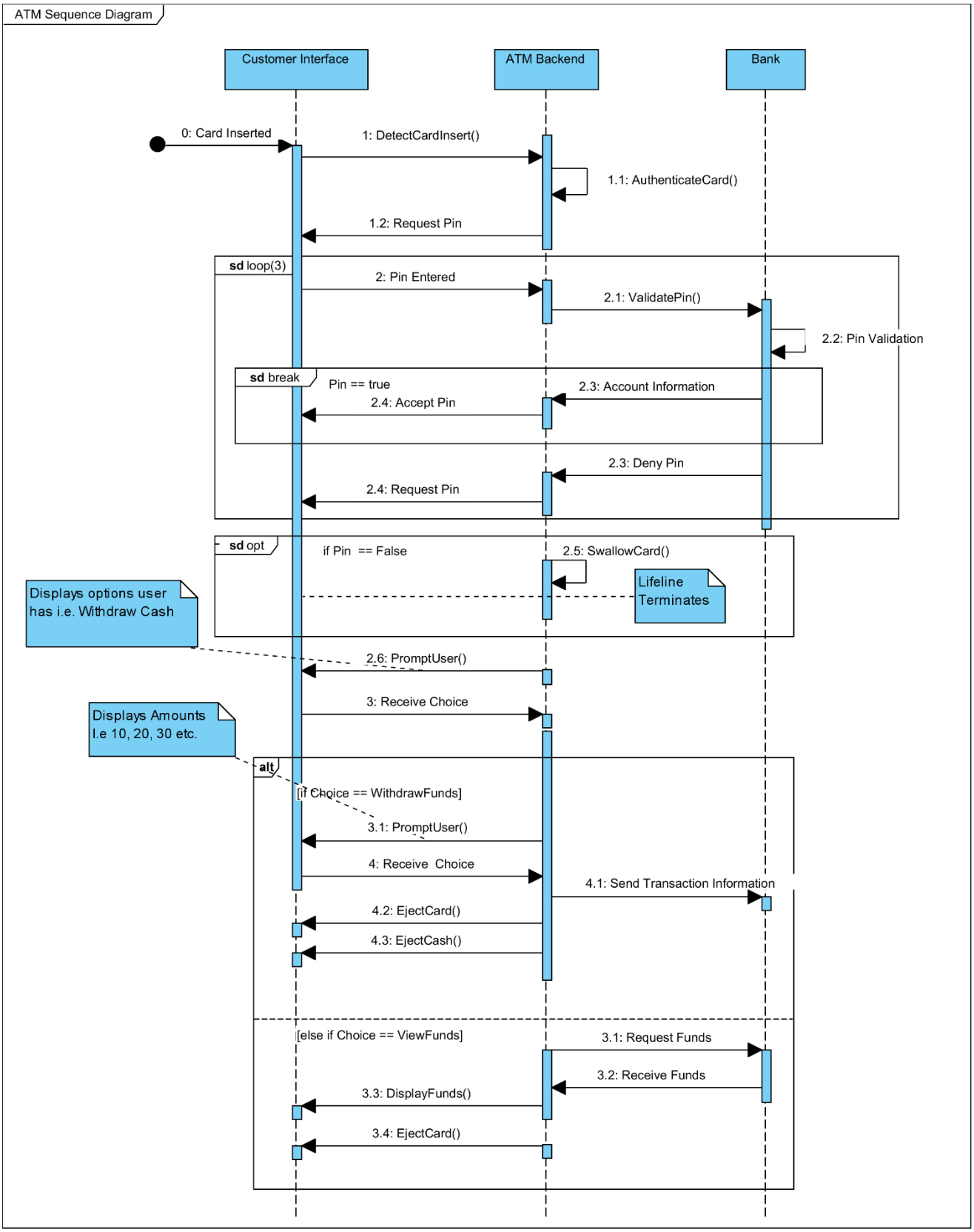
