

## CS4125: System Analysis and Design Project

Table of Contents

[CS4125: System Analysis and Design Project 1](#_Toc448992206)

[Narrative 8](#_Toc448992207)

[Introduction 8](#_Toc448992208)

[SOFTWARE LIFE CIRCLE (SLC) 10](#_Toc448992209)

[Use case Detail Description 12](#_Toc448992210)

[Use Case Diagram 13](#_Toc448992211)

[Use Cases 16](#_Toc448992212)

[Project Plan and Allocation 25](#_Toc448992213)

[System Architecture 26](#_Toc448992214)

[Architecture Package Diagram 26](#_Toc448992215)

[System Architecture Description 27](#_Toc448992216)

[Programing Language Implementation. 27](#_Toc448992217)

[UML Workbench 27](#_Toc448992218)

[System Architecture Hierarchy. 28](#_Toc448992219)

[Reorder Sequence Diagram 29](#_Toc448992220)

[Login Sequence Diagram 30](#_Toc448992221)

[Analysis Design 31](#_Toc448992223)

[Class Analysis Diagram 34](#_Toc448992224)

[Entity Relationship Diagram 40](#_Toc448992228)

[State Diagram 41](#_Toc448992229)

[Descriptions of the pattern used in item 42](#_Toc448992230)

[Critique 42](#_Toc448992231)

[Evaluation of the Analysis and Design Artefacts 42](#_Toc448992232)

# 

# Narrative

Introduction

The European Vending Association estimates that every day 82 million food and drink items are purchased from a vending machine, by European customers (including Russia, Turkey, and Ukraine).

In 2012 the field of operating machines has remained almost stable with 3.77 million operating machines. 80% of these machines where located in Europe’s 6 big leading markets the big 6 (Italy, France, Germany, the Netherlands, Spain and the United Kingdom.).

An Interesting fact according to the European Vending Association is that 60% of these operating machines are hot drinks vending machines whereas the remaining 40% can be evenly divided between cold drinks and a mix of both hot drinks and cold drinks vending machines.

The total turnover from vending machines in 2012 was €11.3 billion.

The total number of items sold through vending machines in 2012 was 30 billion units, of which 24 billion items were dispensed in the Big Six. These vending machine are mostly operated manually, there has not been much automation implementation.

**EasyVending**

We are proposing a system hereafter "EasyVending” to Vending Machine Manufacturers (VMM's), that promises to understand "easy interfacing" and making it the core and heart of every vending machine. EasyVending provides a solution that has the Interest of potential Users, Beneficiaries and Manufacturers in mind by automating vending machines to be less dependent of external interference during potential errors. With the advent of increasing technology and embedded systems growing with computational capabilities, "EasyVending" will make use of these advances and interface with the vending machines main business operations allowing Beneficiaries to manage their machines without prior programming or technological background.

VMM's who are using "EasyVending" promise their customers full control and maintainability. Most vending machines that are currently operating have limited interfacing options which makes the customer often depend on repair services usually conducted by individuals not associated with the customer’s main business that produces extra cost which in most cases can be avoided. Additionally EasyVending provide customers access to their vending machines through interfacing locally and/or remotely, allowing them to add or remove products, change prices, adjust currency settings, generate error reports, manage temperature settings, keep records of temperatures within the machines which has become a key factor for customers that use their vending machines to sell hot beverages, snacks or drinks .

During the initial install and setup, EasyVending provides VMM's an opportunity to set up remote monitoring and remote access to the Vending Machine; which gives VMM's the option to identify and track errors early on and can act accordingly.

EasyVending also allows Beneficiaries of Vending Machines to gain access to local and/or remote interfaces where business operations such as Product mapping, pricing, and temperature can be configured through an intuitive easy to use interface.

EasyVending however stands out from conventional VendingMachine software solutions in so far that, apart from its easy interfacing, it provides a new approach towards VendingMachine automation trying to minimize external interferences and maximising the VendingMachine ability to recognize and resolve potential issues internally.

This is where AlertControl comes in.

AlertControl consist of two bounded interfaces North and South.

The "Northbound"

Interface of Alert Control is concerned to keep an eye on the internal stability of the system, i.e. make sure that temperature sensors don't report values that exceed the required ranges, the current financial status corresponds with the current product sales, no error reporting within the Vending machines Operating System and possible vandalism.

The "Southbound"

Interface of Alert control is concerned with collecting, recording and according to severity sending reports out the Administrators and Managers.

We estimate that most of EasyVending's revenue stream will come from VMM's who are using EasyVending as their solution, in the form of sell commission and/or a one-time payment for EasyVending.

# SOFTWARE LIFE CIRCLE (SLC)

Software Life Circle or Software Development Life Circle (SDLC) is a framework defining task performed for each step of a software development process. Using a software model helps in the development of a software product in a systematic and structured manner. There are various software life circle models standard which exist today. We consider some of these model approach in relation to our project. They are:

Waterfall Model Iterative Model Spiral Model

V-Model RAD Model Agile Model

|  |  |  |
| --- | --- | --- |
| **SLC Model Types** | **Advantages** | **Disadvantages** |
| Waterfall Model | -Better with smaller project  -Low cost involvement  -Simple to implement and it is well known  -Good for in-experience developer or staff | -Poor for complex and  big project  -Inflexible  -Difficult to integrate risk management and maintenance |
| Iterative Model | -On time feedbacks  -Easy refractory when error occurs | -Rigid phases  -Costly |
| V- Model | -Good bases for partitioning of testing  - Each phases has specific deliverables | -Little flexibility and adjusting scope is difficult.  -Very expensive to maintain  -No clear path for problems found during testing phases |
| Agile Model | -Adaptable  -Working software is delivered frequently  -Close, daily cooperation between customer/User and developers | -Minimal emphasis on designing documentation |
| Spiral Model | -Risk Assessment and reduction  -Good for large and mission critical project  -On-time software production | -Very expensive to use  -Requires expertise risk analysis  -Is not good for smaller projects |

