Introduction

With a day at the International Space Station costing close to $6.8 million or $190 000 hourly per crew member at the International Space Station, and cargo costing close $90 000 per kilogram for a passive trip and $120 000 per kilogram for a round trip (Johnson, 2021), keeping the cost of a trip at the lowest is paramount. Making sure the correct rocket is paired with a spacecraft is essential to keeping the cost of a space trip as low as possible. The pairing of a spacecraft and a rocket, whether the Falcon 9 or the Falcon Heavy, is also a process that needs to be efficient and should not exacerbate the payload. Having been tasked with building a simulation for a space mission, the Misfits have built a simulation that attempts to keep the payload of a SpaceX trip to its lowest possible, this using design patterns that also assist in making the simulation efficient. The launch of a space shuttle is also complex, which we have managed to simplify using these design patterns, which also assist in providing security and efficiency.

Functional Requirements

Rocket Testing

Rockets require testing before it is allowed to launch by running the launch simulation. If the rocket fails, then it cannot proceed to the launch simulation, upon failure tests are repeated. The test run on rockets is called a Static Fire Test. In order to increase the accuracy of test results, each test must be run 10 times. Almost all tests must pass, a 5% error margin is allowed.

Firstly, a number between 0 and 100(inclusive) is generated randomly for ten Static Fire tests. The above-mentioned number will be divided by 100 to get a randomly generated probability. A rocket fails a test if probability is lower than 95% and passes if probability is greater or equal to 95%. The number of passed tests will have to be tracked using an accumulator and the value is passed into a Binomial Test function, which will determine whether the 5% margin of error is satisfied. If the Binomial test has determined the test to be unsuccessful, then the rocket will return to the lab where the problem should be fixed. After the engineers are confident that the problem is fixed, the rocket should get retested in the same way as in the above-mentioned point. To reduce the possibility of a rocket failing every test and then being recalled only to return and fail again, the lower bound of the random generator gets updated to the maximum number of the randomly generated values amongst the ten tests. This means the rocket will surely improve after being recalled.

Configuration Deciding Phase

Due to the high costs of building or operating the different rockets and spacecraft, it is imperative to carefully decide on the configuration of the vehicles. This configuration will be heavily influenced by the objective of the trip. The two main aims of the space shuttle launches are to deliver humans and cargo to the International Space Station and also to deploy a constellation of Starlink satellites and scatter them across Low Earth Orbit. In order to optimise costs, the appropriate rocket configuration must be decided depending on one of the aforementioned objectives.

Transporting humans

To transport humans from one of the four SpaceX launch facilities to the International Space Station. A spacecraft called The Crew Dragon will be required. It consists of two main parts, a space capsule and an expendable trunk module The spacecraft can have a maximum of seven people aboard. The total price for a successful Crew Dragon launch comes to $160 million which is about $23 million per astronaut if all seven seats are filled. To optimize a fleet of human-only trips to the International Space Station, each spacecraft must be as close to maximum capacity as possible so that less of them are used. In the case of private travel for the sake of space tourism, SpaceX charges close to $55 million per seat which can potentially earn SpaceX a return of $225 million if all seven seats are taken.

Transporting Cargo

The space travel company uses a different spacecraft from the Dragon series, simply called The Dragon Spacecraft, in order to transport cargo to the International Space Station. The Dragon Spacecraft can carry a total of 6 000kg worth of cargo to orbit and 3 000 kg for return. The load is split into two components, pressurized cargo which is stored inside the capsule and unpressurized cargo which is stored in the trunk. In order to select the appropriate configuration, the full weight of the cargo will need to be known and properly distributed into the trunk or capsule depending on the type of cargo. The total cost to launch the spacecraft also sits at $160 million which would be $26 670 per kilogram of cargo.

Transporting Cargo and Humans

The Crew Dragon Spacecraft can transport cargo and humans. The maximum number of crew members on such a spacecraft is 7 and the payload capacity of the cargo is 6 000 kg to orbit and 3000 kg for return. The price of this spacecraft is the same as transporting humans and cargo separately, $26 670 per kilogram for cargo and $55 million per seat.

Launching Starlink Satellites

In order to launch SpaceX's Starlink satellites into orbit, the company has two rockets to consider using. These rockets are the Falcon 9 and Falcon Heavy (which comprises three

Falcon 9 rockets). A single Falcon 9 Rocket can launch up to 60 Starlink Satellites meaning that it's counterpart, the Falcon Heavy, can have a maximum of 180 Starlink satellites. The price of launching a new single Falcon 9 rocket into orbit is $62 million (roughly $1 million per satellite). while the Falcon Heavy costs $150 million (roughly $833 334 per satellite) at most. These rockets consist of two stages.