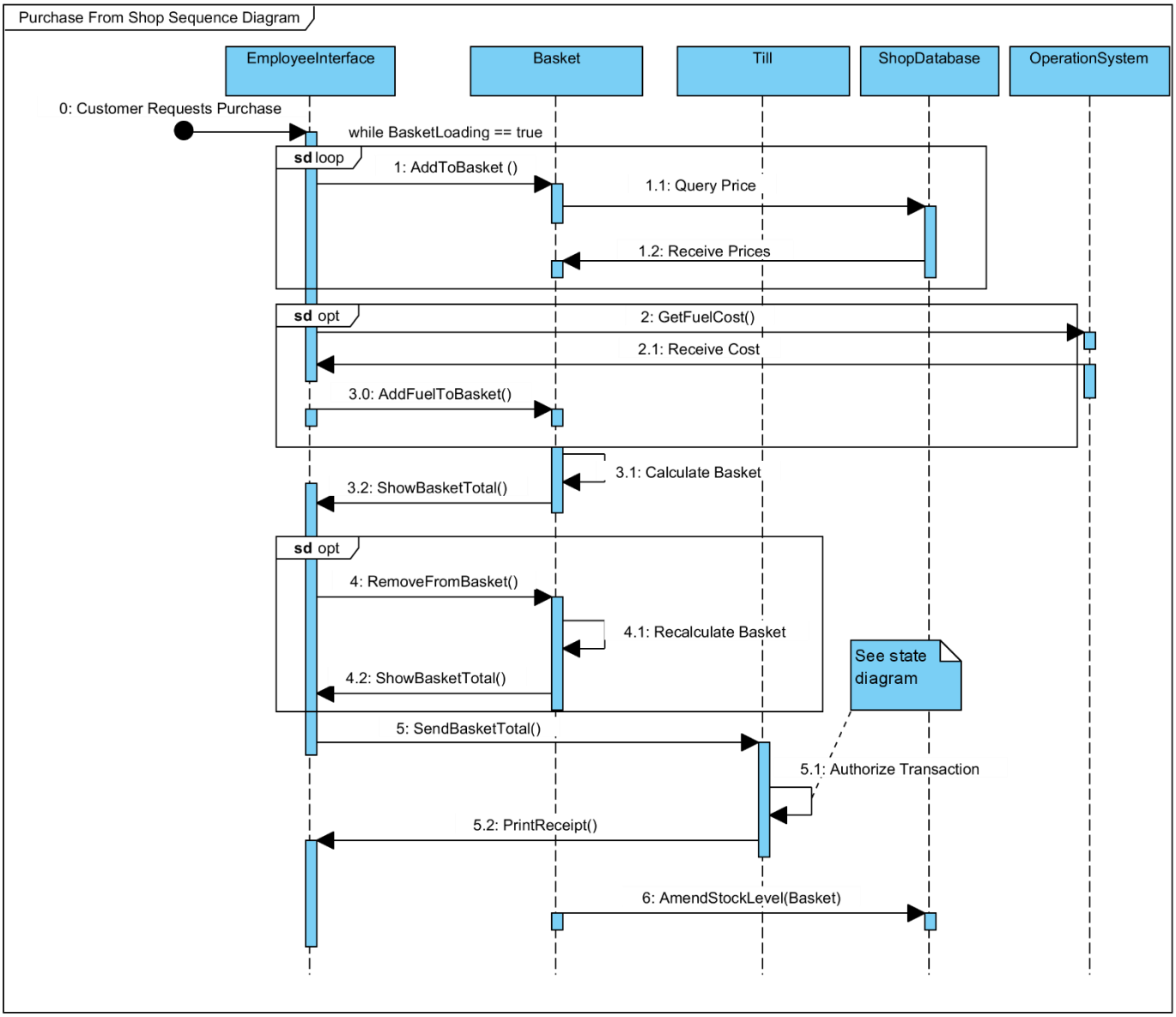