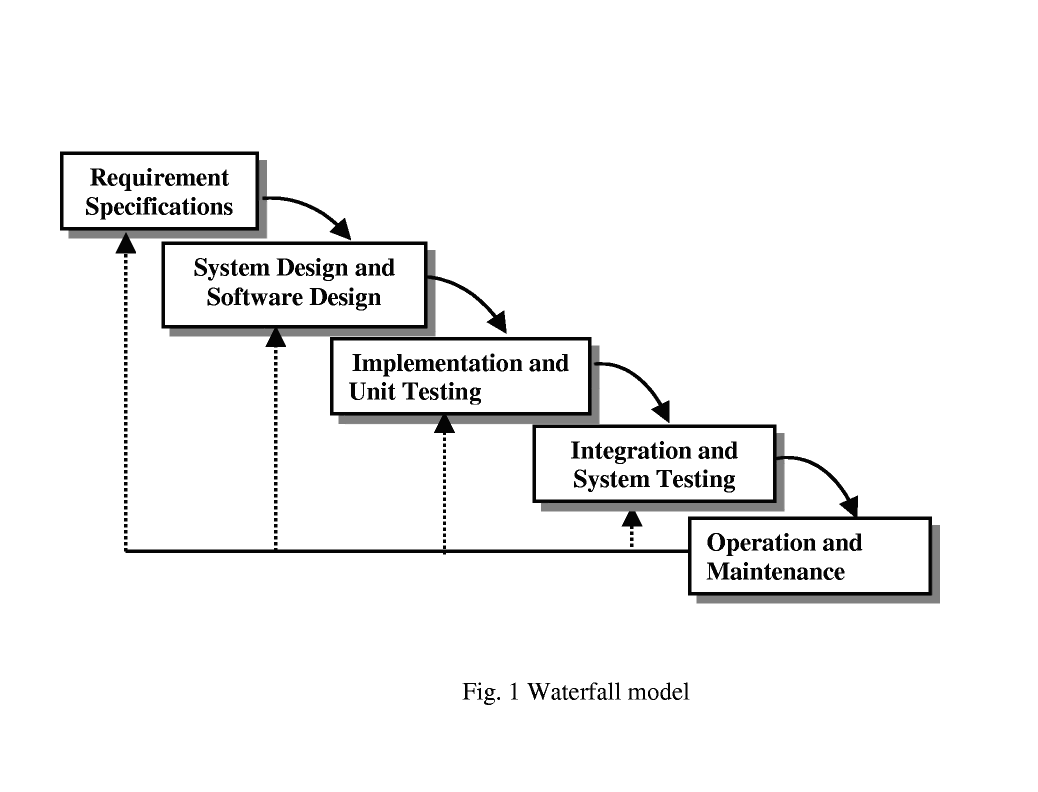
Considering the pros and cons of these model and based on the nature of our project EasyVending, we’ll be using the Agile Model SLC because of the flexibility and implementation using scrum approaches in completing the project requirement.

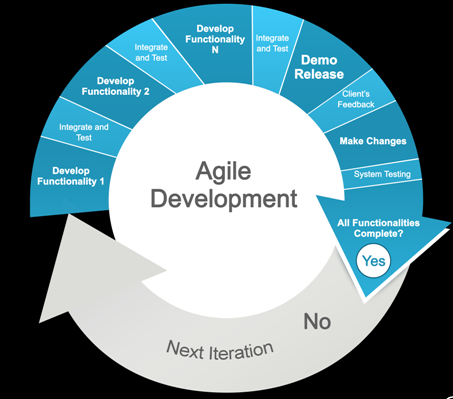
Waterford model will not suit our project because the project is prone to change frequently as we progress in the development.

In Iterative model, each phase rigidity will not work with EasyVending because each phases of our project may require developmental changes.

V-model will not be suitable because of the difficulty in scope adjustment. The dynamic modern development requires a more scope adjustment based on the frequent change in stakeholders requirements.

Spiral model is not chosen because of the expert risk analysis and the taste for bigger project which does not suit our project.

The flexibility and adaptability of agile approach will suit our mode of operation because there may be few adjustment we’ll be implementing as we build the project (EasyVending).

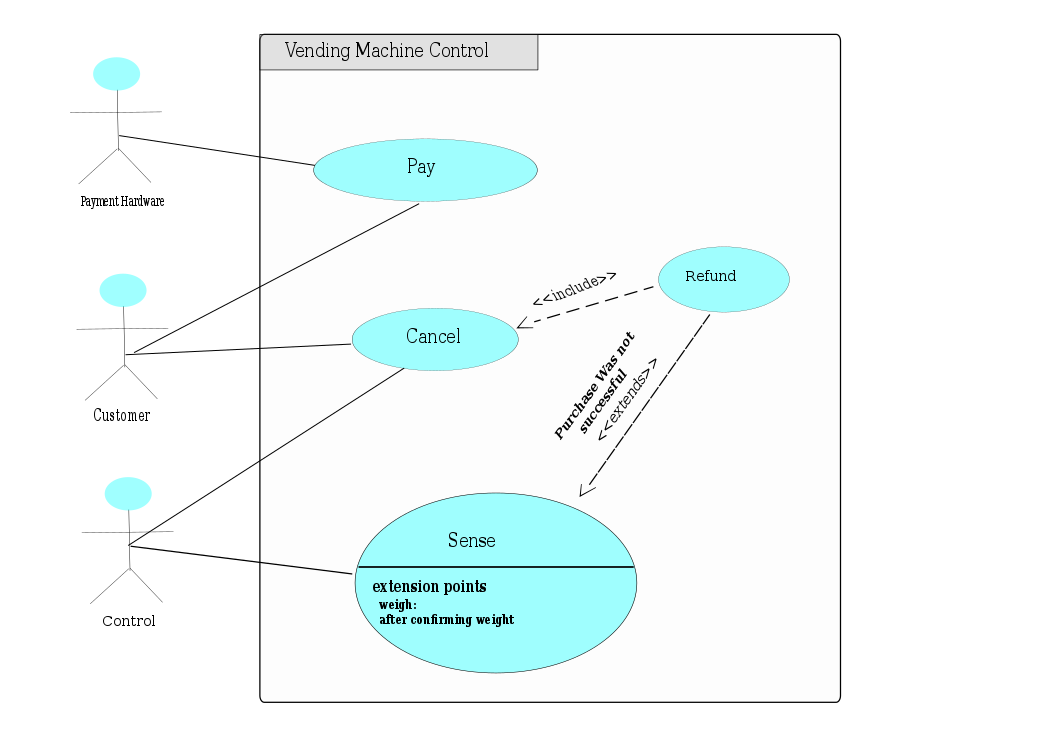
         

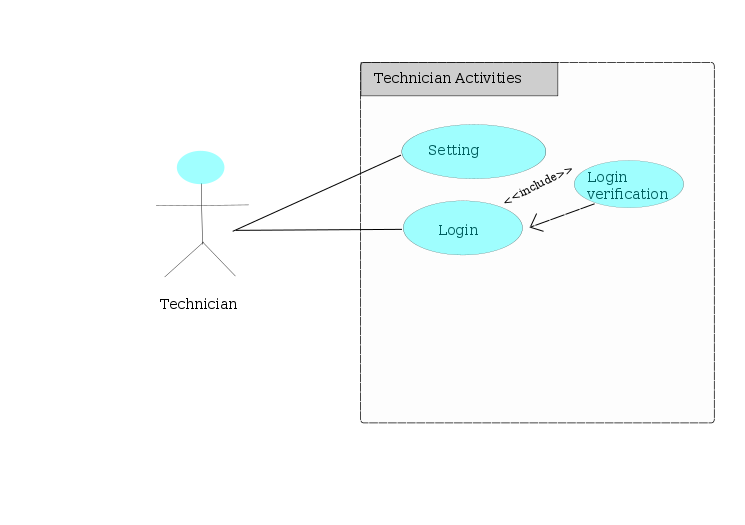
Adapted from (<http://www.agsgov.com/what-we-do/software-development/agile-development-lifecycle-2/> )