Optimization Approach

Since, the main objective is to optimize the cost for each launch by providing an appropriate rocket configuration. To do this, the price per payload will be calculated using the input data from the user. The user will first specify the type that they want to launch up into space. After that, the quantities will be specified for each selected item. Using the aforementioned launch costs for each item, the system will determine which configuration is the most appropriate

Initial Design

See task1.pdf for initial project design

Design Patterns

Builder

A Falcon Heavy rocket consists of three cores, 27 Merlin engines. The Falcon 9 consists of 9 Merlin Engines. Both the Falcon Heavy and the Falcon 9 have standing legs, which we have decided are negligible due to the fact that the do not greatly impact in the cost of the rocket and they are almost identical in both the Falcon 9 and Falcon Heavy and the purpose of this assignment is to reduce cost, this cost was thus rendered negligible. The rockets need to be put together or built, and the most intuitive design pattern to do this is the Builder Design Pattern. Similarly with the Space Shuttle, which consists of a spacecraft and a rocket, the two will be brought together to build the shuttle.

Participants

• **Director**: SpaceX

○ construct() used to construct the Space Shuttle

• **Builder**: ShuttleBuilder

- Provides an interface for SpaceShuttleBuilder
- buildRocket() builds a Falcon 9 or a Falcon Heavy
- buildShuttle() builds the Space Shuttle consisting of one of the Falcons and one of the Dragons

• **ConcreteBuilder**: SpaceShuttleBuilder

• **Product:** SpaceShuttle - Holds pointers to the different components: Rocket, SpaceCraft and Starlink

Prototype

The creation of a Starlink satellite is a lot less complicated than that of the rockets and spacecrafts and satellites are identical. The Factory Method design pattern could have been used for the production of the Starlink satellites but was far too complex for the simple task at hand, therefore the Prototype design pattern is used to simply create the desired number of Starlinks by cloning them.

Participants

• **Prototype:** Satellite

• **ConcretePrototype**: Starlink

- clone() returns the starlink object

• **Client**: StarlinkReplicator – This class performs the cloning by cloning the passed in Starlink Satellite

Factory Method

Spacecrafts are created using the Factory Method design pattern. As there are two different types of spacecrafts which are majorly similar, the factory method design pattern works best as there is an interface for producing the Spacecraft, however it’s subclasses that ‘decide’ whether to produce a Dragon spacecraft or a Crew Dragon spacecraft. The decision as to whether a Dragon or Crew Dragon will be created depends on whether a trip will contain cargo, crew members or both. It is possible that the mission will be to transport Starlink satellites, which does not require a spacecraft.

Participants

•**Creator:** SpaceCraftFactory

•**ConcreteCreator:** CrewDragonFactory, DragonFactory

•**Product:** SpaceCraft

•**ConcreteProduct:** CrewDragon, Dragon

Memento

A Space Shuttle can contain a number of different configurations, whether it contains a Crew Dragon and a Falcon Heavy or a Dragon and a Falcon 9, it needs to be stored in a safe manner as to be used by other classes. A number of different configurations will be tried, and the winning configuration (based on cost) will be stored. Once the winning configuration is chosen it is imperative that it not be corrupted or accidentally changed in any way, as this will affect the price of the trip and possibly cause safety concern, this may be disastrous. The Memento design pattern ensures encapsulation, serving as a solution for this possible issue.

Participants

• **Originator**: SpaceShuttle

• **Memento**: WinningShuttle

• **Caretaker**: WinningConfig

Chain of Responsibility

The function of a satellite is to collect information and transmit it back to earth, this is usually using complex methods such as radio waves. The simplest way to simulate how this information is transmitted through many different satellites before reaching the crew on the ground is by use of the chain of responsibility design pattern. The starlinks communicate the coordinates they can take up in the sky, each Starlink will have a chance to choose a position and then communicate with all the other Starlinks to make sure that, that position isn't already occupied by another satellite, this is done through lasers, if a position is taken the starlinks attempts to find another position. Once a Starlink finds a free position it relays the information back to GroundCrew, who will receive this information via the Radio

Participants

• **Handler:** Handler

• **ConcreteHandler**: Starlink, GroundCrew

• Message - Acts as the request which is being handled.

  - The Communication Medium variable determines which Object handles it.

