Software Design Pattern

Contents

[Model View Controller 2](#_Toc22072237)

[Use case diagram – SP Control 3](#_Toc22072238)

[Use case of system 4](#_Toc22072239)

[Class diagram 5](#_Toc22072240)

[Data dictionary 6](#_Toc22072241)

[Sequence diagram 8](#_Toc22072242)

[State diagram 9](#_Toc22072243)

## Model View Controller

The Model/View/Controller (MVC) triad of classes is used to build user interfaces.

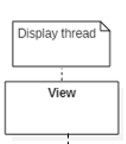
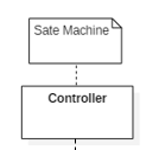
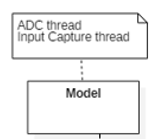
MVC consists of three kinds of objects:

1.- The Model is the application object.

2.- The View is its screen presentation.

3.-The Controller defines the way the user interface reacts to user input.

MVC helps to increase flexibility and reuse of the software. A view must ensure that its appearance reflects the state of the model. Whenever the model's data changes, the model notifies views that depend on it. In response, each view gets an opportunity to update itself.

Diagram MVC

In the MVC diagram is represented the three main interactions between the actor and the system

* When the system is on, the state machine indicates to the system an initialization.
* After initialization, the system is ready to start and waiting that the actor presses the run button.
* The system shall run all the time unless the stop button is pressed.

## Use case diagram – SP Control

The Use Case diagram is a behavioral diagram that shows the actions that can be performed by actors interacting with the system. The stick figures represent actors who have the ability to interact with the system. The ovals are use cases and represent the services that the system provides. The large rectangular box represents the system boundary that separates the actors from the system itself. The solid lines are associations to show an interaction between an actor and a use case. The dotted lines represent either includes or extends.





View-Display

Model



Controller

## Use case of system

This section is used to define the Use case system by describing each individual section with a table as follows:

|  |  |
| --- | --- |
|  |  |
| Use case | READY |
| Actors | Model |
| Description | The state of initialization of the system, sets the SetPoint to zero, and triggers the drivers in the module, as well as initializing the display, until “RUNNING” is triggered. |
| Type | Primary (or secondary) |
| Cross Refs | Number in diagram section |
| Use cases | Enable |

|  |  |
| --- | --- |
|  |  |
| Use case | RUNNING |
| Actors | Model |
| Description | The drivers are engaged, the ADC module is receiving the information from the input capture module, and the display is updating displayed information, until “ON\_HOLD” is triggered. |
| Type | Primary (or secondary) |
| Cross Refs | Number in diagram section |
| Use cases | Enable |

|  |  |
| --- | --- |
|  |  |
| Use case | ON\_HOLD |
| Actors | Model |
| Description | At this point, a stop scanning of the sensor (inputs), and disconnects from drivers, until the “RUNNING” state is triggered again. |
| Type | Primary (or secondary) |
| Cross Refs | Number in diagram section |
| Use cases | Enable |

## Class diagram

|  |
| --- |
| **Speed controller** |
| -fault  -controller On  -Set RPM |
| +Active () |

|  |
| --- |
| **Drivers** |
| -fault  -controller On  -Set RPM |
| +Active () |

|  |
| --- |
| **Speed controller\_ Display** |
| -speed |
| +Refresh\_screen ()  +Display () |

|  |
| --- |
| **Integrative Project** |
| -fault  -controller On  -Set RPM |
| +Active ()  +Change\_Speed ()  +Off\_Button ()  +System\_fault () |

|  |
| --- |
| **PWM controller** |
| -fault  -controller On  -Set RPM |
| +Active () |

## Data dictionary

The following is a data dictionary that describes the classes above. The data dictionary names the class and the class description. Then the individual attributes and operations of the class are listed and described. Next, any relationships between the class and other classes are explained. Any UML extensions are listed at the bottom of each entry.

|  |  |
| --- | --- |
| Element Name | Description |
| Integrative Project | Main controller of the system. Responsible for making logical decisions based on various input. |
|  |  |
| |  |  | | --- | --- | | Attributes |  | |  | Boolean:fault | |  |  | | |  | | --- | |  | | Fault in the system | |  | |
| |  |  | | --- | --- | | Operations |  | |  | Activate():void | |  |  | |  |  | | |  | | --- | |  | | Turn on/activate the system | |  | |  | |
|  |  |
| Relationships | The Controller has a relationship with all parts of the system since it is the main component. It receives input from sources: The **Driver** interface and the **Controller** module. It gives out input to the Speed controller\_ **Display** module. |