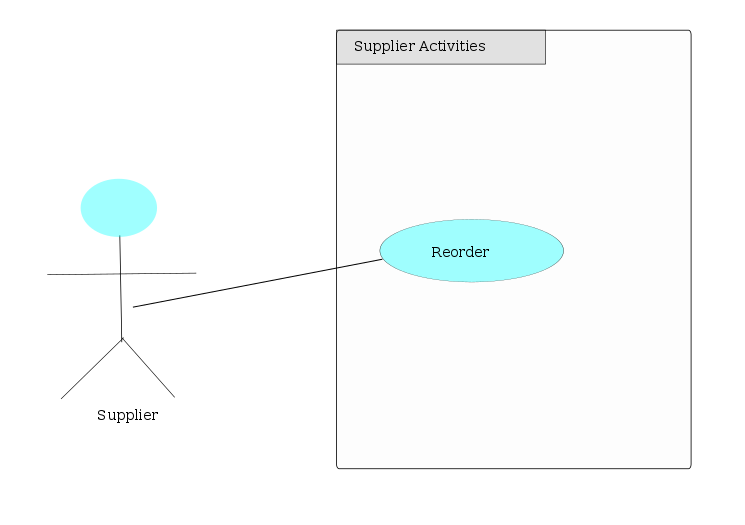
### Use case Detail Description

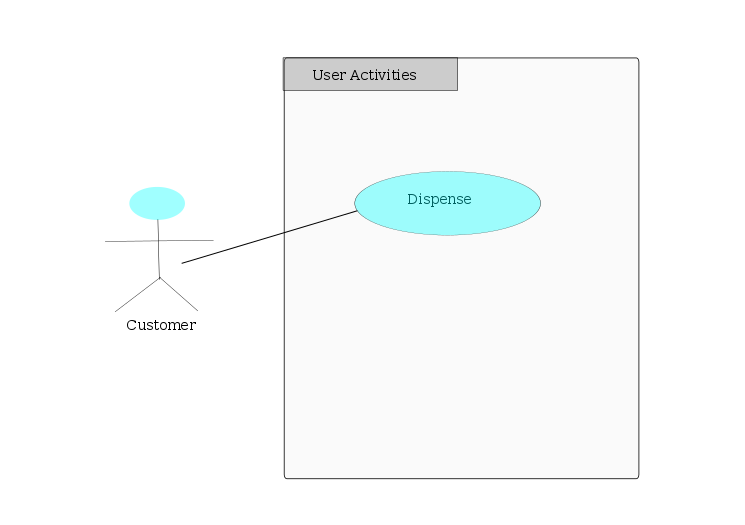
|  |  |  |
| --- | --- | --- |
| **USE CASE 2** | Business Setup | |
| **Goal in Context** | The beneficiary chooses products and goods to put in the machine | |
| **Scope & Level** | Delivery & Beneficiary Company | |
| **Preconditions** | Login confirmation | |
| **Success End Conditions** | Everything is validate and put in the machine ( Price + slots allocation) | |
| **Failed End conditions** | Input not accepted (wrong quantity, wrong format for price…) | |
| **Trigger** | Input request comes in | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | Beneficiary choose product Mapping from management interface |
|  | 2 | He enters product name into corresponding input box and click OK button |
|  | 3 | Beneficiary select a product name from a dropdown list |
|  | 4 | He selects a desired slot location for a dropdown menu |
|  | 5 | He selects a desired price from dropdown menu |
|  | 6 | He selects a desired temperature range or status from the current product dropdown menu |
|  | 7 | He selects a quantity for product from a dropdown menu |
|  | 8 | Finally he presses save on management |
| **EXTENSIONS** | **Step** | **Branching action** |
|  | 1a | Beneficiary enters wrong format for price |
|  | 7a | Beneficiary enters wrong amount of product |
| **VARIATIONS** |  | **Branching action** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Use Case Diagram









# Use Cases

**Machine Installation**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Installs Vending Machine  2) Enters (default Password, default Username, location, ID)  5) Select change login from Main Menu  6 ) Enters username in provided Input Box  8) Enters password in provided Input Box  9) Presses save  12) Enable remote access and monitoring for beneficiary | 10) Validates and accepts credentials  11) send credentials to remote servers  13) System operates and is accessibles locally and / or remotely by technician |
| **Alternative route** | |
|  | 10a) Credential submission not accepted Network error prevents sending new credentials to remote servers |

**Non-Functional Requirements**

Security

Login credentials should be encrypted when send over network

**Vending Machine - Business setups**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Chooses Product Mapping from Main Menu  2) Enters product name into corresponding input box and clicks OK button  5) Selects a product name from dropdown list  6) Selects a desired slot location from dropdown Menu  8) Selects desired Price from Dropdown Menu \*\*1 Selects a desired temperature range for the current product \*\*2 Selects N/A from temperature dropdown to indicate no temperature needed  10) Selects a Quantity for product from a dropdown Menu  12) Presses save on Management Interface | 7) Validates an accepts slot allocation  9) Validates and accepts price input  11) accepts temperature range  13) Accepts and stores created configurations |
| **Alternative route** | |
| Network error prevents sending credentials to remote servers | 2a) Credential submission not accepted. |

**Non Functional Requirements**

Performance

The system should process given configuration in not more than 5 seconds

Usability

The machine provider must inform possible suppliers about changes made in the Vending machine business setup

**Reorder**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Open machine’s door  2) Places goods into machine  3) Locks the door | 4) System response to suppliers locking the door  5) System calculate individual weight for products in allocated slots -System initialises AlertControl according to configuration -System calculates an estimate of overall product value -System calculates offset against current available money in the Machine |
| **Alternative route** | |
| System fails to estimate overall estimate of current money in the system | 2a) Supplier places products in the wrong slots. |

**Non-Functional Requirements**

Extensibility

The internal process of allocating and specifying product should be modifiable by the beneficiary

Performance

The system should update the current and financial state in not less than 30 seconds

**Purchase Vending Machine**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Request vending machine type to a specific location from Machine provider  3) Pay for the vending machine | 2) Location of installation is define and mapped out  4) Machine installation is carried out and machine details (Type, location, ID, Customer) is logged on the Machine provider database server |
| **Alternative route** | |
| System refuses to log and confirm details | 4a) Customer machine installation is faulty |

**Non-Functional Requirement**

Availability

Vending machine should be made accessible and operational.