    - "Radio" - GroundCrew
    - "Laser" - Starlink

Observer

From launch to landing, the crew in the Space Shuttle need to know what the Space Shuttle is doing, there needs to be a way for the two to communicate. The Observer design pattern ensures that a change in the Space Shuttle will be relayed to the crew members so that they are aware where in the trip they are, if they have arrived, and if there is something of concern within the operation of the Space Shuttle. MissionControl class will observe any changes in the Rocket and SpaceShuttle class, crew will be able to see output on any changes in the mentioned classes through the Mission Control class

Participants

• **Subject**: Subject

• **ConcretSubject**: Rocket, SpaceShuttle

• **Observer**: Observer

• **ConcreteObserver**: MissionControl

Command

We have shown how the Space Shuttle and rockets communicate with the crew members, but how will crew members communicate with the rockets? The rocket needs to be commanded to start a launch, to halt the launch and to return home. The command design pattern was implemented to make this communication possible. Because we do not want the crew to have direct access to the internal system of any of the components on the rocket, the command design pattern relays the message to either start launch, stop a launch or return home from the crew members to the rocket.

Participants

• **Command**: Command

• **ConcreteCommand**: LaunchCommand, HaltCommand, ReturnCommand

• **Invoker**: Controller

• **Receiver**: Rocket, SpaceCraft

State

The Merlin Engines that the Falcon Heavy and Falcon 9 Rockets both have two stages. A process called stage separation is done by the interstage which allows the first stage to be separated from the second stage. The State design pattern handles which stage the rocket is currently being pushed into orbit with and makes sure that stage two is not fired prematurely as it is compulsory that stage one fires before stage two.

Participants

• **State**: Stage

• **ConcreteState**: Stage1, Stage2

• **Context**: Rocket

Template Method

Considering that each rocket follows the same procedure for a mission, the template design pattern will be used to provide an interface for rockets, this way rockets will not stray away from the simple launch, attach and detach sequence, the only difference is the internal structure of these methods and how a rocket achieves a mission

Participants

• **AbstractClass**: Rocket

• **ConcreteClass**: FalconNine, FalconHeavy

Composite

A Falcon 9 contains 9 Merlin engines, a Falcon Heavy, containing 27 Merlin engines, is simply three Falcon 9 rockets. The composite design pattern is used to compose a Falcon Heavy rocket using three Falcon 9 rockets. The FalconNine class defines the primitive objects of the composition, while the FalconHeavy class is the composite participant, which logically could serve as the base class of three FalconNine classes which would be treated uniformly and differently from the FalconNine classes which are not derived from the FalconHeavy class. To avoid having duplicate classes when only identical objects are needed, this is done through delegation using the addFalconNine() method instead of inheritance.

Participants

• **Component**: Rocket

• **Leaf**: FalconNine

• **Composite**: FalconHeavy

Decorator

Crew and cargo are the main addition to the space mission. We needed an intuitive way to represent human beings and food, gear, etc. being added to the space craft. The decorator pattern works well in this sense as it ‘decorates’ the space craft with crew and/or cargo, adding them to either the Dragon or the CrewDragon.