Figure 7


Purchase From Shop Sequence Diagram
EmployeeInterface
Basket
Till
ShopDatabase
OperationSystem
0: Customer Requests Purchase
while BasketLoading == true
sd loop
1: AddToBasket ()
1.1: Query Price
1.2: Receive Prices
sd opt
2: GetFuelCost()
2.1: Receive Cost
3.0: AddFuelToBasket()
3.1: Calculate Basket
3.2: ShowBasketTotal()
sd opt
4: RemoveFromBasket()
4.1: Recalculate Basket
4.2: ShowBasketTotal()
See state diagram
5: SendBasketTotal()
5.1: Authorize Transaction
5.2: PrintReceipt()
6: AmendStockLevel(Basket)

Liam O'Neil - B513647, Mariano Etchart – B528854

Figure 8

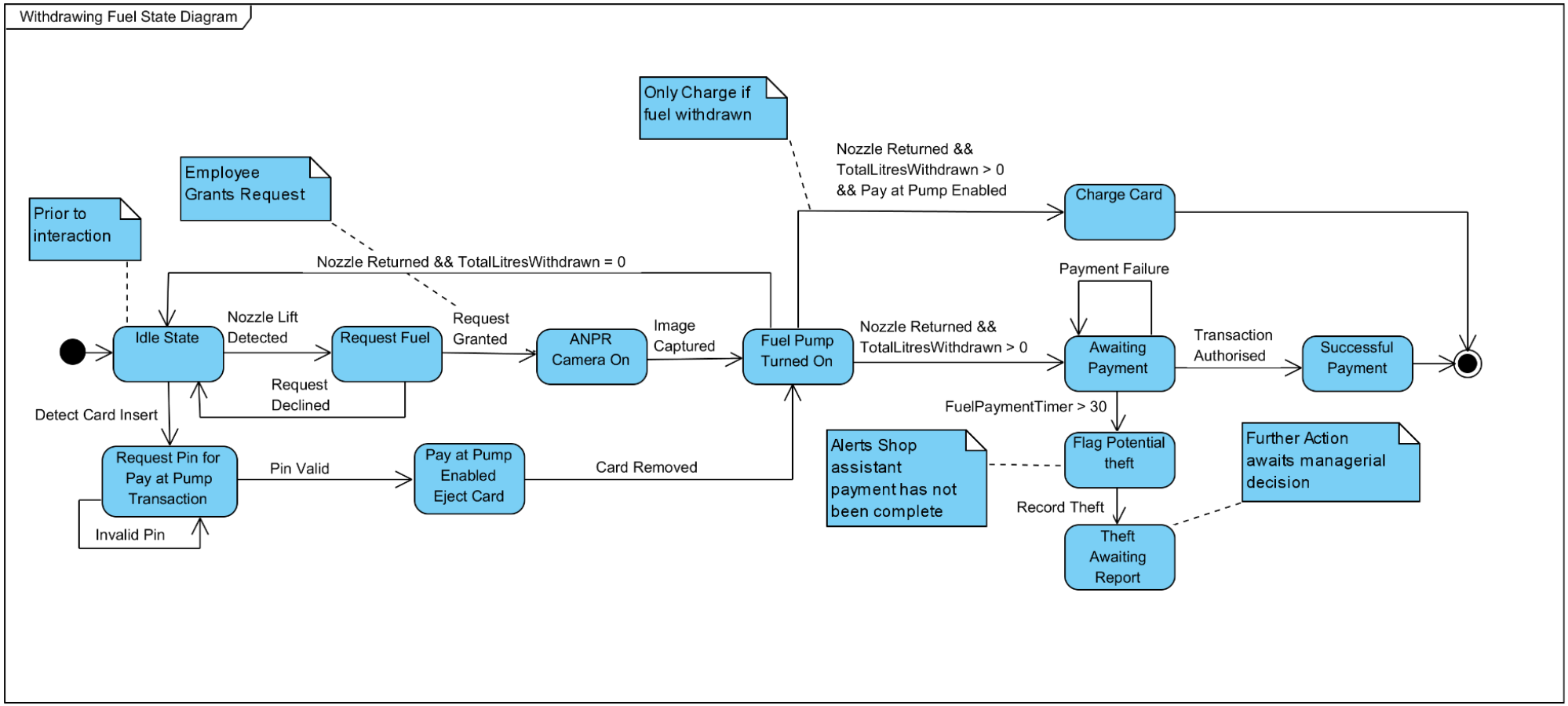
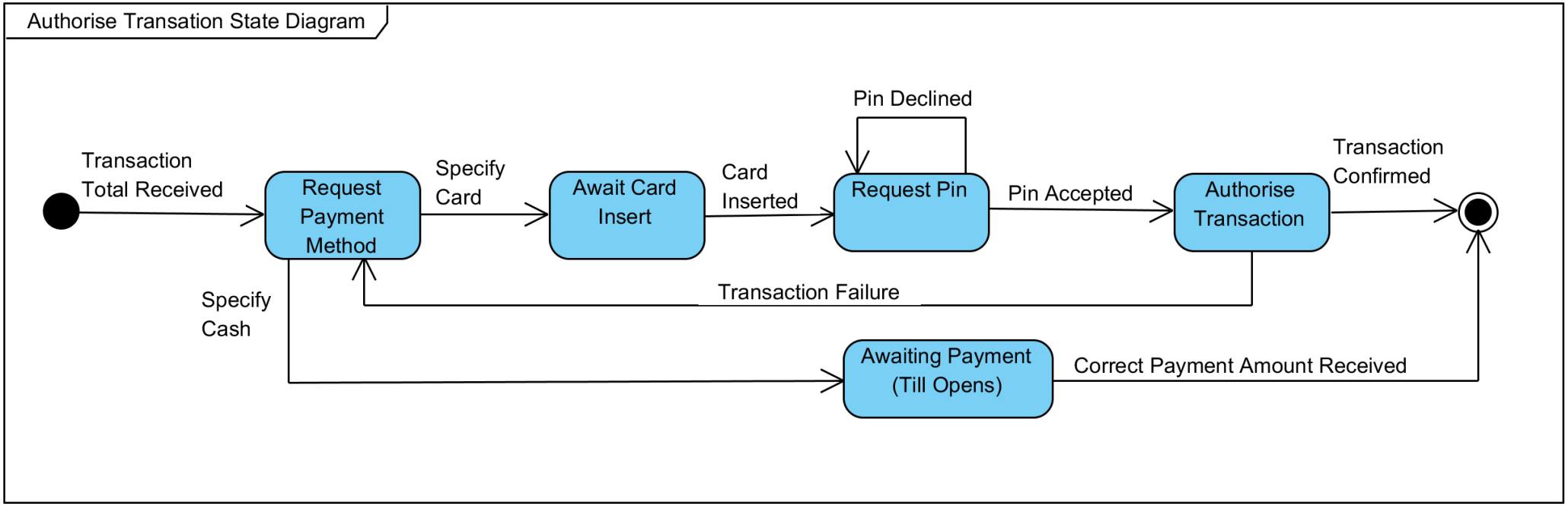


Figure 9


Authorise Transation State Diagram
Transaction Total Received
Request Payment Method
Specify Card
Await Card Insert
Card Inserted
Request Pin
Pin Declined
Pin Accepted
Authorise Transaction
Transaction Confirmed
Transaction Failure
Specify Cash
Awaiting Payment (Till Opens)
Correct Payment Amount Received

Liam O'Neil - B513647, Mariano Etchart – B528854

Figure 10