**Purchase items**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Select a preferred item by pressing a button  3) Insert cash or card for payment  6) Get or pick up item from the base of the base of the machine | 2) Vending machine reads and calculate the prefer item details (Type, slot-line, numbers, amount)  4)Vending machine display amount due and receive the proportionate cash value  5) Vending machine dispense or drop the item selected for user to pick it up |
| **Alternative route** | |
| Vending machine refuse to dispense item    Vending machine refuse to dispense item and refund cash. | 4a) User select item and did not insert cash or payment  4b) User select wrong item and insert cash (lower than the item value) |

**Non Functional Requirements**

Performance

The response from Vending machine is faster and correct

**Cash evaluation**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Read and confirm the cash/card properties that was inserted  3) Compare the cash/card amount to the calculated value of the selected items. | 2) Detect the genuineness of the currency or card properties.  4) Accept the cash/card and send data to confirm (in the case of credit/debit/visa card) |
| **Alternative route** | |
| System cannot dispensed items. | 2a) Cash/card cannot be verify |

**Non-Functional Requirement**

Performance

System process of validation and verification of inserted cash/card is quick

**Reads temperature**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Sense the ambient environ for change in temperatures proportionate to the value set, for the environs. | 2) Adjust the degree of coldness relatively |
| **Alternative route** | |
| Temperature function refuse to work | 2a) Temperature sensor not reading correctly the set-up temperature environ |

**Non Functional Requirements**

Reliability

Vending machine send alert to Machine provider and reset itself

**Alert Control - Vandalise**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Handles machine with aggressive force and malicious intent | 2) Internal Acelerometers report data higher than the specify threshold  -System sends alert and report corresponding logs to remote servers -System sends an alert email to the Beneficiary. |
| **Alternative route** | |
| System not able to detect possible vandalism  System interprets a normal purchase as vandalism | 2a) Internal Accelerometers may be faulty |

**Non Functional Requirements**

Extensibility

The system should allow system wide recalibration of sensors

**Alert Control - Report Weight Data**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Reports data from sensors that does not correspond to overall estimation of current weight | 2) The system check if a purchase was made  -System check what was purchased -The system calculates weight imbalance against the individual weight of product that was purchase -The System determines if the weight balance corresponds to a possible product that was not dispensed due to error -The system validates should a refund be issued -The system reports weight imbalance and actions taken to remote server and logs |
| **Alternative route** | |
| System fails to determine if current weight imbalance was related to a foregoing purchase and cannot decide whether it should issue a refund. | 2a) System cannot read sensor |

**Non Functional Requirement**

Performance

The response to a weight imbalance and the decisive action to take should be performed in no more than 4 seconds.

**Login setup**

|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1) Enters username  2) Enters password | 3) Validate Credentials  4) Allow access to management interface |
| **Alternative route** | |
| Machine Provider supplies wrong username or password | 3a Vending Machine denies access to management interface. |

**Non Functional Requirements**

Performance

The management interface should load in a timely fashion within 1 to 3 seconds

The management interface should be easy to use and functions should be displayed in a disambiguated manner.

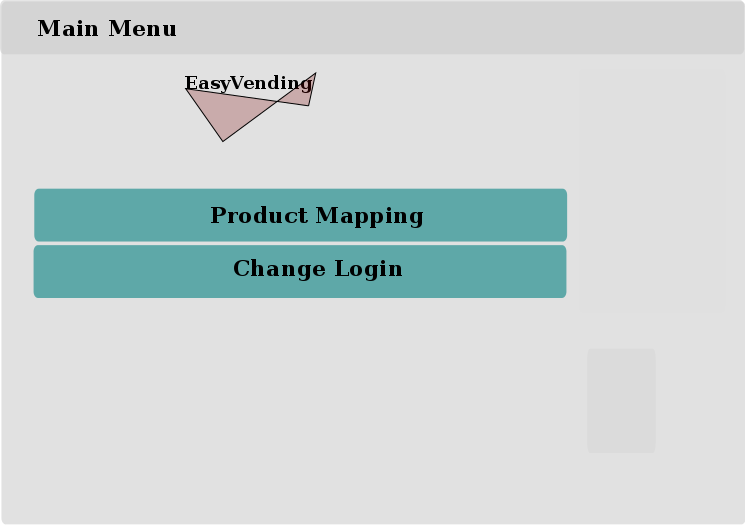
**Login**

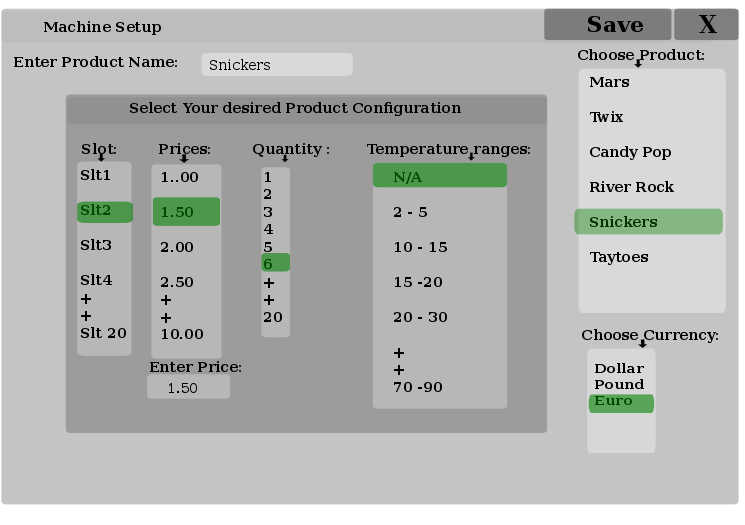
|  |  |
| --- | --- |
| **Actor’s Action** | **System Response** |
| 1)Select change login from management interface  2) Enters new username in provided input Box  3)Enters new password in provided input box  4) Presses save | 5) Validate new credentials  6) Stores new credentials |
| **Alternative route** | |
| Vending Machine does not accept input and alert beneficiary (customer) the error | 5a) Beneficiary (customer) supplies password not adhering |