Participants

• **Component**: SpaceCraft

• **ConcreteComponent**: CrewDragon, Dragon

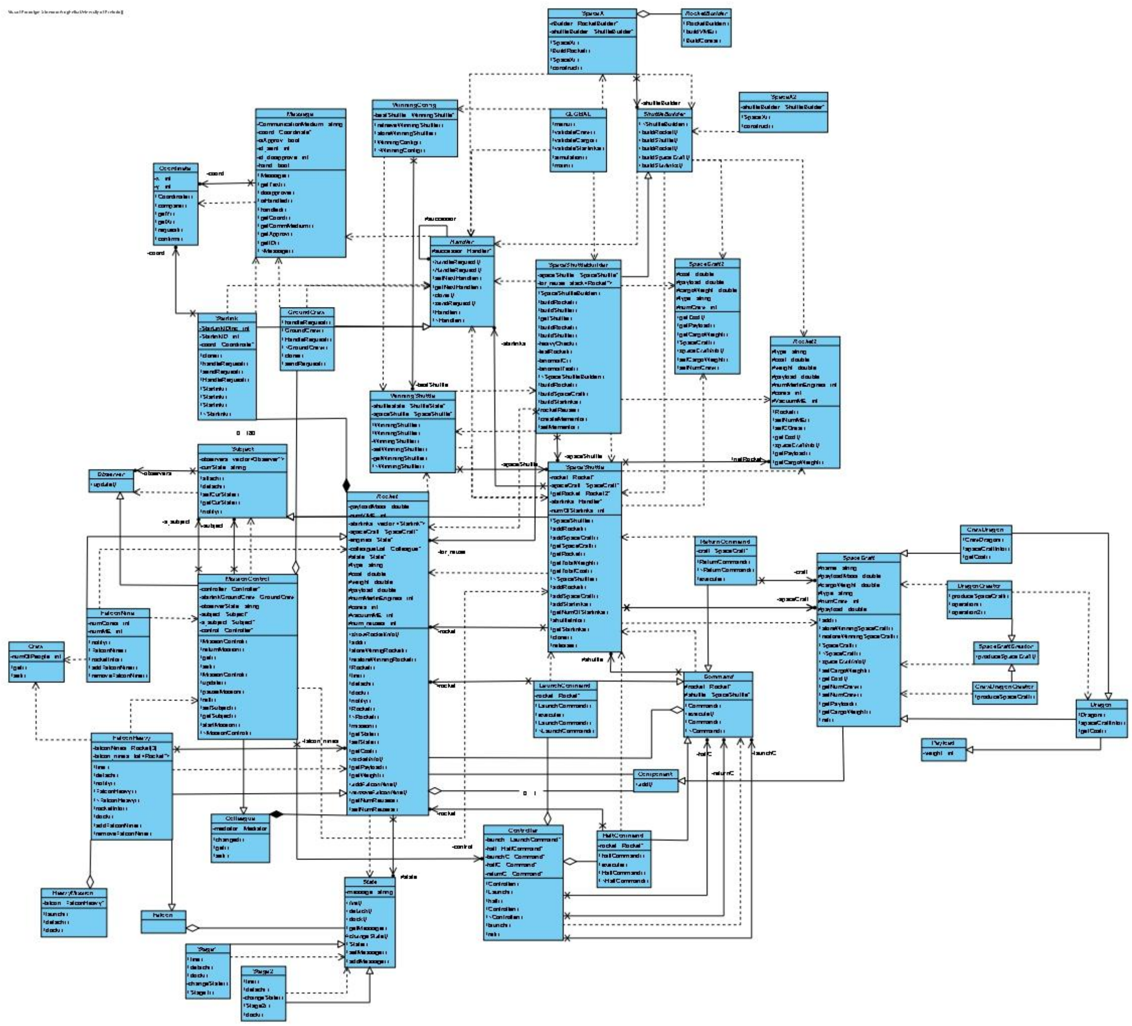
• **Decorator**: Decorator

• **ConcreteDecorator**: Cargo, Crew

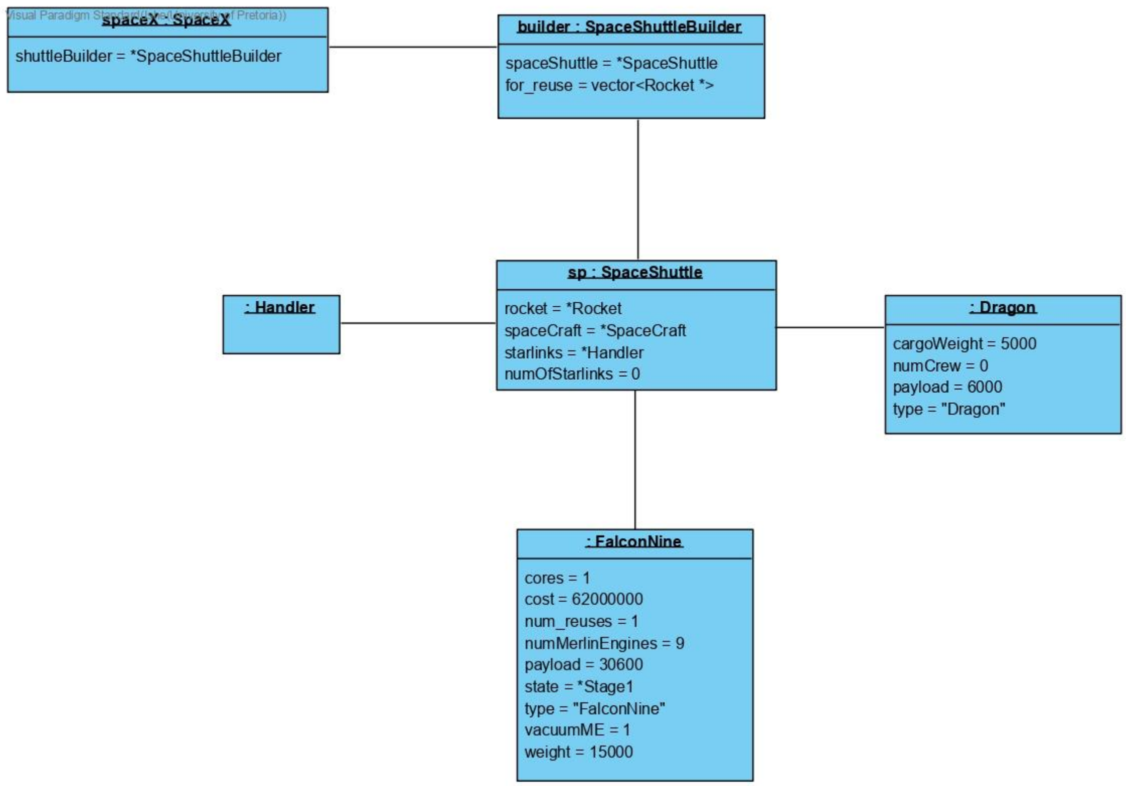
Diagrams

Class Diagram

The UML class diagram below represents the design patterns laid out above