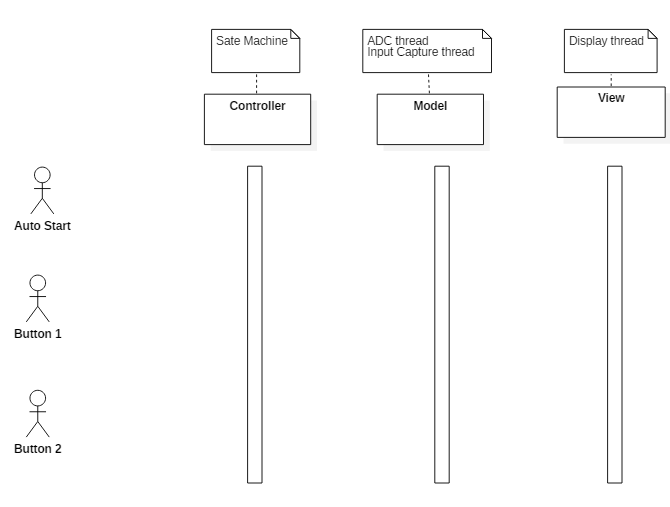
|  |  |
| --- | --- |
| Element Name | Description |
| Speed controller\_ Display | Speed controller display of the system. Responsible for displaying information to the user. |
|  |  |
| |  |  | | --- | --- | | Attributes |  | |  | Boolean:fault | |  |  | |  |  | | |  | | --- | |  | | Fault in the system | |  | |  | |  | |
| |  |  | | --- | --- | | Operations |  | |  | Activate():void | |  |  | | |  | | --- | |  | | Turn on/activate the system | |  | |
|  |  |
| Relationships | The Speed controller\_ **Display** module relates to the Integrative project by taking information as input to display and interact with the User. |

|  |  |
| --- | --- |
| Element Name | Description |
| Speed controller | Speed controller of the system. Responsible for the adjustment of speed using a PI algorithm |
|  |  |
| |  |  | | --- | --- | | Attributes |  | |  | Boolean:fault | |  |  | |  |  | | |  | | --- | |  | | Fault in the system | |  | |  | |  | |
| |  |  | | --- | --- | | Operations |  | |  | Activate():void | |  | Pwm () | | |  | | --- | |  | | Turn on/activate the system | | Signal function in charge of setting controller | |
|  |  |
| Relationships | The Speed **controller** module relates to the Integrative project by setting input to the model. And depends directly from a **PWM** defined. |

|  |  |
| --- | --- |
| Element Name | Description |
| Drivers | Speed controller display of the system. Responsible for displaying information to the user. |
|  |  |
| |  |  | | --- | --- | | Attributes |  | |  | Boolean:fault | |  |  | |  |  | | |  | | --- | |  | | Fault in the system | |  | |  | |  | |
| |  |  | | --- | --- | | Operations |  | |  | Activate():void | |  |  | | |  | | --- | |  | | Turn on/activate the system | |  | |
|  |  |
| Relationships | The Speed controller\_ **Display** module relates to the Integrative project by taking information as input to display and interact with the User. |

## Sequence diagram

The sequence diagrams below represent specific scenarios that the Integrative Project system will encounter. The boxes at the top represent objects in the system. These same objects appear in the class diagram above. The solid lines represent a message sent from one object to another. The dotted lines represent a response from an object.



Main()

Install threads()

Install drivers()

Initialize values

Initialize Display

Run

Read ADC()

Send PWM()

calculate()

refresh screen()

READY

Running

Pause

Close drivers()

Stop comm()

refresh screen()

OnHold

## State diagram

A [state diagram](file:///C:\Users\jesus\Documents\GitHub\Proyecto_Integrador\ESTRUCTURA%20DEL%20PROYECTO\1)%20Requirements\stakeholder\Maquinas%20de%20estados_V1.pdf) is shown in Figure 1. The state diagram shows all the possible states that the system can be in and all the transitions necessary to get to those other states. The boxes represent different states and the arrows represent the transitions between states.