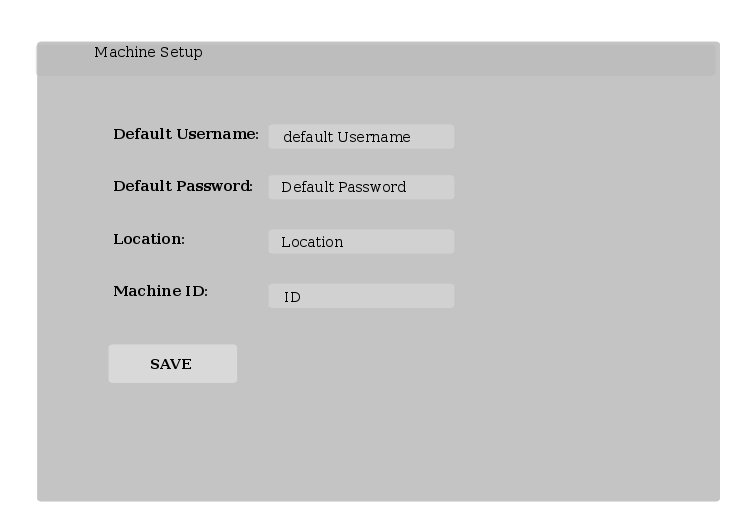
**Non Functional Requirements**

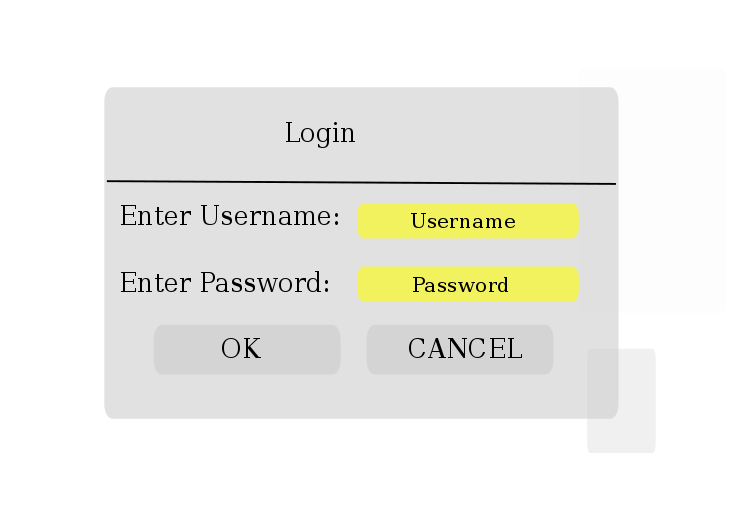
Security

New username and password must be stored in a save location and encrypted









# Project Plan and Allocation

Project Manager---> Benjamin Keil

Requirement Analyst---> Seun Adebowale

System architect---> Remi Ruppel

--->Padraic O’neil

Group → [Project management, Requirement analyst, System architect]

Developer and Tester → Group.

|  |  |  |  |
| --- | --- | --- | --- |
| Deliverables | Job Description | Job Allocation | Target Week |
| Presentation | General presentation  (Cover page, Table of Content, Company logo) | Group | 5 |
| Narrative | Narrative Description of business scenario | Project Manager | 5 |
| Software Life Cycle (SLC) | Discuss and justify SLC and risk management strategy | Requirement Analyst | 5 |
| Requirements | -Use case diagram(s)  -Structured use case description(s)  -Non-functional requirements/ attributes  -Screen shot/report form/GUI prototype | Group | 6  6  6  6 |
| System Architecture | -System architecture diagram with interface | Requirement Analyst | 6 |
| Analysis Sketches | -Method used to identify candidate classes  -Class diagrams  -Interaction diagrams/Communication diagrams  -Entity relationship diagrams | -Group  -System Architect  -Project Manager  -Requirement Analyst | 7  7  7  7 |
| Code | -Code implementations | Group | 8 -10 |
| Design Blueprints- Based on code | -Architectural diagrams  -Class diagrams  -State chart |  | 11  11  11 |
| Critique | Evaluate the analysis versus design artefact(Blueprints) |  | 12 |
| References | Sources of information/ learning |  | 4-12 |

# System Architecture

### Architecture Package Diagram

C:\Users\s\Dropbox\Apps\drawio\ArchDesESV.png

### System Architecture Description

Our unanimous decision for the system can be divided into three levels of subsystem operations

**The Database System**

**The Server System**

**The Client/User System**

**The Database System:** The database system will handle the information storage for the client system. The storage of the information will be done in locally and remotely on the server. It will be structure in a simple relational database that makes query information efficient.

**The Server system:** Provides access to the database information and make it available when there is query to confirm the user’s details from the Client side. It is also responsible for the live information monitoring from the client system.

**The Client System:** The Client system is the information query and retrieving interaction platform. Suppliers and Beneficiary (Customer) of the system can login to access operational privileges provided by the Vending Machine. It does not interact with the Database system directly but sends and receives queried data via the server system.

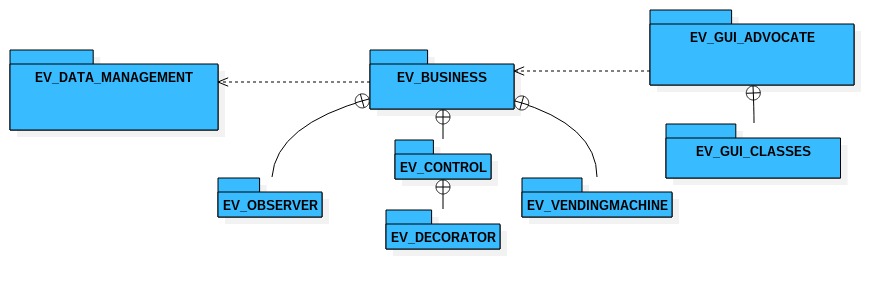
Programing Language Implementation.

Considering the system use and environment where it will be used; we agreed together to consider portability of the system function although performance is crucial as well. We can implement performance and the system adoption grows. We are using   ‘Java’ programing language due to its distribution properties (i:e It can easily be shared and utilise as component packages into various and different system  because of the java virtual machine accessing privileges. )