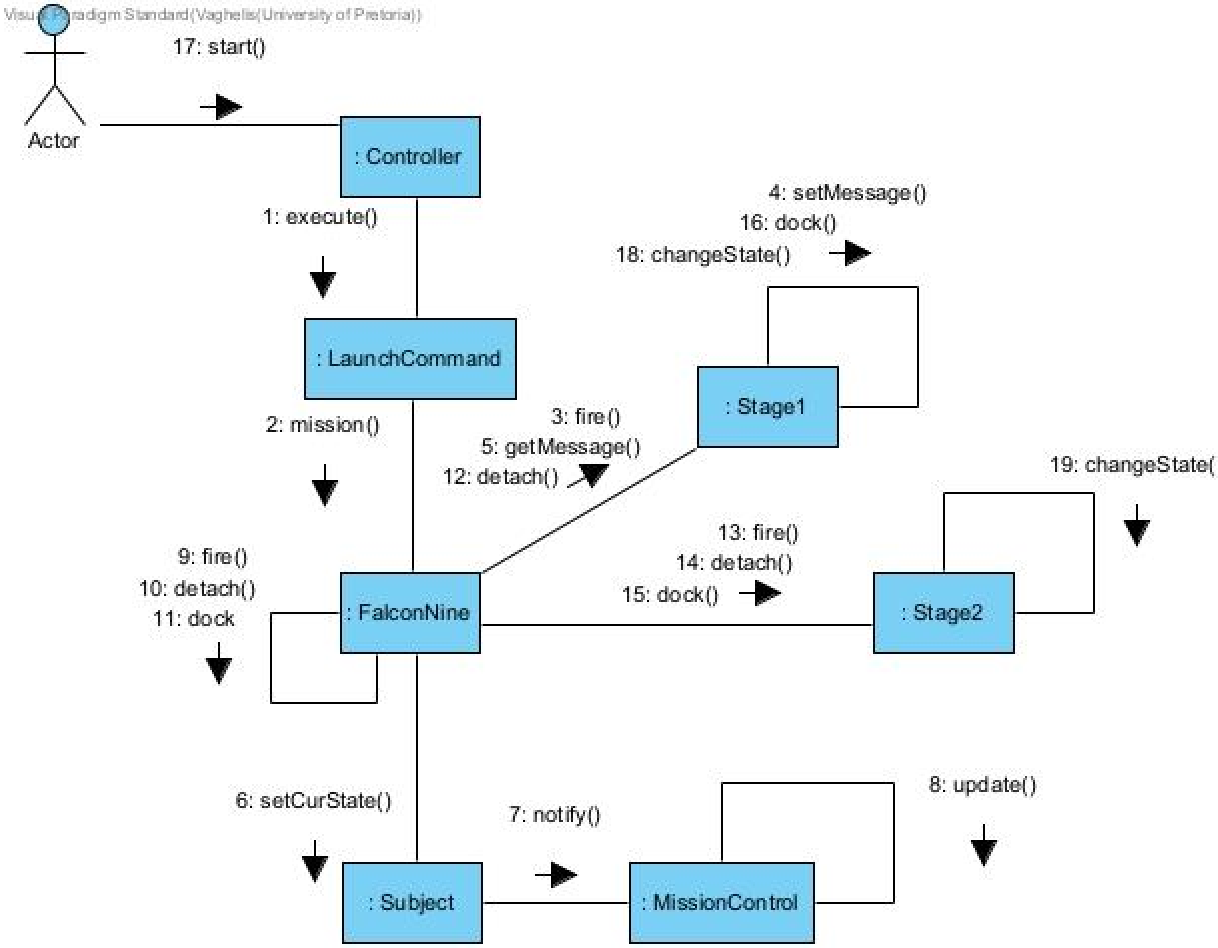


Object Diagram

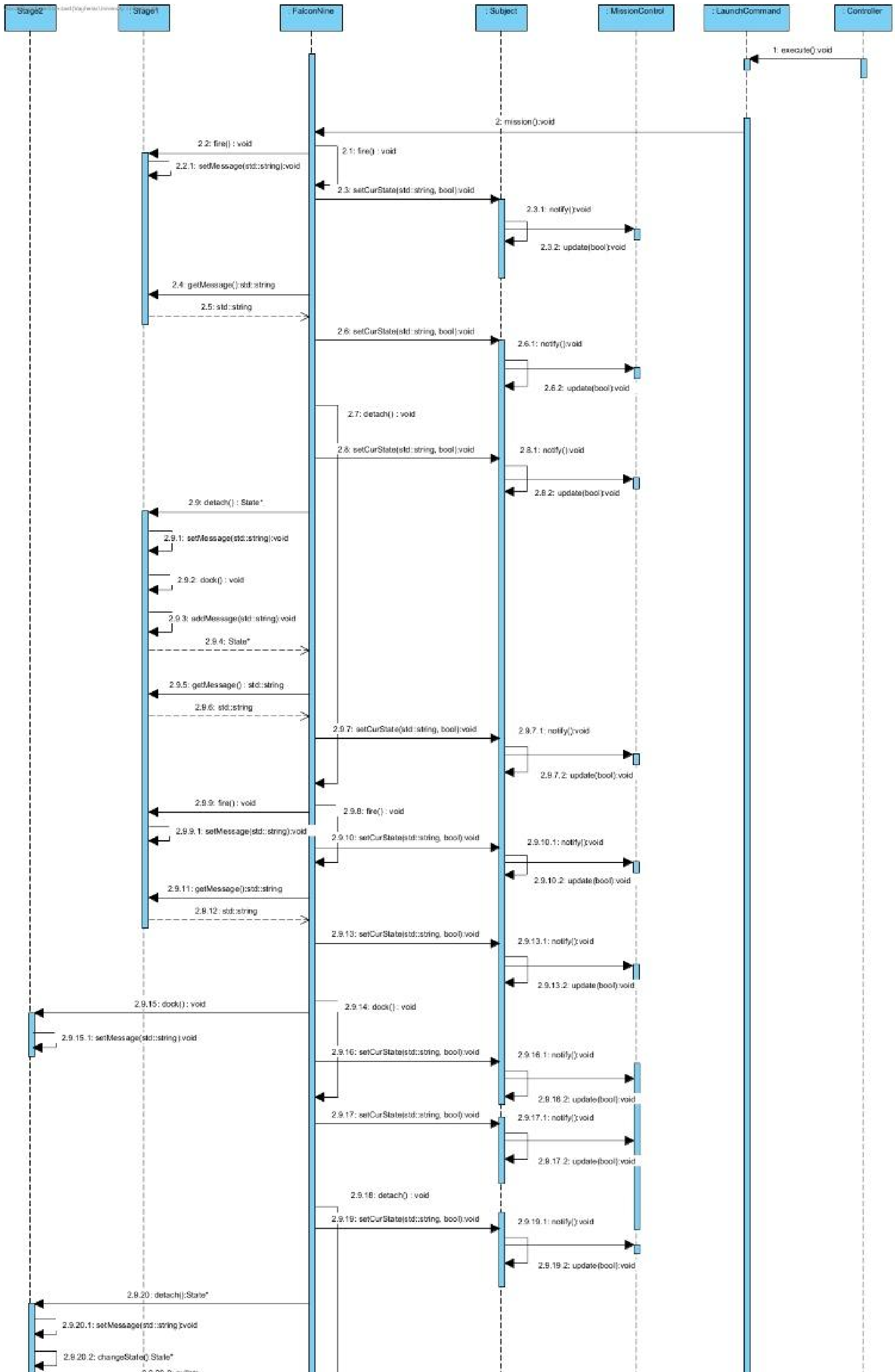


Visual Paradigm Standard(University of Pretoria))
spaceX : SpaceX
shuttleBuilder = *SpaceShuttleBuilder
builder : SpaceShuttleBuilder
spaceShuttle = *SpaceShuttle
for_reuse = vector<Rocket *>
: Handler
sp : SpaceShuttle
rocket = *Rocket
spaceCraft = *SpaceCraft
starlinks = *Handler
numOfStarlinks = 0
: Dragon
cargoWeight = 5000
numCrew = 0
payload = 6000
type = "Dragon"
: FalconNine
cores = 1
cost = 62000000
num_reuses = 1
numMerlinEngines = 9
payload = 30600
state = *Stage1
type = "FalconNine"
vacuumME = 1
weight = 15000

