Software Design Pattern

Contents

[Model View Controller 2](#_Toc21698534)

[Use case diagram – SP Control 3](#_Toc21698535)

[Use case of system 4](#_Toc21698536)

[Class diagram 5](#_Toc21698537)

[Data dictionary 6](#_Toc21698538)

[Sequence diagram 7](#_Toc21698539)

[State diagram 7](#_Toc21698540)

## 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

## 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. If use case A includes use case B then that means if case A happens case B also happens. If case A extends case B then case A is a more specialized version of case B.

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 | Disable (or active) |
| Actors | Model |
| Description | If all sensors are working properly and there are no internal problems with the system,  prepare the system for use. Otherwise, alert the User to the presence of a problem and  do not activate the system. |
| Type | Primary (or secondary) |
| Cross Refs | Number in diagram section |
| Use cases | Enable |

## Class diagram

|  |
| --- |
| Speed controller |
| -fault  -controller On  -Set RPM |
|  |

|  |
| --- |
| Speed controller |
| -fault  -controller On  -Set RPM |
|  |

|  |
| --- |
| Drivers |
| -fault  -controller On  -Set RPM |
|  |

|  |
| --- |
| Speed controller\_ Display |
| -speed |
| +Request\_data ()  +Display () |

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

|  |
| --- |
| Speed controller |
| -fault  -controller On  -Set RPM |
|  |

## 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](file:///C:\Users\jesus\Documents\GitHub\Proyecto_Integrador\ESTRUCTURA%20DEL%20PROYECTO\1)%20Requirements\stakeholder\20190527%20Requisitos%20del%20proyecto%20integrador%20CESEQ.pdf) | 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 4 sources: The radar, the driver interface, the path prediction module, and the car status module. It gives out input to two sources, the status information class and the longitudinal control controller class. |

## Sequence diagram

## State diagram

A [state diagram](../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.