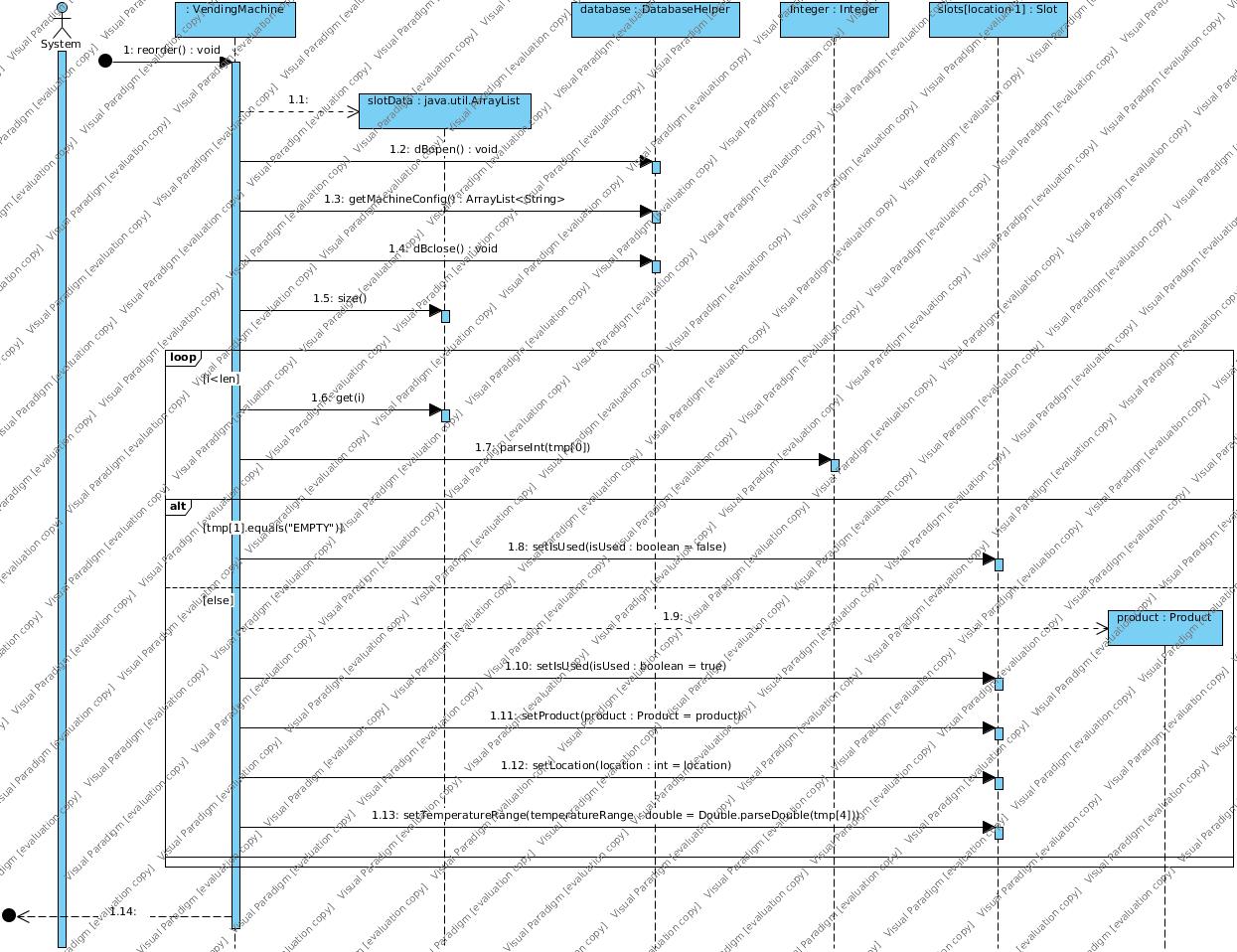
UML Workbench

There are various drawing tools available to design the UML structure and other diagrams that were presented in this project. Our group work from both linux and windows system; which makes it difficult to stick to one drawing application because of individual preferences. Some of the drawing application we used are InkScale for the Use cases diagrams, StarUML for the Class diagram, draw.io for Entity Relationship diagram and other diagrams that we created.

### System Architecture Hierarchy.



### Reorder Sequence Diagram



### Login Sequence Diagram

# E:\EV_BUSINESS.EV_VENDINGMACHINE.VendingMachine.validateLogin(String, String).jpg

# Analysis Design

Using Data Driven Design, we identify our classes by Noun Identification Techniques.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technician | Password | Product name | Configuration | Place Goods |
| Login | Input Box | Slot | Response | Cancellation |
| Remote Access | Allocation | Calculation | Vending Machine | Validation |
| Storage | SystemControl | Location | Server | Menu |
| Suppliers | Identification | Product | Price | Username |
| Management Interface | Dropdown Menu | Request | Payment | User |
| DatabaseHelper | Purchase | Preferred Item | Card | Pick-Up |
| Amount | Display | Dispense | Credential | Refund |
| Comparison | Detection | Currency | Verification | Sensor |
| Value | UserInterface | Adjustment | Temperature | Thresholds |
| Alert | Imbalance | SystemAdvocate | Check | Selection |
| Error |  |  |  |  |

Heuristic

Red Colour – Too Vague

Green Colour- Is an Operation

Blue Colour – Is an attribute

Brown Colour- Out of Scope

Purple Colour - Similarity

Filtered List

|  |  |  |  |
| --- | --- | --- | --- |
|  | Slot | SystemControl | Vending Machine |
| Server | Product | Management Interface | DatabaseHelper |
| Payment | Sensor | SystemAdvocate |  |
|  |  |  |  |

**Slot**

The Slot class will be used to identify and manage slots in the system. The Slot class is responsible to provide data such as a slot location and the associated product configuration data. Configuration data can consist of (weight, temperature)

**SystemControl**

The Control class is a generic that will survey the internal state of the machine by reading and validating reported sensor and payment hardware data. The Control class is responsible to identify possible impacts, weight, temperature and account imbalances, network or signal failures and alert the SystemAdvocate to invoke the necessary actions to be taken.

**VendingMachine**

This is a Collaboration of Slots. Sensors and Products that can be configured, updated, deleted and managed by the technician who manages the machine. The Vending machine has information of current available cash in the System stored and managed by the machines payment hardware.

**Server**

Is responsible to implement internet traffic services allowing remote access and email alert exchange and error log

**Product**

The Product Class is an entity class that provides data about the current product. Data consist of: (product price, product weight and product name).

**ManagmentInterface**

Is responsible to provide a generated user interface allowing access and modification to internal machine configurations such as product, slot and login details.

**DatabaseHelper**

Is Responsible to manage the local database in the system, implementing CRUD operations on the database which will store information corresponding to product and slot configurations for re-use.

**Sensor**

The Sensor Class is a collection of active sensors, reporting data from available sensors in the system. Reported Data will consist of temperature, weight, and vibration.