Communication Diagram

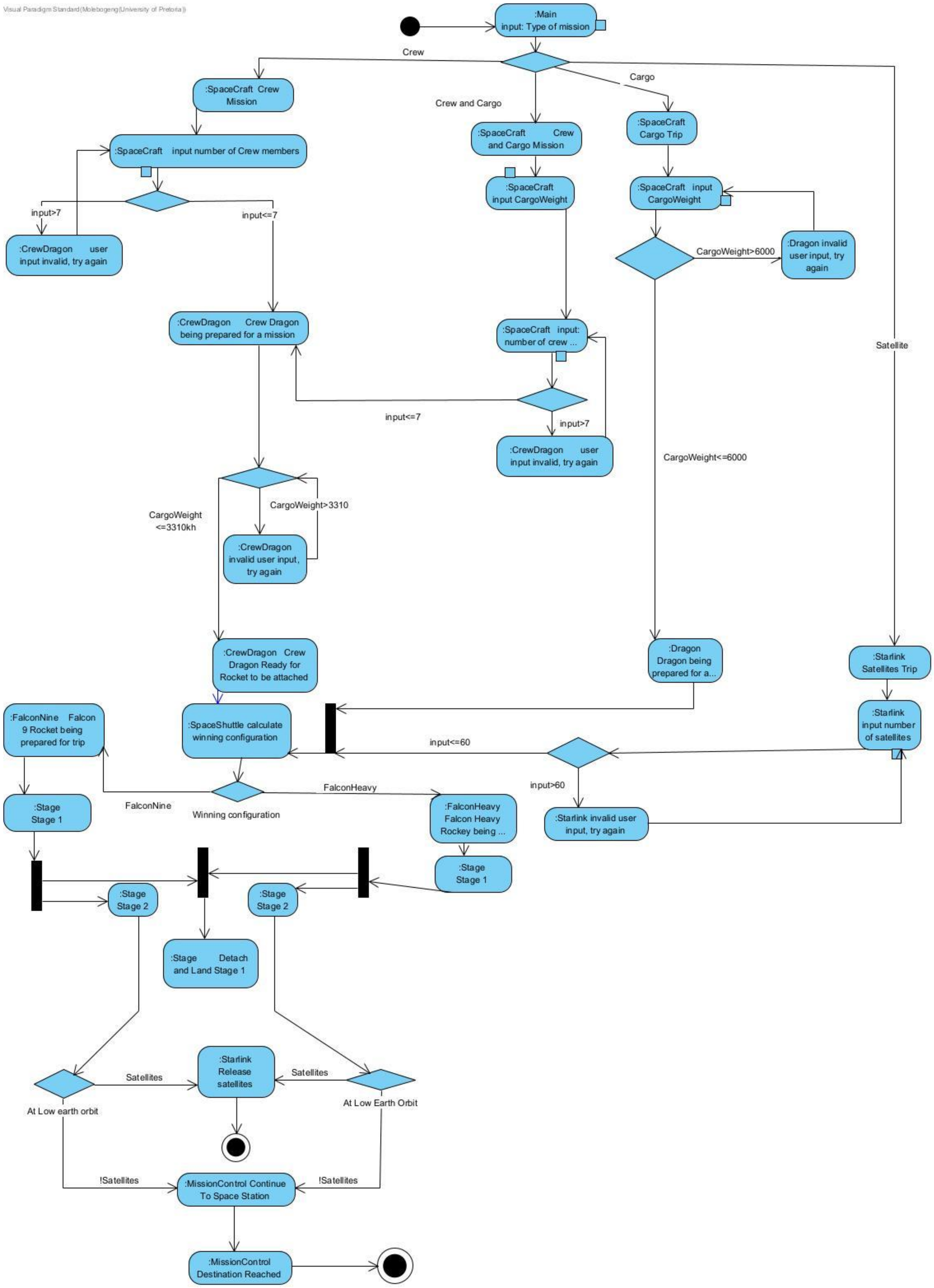


Sequence Diagram

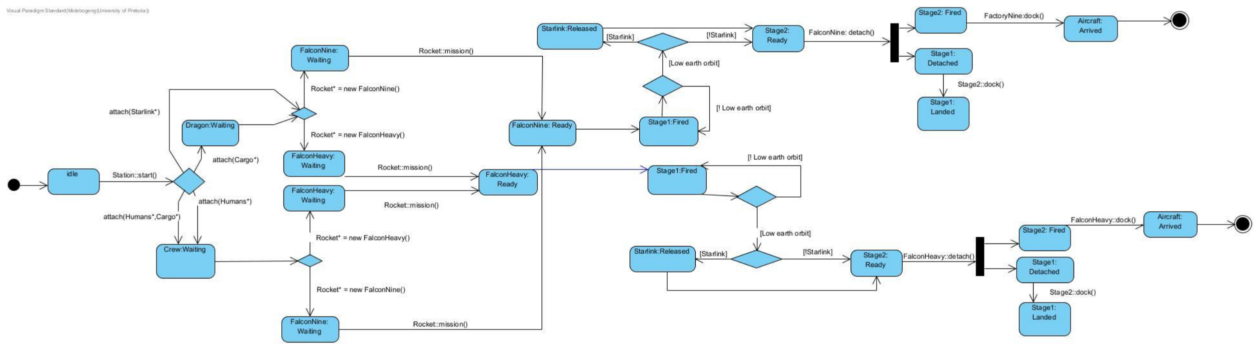


Stage2
: Stage1
: FalconNine
: Subject
: MissionControl
: LaunchCommand
: Controller
1: execute():void
2: mission():void
2.2: fire() : void
2.1: fire() : void
2.2.1: setMessage(std::string):void
2.3: setCurState(std::string, bool):void
2.3.1: notify():void
2.3.2: update(bool):void
2.4: getMessage():std::string
2.5: std::string
2.6: setCurState(std::string, bool):void
2.6.1: notify():void
2.6.2: update(bool):void
2.7: detach() : void
2.8: setCurState(std::string, bool):void
2.8.1: notify():void
2.8.2: update(bool):void
2.9: detach() : State*
2.9.1: setMessage(std::string):void
2.9.2: dock() : void
2.9.3: addMessage(std::string):void
2.9.4: State*
2.9.5: getMessage() : std::string
2.9.6: std::string
2.9.7: setCurState(std::string, bool):void
2.9.7.1: notify():void
2.9.7.2: update(bool):void
2.9.9: fire() : void
2.9.8: fire() : void
2.9.9.1: setMessage(std::string):void
2.9.10: setCurState(std::string, bool):void
2.9.10.1: notify():void
2.9.10.2: update(bool):void
2.9.11: getMessage():std::string
2.9.12: std::string
2.9.13: setCurState(std::string, bool):void
2.9.13.1: notify():void
2.9.13.2: update(bool):void
2.9.15: dock() : void
2.9.14: dock() : void
2.9.15.1: setMessage(std::string):void
2.9.16: setCurState(std::string, bool):void
2.9.16.1: notify():void
2.9.16.2: update(bool):void
2.9.17: setCurState(std::string, bool):void
2.9.17.1: notify():void
2.9.17.2: update(bool):void
2.9.18: detach() : void
2.9.19: setCurState(std::string, bool):void
2.9.19.1: notify():void
2.9.19.2: update(bool):void
2.9.20 : detach():State*
2.9.20.1: setMessage(std::string):void
2.9.20.2: changeState():State*

Activity Diagram


:Main
input: Type of mission
Crew
