**Rationale:**

The chosen architectural style I am going to implement is an independent component event based system using implicit invocation. The justification for the chosen style is because one component can continue to make progress independent of the states of other components. The system is to be run on a network of computer systems consisting of multiple consoles, pumps and a single server.

The way in which this architectural style works is that individual components announce data that they wish to share with their environment as a set of unnamed components and can be invoked into a class of data. The structure makes use of event handlers to manage the communication amongst components.

Using the independent component event based system there are a number of quality aspects that need to be taken into consideration. There are two main factors that need to be analysed:

* Modifiability, especially when performance tuning via reallocating work among processes and processes to computers is necessary.
* Performance, especially to achieve maximal utilisation of the computational resource is important.

The interface of a component in this style usually contains multiple events that can assess and activate the use of procedures and functions that other components are dependent upon. This enforces the quality aspects that need to be taken into consideration above.

**Component and connector description:**

1. Master Control: This module will take the conditional inputs from the pump and the console based on hardware assumptions that the pump details are entered into the system upon the nozzle being lifted. A further assumption can be made from the console that upon a key press a confirmation can be made when the payment method has been called.
2. Input:

2.1 Pump ID: The identity of the pump is assigned when the customer arrives at the pump chosen and once one of the pump station nozzles has been lifted.

2.2 Fuel Type: Is an assigned value to each nozzle on the pump station and will reflect on the Unit Price later in the system.

2.3 Payment details (Pump Payment): This is a set of details obtained from the card reader and will extract the card number and account information etc.

1. Pump Start:

This is a hardware component that causes an event to occur. This event drives the Pump ID and Fuel Type to create a transaction record which updates the cost dependant on the Volume and Unit price selected at the pump.

1. Transaction: This is an active software component that will perform the following functions

4.1 Volume: The amount of fuel being taken at the pump will be constantly assessed and stored for calculation by this class.

4.2 Unit Price: This is a variable storing the data that is pulled and which is dependent upon the Fuel Type selected on the nozzle uplift.

4.3 Cost: The cost of the transaction is determined by the Volume and Unit Price multiplied together. The transaction class performs this calculation and pulls

1. Pump Finished: This is a hardware component that locks the pump from further use by any other customers. The component then sends a status reflections to the Pump Status method based on where payment details have been submitted or whether the nozzle has been returned.
2. Pump Status: The component processes all the Pump Definition data, Transaction data, Status of the pump and submits the record to the database. If no payment has been made a function within the pump status will output a message to the console alerting staff that the status of a single pump has not been completed due to no payment being made.
3. Database: The database will include a single entry for each transaction made through the system and store them for retrievable at the interface element of the console.

1. Output: The output will collate the relevant information from the database and output the request at the master control.

**Structure:**

DB

Transaction

Cost

Volume

Payment details

Unit price

Pump ID

Fuel type

Pump Start

Pump Finished

Pump Payment

Pump Definition

Input

Pump Status

Output

Master Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | | | Change | |
| No. | Description | Type | Component | Change |
| 1 | User wants to add the facility to charge electric vehicles. | Indirect | Input, Pump Definition, Transaction | A new input type will be required which therefore changes the pump definition and fuel type desired. This in turn requires a modification to the transaction where a different unit price and volume measurement will need to be added. |
|
|
|
| 2 | User wants to add a new pump from a different manufacturer. | Indirect | Master Control, Input, Pump Definition | The master control works as the user interface and the main program. This also requires modifications in the input and pump definition components to accommodate for compatible functionality. |
|
|
|
| 3 | User wants a new interface for the cashier with touch screen | Indirect | Master Control, Output | The master control and output will require modification as they directly affect the interface of the cashier. |
|
|
|
| 4 | User wants to accept payment via an NFC device | Indirect | Input, Pump Payment, DB | The input component will need to be changed to accommodate for the interface change on the pump. Secondly, the pump payment component contains all the payment details and manipulates them into a format acceptable for the database. In this case details may be stored in a different format or more details may be required. |
|
|
|
| 5 | User wants to add pay at pump facility | Indirect | Input, Pump Definition, Pump Payment, Transaction, DB | The pay at pump facility requires a different input type to be assigned to the pump, this requires modifications to be made to the pump definition and pump payment components to accommodate for a merger of the data contained within those components. This will also impact the transaction, DB and the flow of data from input to transaction. |
|
|
|

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | User wants to add an additional grade of fuel | Indirect | Input, Pump Definition, Transaction | A new input type will be required which therefore changes the pump definition and fuel type desired. This in turn requires a modification to the transaction where a different unit price and cost calculation will be performed. |
|
|
|
| 7 | User wants to add another pump. | Direct | No components affected | Not application |
|
|
|
| 8 | User want to add another cashier console | Direct | No components affected | Not application |
|
|
|
| 9 | User want to add the ability to combine multiple pump usages into a single transaction | Indirect | Transaction, Pump Payment, Pump Status | The transaction component will no longer be dependent upon a pump start to define a pump status. The component will require additional data to be stored for processing the pump status and defining a payment for the transaction. The pump status and pump payment components will need to accommodate for this change and require further connections to be added. |
|
|
|