**SystemAdvocate**

The SystemAdvocate class is responsible to react to possible alerts coming from the Control class, indicating a possible fault in the system. (Possible faults are faults that have been described in the control class). The SystemAdvocate class will validate and verify what action should be taken to rectify error. Possible actions consist of (reset, initialize refunds, disable an individual product if sold out, and initialise error reports, logs, and outgoing alert messages)

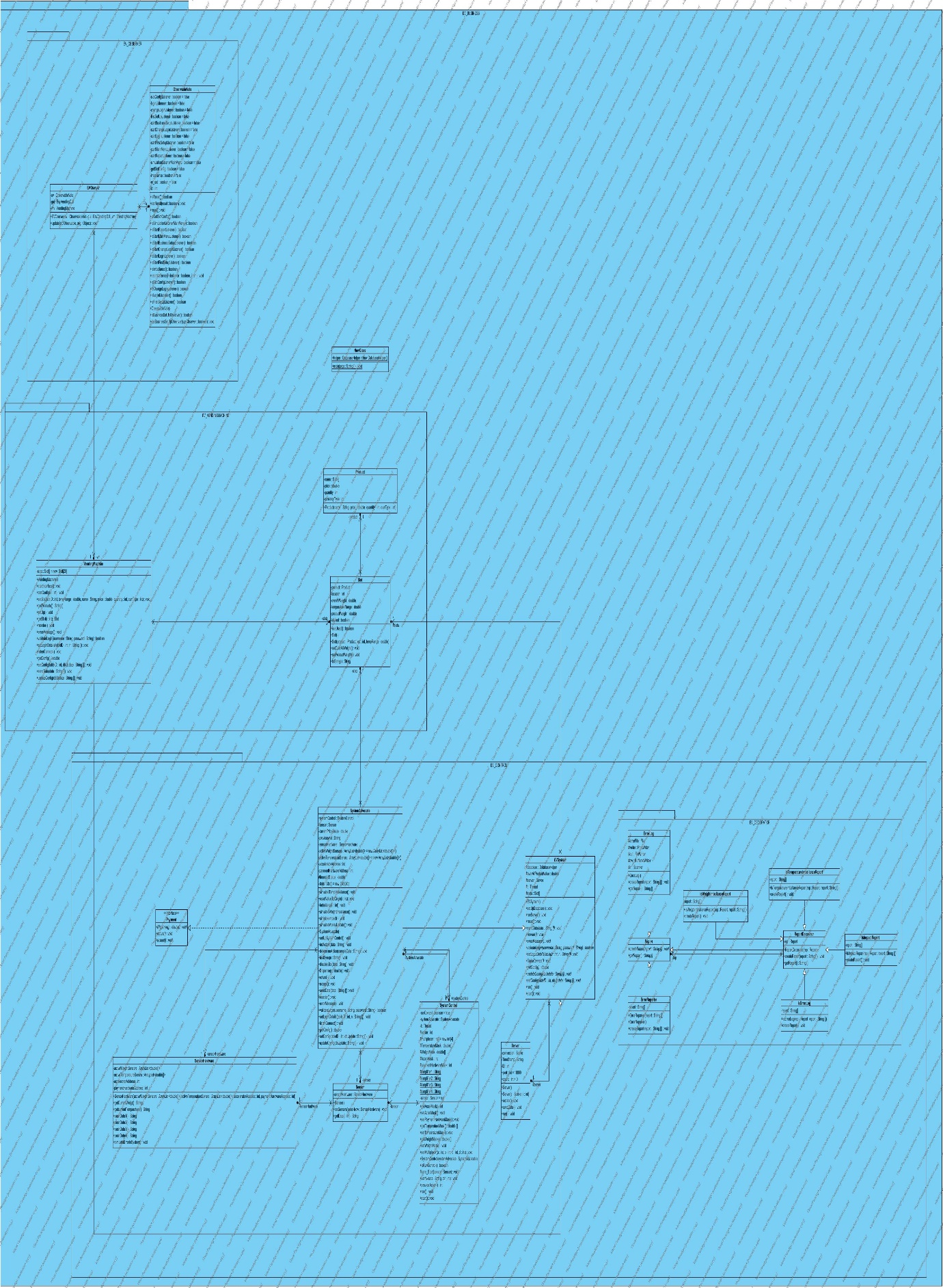
**Payment**

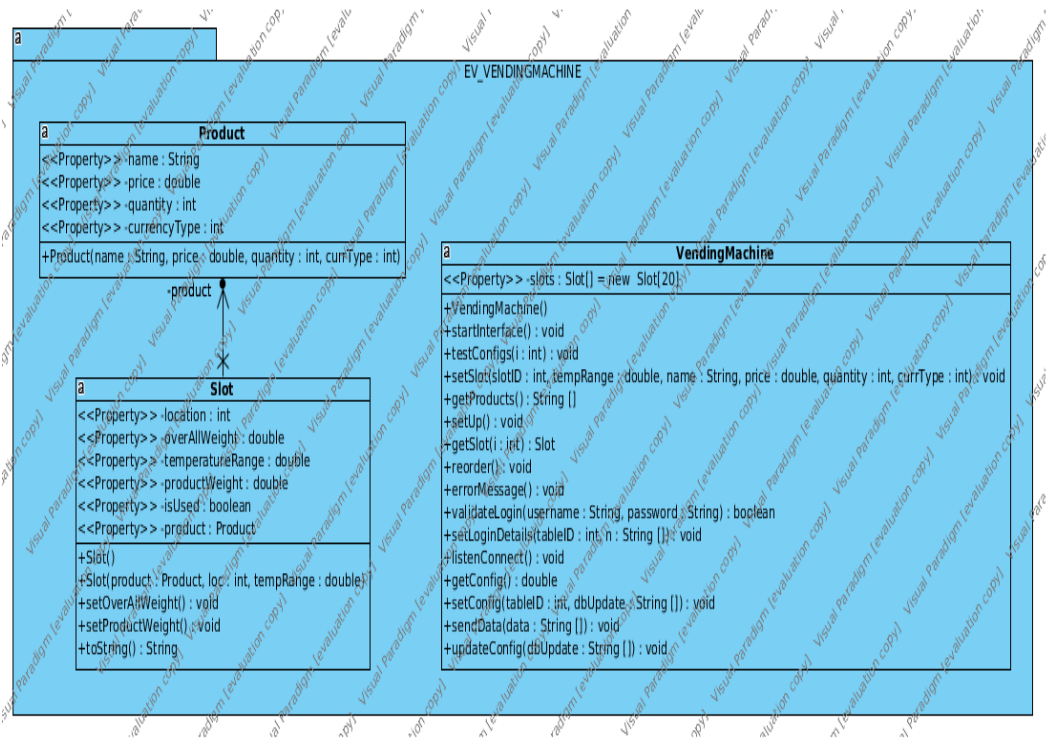
The Money class is responsible to identify payment methods (cash or card payments), dispense outstanding change to users.