Crew and Cargo
Cargo
Satellite
:SpaceCraft Crew Mission
:SpaceCraft input number of Crew members
input>7
input<=7
:CrewDragon user input invalid, try again
:CrewDragon Crew Dragon being prepared for a mission
:SpaceCraft Crew and Cargo Mission
:SpaceCraft input CargoWeight
:SpaceCraft input: number of crew ...
input<=7
input>7
:CrewDragon user input invalid, try again
:SpaceCraft Cargo Trip
:SpaceCraft input CargoWeight
CargoWeight>6000
:Dragon invalid user input, try again
CargoWeight<=6000
CargoWeight>3310
CargoWeight <=3310kh
:CrewDragon invalid user input, try again
:CrewDragon Crew Dragon Ready for Rocket to be attached
:Dragon Dragon being prepared for a...
:Starlink Satellites Trip
:Starlink input number of satellites
input<=60
input>60
:Starlink invalid user input, try again
:SpaceShuttle calculate winning configuration
:FalconNine Falcon 9 Rocket being prepared for trip
FalconNine
Winning configuration
FalconHeavy
:FalconHeavy Falcon Heavy Rockey being ...
:Stage Stage 1
:Stage Stage 2
:Stage Stage 1
:Stage Stage 2
:Stage Detach and Land Stage 1
Satellites
Satellites
:Starlink Release satellites
At Low earth orbit
At Low Earth Orbit
!Satellites
!Satellites
:MissionControl Continue To Space Station
:MissionControl Destination Reached

State Machine Diagram



Final Project

Github: https://github.com/Ivy-leet/COS214-Project

Master branch: https://github.com/Ivy-leet/COS214-Project/tree/master

Google Docs:https://docs.google.com/document/d/1CxtLXyrQoeuoTRp6z5TaBD9app77zxOkUQutc38IVJk/edit?usp=sharing

References

Johnson, M., 2021. *NASA.* [Online]

Available at: https://www.nasa.gov/leo-economy/commercial-use/pricing-policy

[Accessed 22 November 2021].

SpaceX, 2021. *SpaceX.* [Online]

Available at: https://www.spacex.com/

[Accessed 22 November 2021].