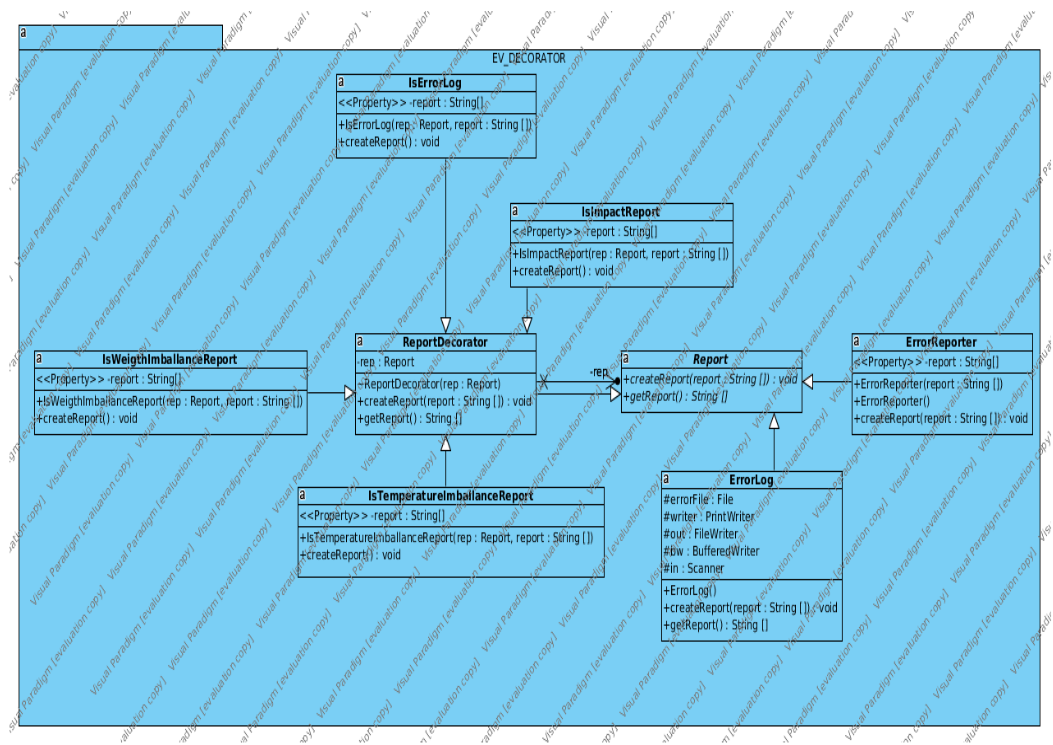
### Class Analysis Diagram

### CD.png

Class Design Diagram.





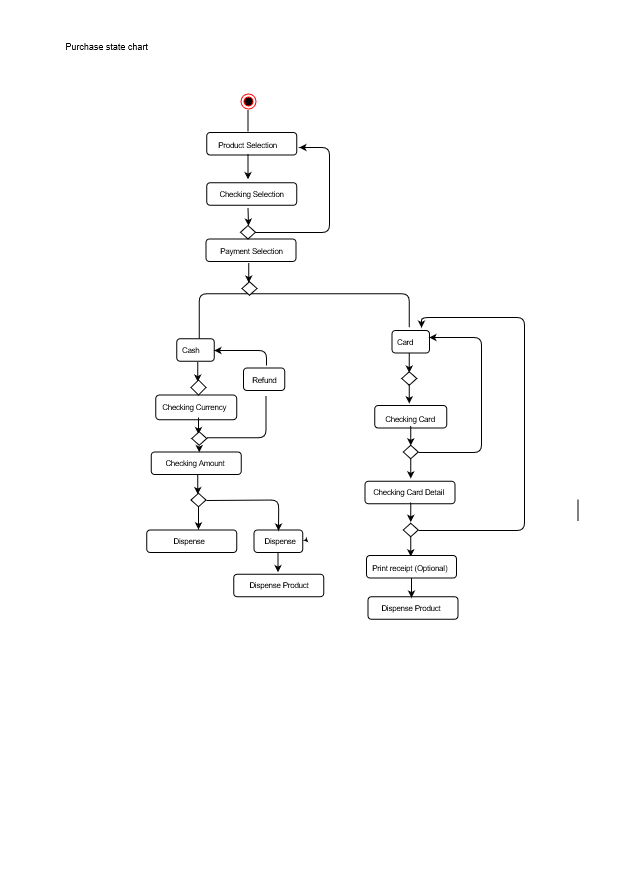


### E:\Observer.png

### E:\Control.png

### Entity Relationship Diagram

### State Diagram



### Descriptions of the pattern used in item

**Observer**: The main goal of the observer pattern is to get an object related to a subject.

Here, the subjects are various and we are catching the subjects and storing them in a new ObservableVals.java. This way, the observers are notifying each time the state of a subject is changed. We are next using those instances ObservableVals.java in various ways. We need it for creating the factory and set the observer on new methods which are created.

**Factory**: The factory here is a GUIFactory.java. It is used with a switch. The ID’s of the switch are the 6 methods of the GUIAdvocate.java classes linked with the Observer. The GUI will generate for each id a proper method and call the **MainMenu.java, Login.java, MachineSetup.java, ChangeLogin.java, BusinessSetup.java and Simulator.java**. The instantiation of all the classes are stacked only once and we are avoiding duplication. Moreover, the main goal of Factory is to allow a better evolution in the future.

**Decorator pattern**: The main goal of the decorator pattern is to allow a better flexibility with the code but it is an alternative to having a sub-classes generation. In our project, we are using the decorator pattern for the report. We are creating two methods, one for create a report and another one to get the report.   
The two methods that are used next in all the classes of the EV\_Decorator package for each report type are called in a Report.java class from the ReportDecorator.java class.  
The different Report classes **(ErrorLog.java, ErrorReporter.java, IsErrorLog.java, IsImpactReport.java, IsTemperatureImballanceReport.java, and IsWeightImballanceReport.java**) are next calling the createReport, getReport methods and setting the report information with a setReport method.

**MVC** : In our project, we have implemented an MVC architectural pattern by using Javax.Szing.JFrame.

# Critique

### Evaluation of the Analysis and Design Artefacts

The overview of the project analysis and design artefacts when considering what we have done and what could have been done. There are few adjustment that was made compared to when the project idea was conceived. We discovered that we spent much time trying to implement the entire project within the short limited time frame given, which make us lose focus for the obvious prioritised implementation. The following are the critiques details

Interface

Cohesion and Coupling

#### Cohesion and Coupling:

##### What we have done

The project has medium coupling and medium cohesion. Medium coupling because of the class package components (e.g Business layer packages) that is composing of some classes in another package components (e.g Gui layer packages).

The threading concurrency synchronisation is interjectory.

##### What could have been done

#### Interface:

##### What we have done

The implementation of observer standard library class.

##### What could have been done

The implementation of the observer class package could have be made an interface for easy extension of the project.

Rerferences

<http://www.ijcsi.org/papers/7-5-94-101.pdf>     [Accessed Feb 26th 2016]