2021 COS 214

MAGNIFICENT 7 PROJECT REPORT

Team Members

Caleb Groeneveld - u19340631

Damian Jordaan - u20473509

Cornelius Fourie Engelbrecht - u18021388

James Lake - u19153113

Francois Viviers - u18055461

Zelda Wiese - u18191135

Keelan Cross - u19151952

TABLE OF CONTENTS

INTRODUCTION

The following report indicates a detailed description of our system to help Elon Musk simulate SpaceX and Starlink in order for them to better plan and optimize their launches.

For the project we identified the requirements for optimizing the launches of the rockets, we have designed and modelled a system using our integrating design patterns. Our system tests the rockets and insures the launch is successful.

Our system test and retests the falcon rockets by using a static fire which starts the engines and which will ensure that everything works as it should. We ensured that the dragon spacecraft sends and returns our humans and the cargos to the International Space Station and the Dragon Spacecraft.

Our system ensures that the Starlink satellites communicate with each other with the use of lasers and radio signals to communicate with the workers on the ground.

The last part of our system is the launch simulator. It makes it possible to run a simulation in test mode which will run as if it is a real launch. The simulations can be interrupted, tweaked and then allowed to continue. This launch simulator can simulate an actual launch simulations which can also be stored and run in batches.

OUR DESIGN PATTERNS

We used 13 Design patterns within our system, namely: Iterator, Memento, Adaptor, Command, State, Decorator, Abstract Factory, Composite, Strategy, Singleton and Facade. We used these design patterns to speeds up our design and helps to communicate our system to others. Each design pattern we use explains how the classes within our system work together.

No. 01 — Iterator

"Provide a way to access the elements of an aggregate object sequentially without exposing its underlaying representation." We used the iterator within the Starlink class to hold the 60 satellite's within the linked list.

No. 02 — Memento

"Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later." We used the memento design pattern to store the instance of the simulation object. (It stores the payload and state within the simulation object.)

No. 03 — Adapter

"Convert an interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces. " We use the Adapter design pattern to add additional functionality to the Starlink class, to allow the Starlink to release the satellite.

No. 04 — Command

"Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations." The command design pattern ensures that the command to launch the spacecraft happens or rotate and steer the rocket from the ground.

No. 05 — State

"Allow an object to alter its behavior when its internal state changes. The object will appear to change its class." We use the state design pattern to check if the rocket has been tested before, if it is working or if it is broken. This design pattern allows the rocket to be launched or go trough a few static tests to determine the rocket state.

OUR DESIGN PATTERNS

CONTINUES

No. 06 — Abstract Factory

"Provide an interface for creating families of related or dependent objects without specifying the concrete classes." We used the abstract factory pattern within our RocketBuilder and MerlinEngineFactory to create a Merlin Engine.

No. 07 — Composite

"Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly." We used the Composite design pattern within our system because the staging of a multistage rocket works within the same way as the composite design pattern.

No. 08 — Singleton

"Ensure a class only has one instance, and provide a global point of access to it." We decided the Singleton design pattern would be useful as it would allow a single point of access and lower the interface complexity.

No. 09 — Strategy

"Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it." We used the strategy design pattern within the builder design pattern to allow for multiple variations of the builder design pattern functionality and differentiate between single and four stack rockets.

No. 10 — Template

"Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure." We use the template design pattern within the composite design pattern. It is used to add extra functionality to the base rocket.

OUR DESIGN PATTERNS

CONTINUES

No. 11 — Decorator

"Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality." We used the decorator design pattern because it works perfectly with the way rocket motors interact with the engine nozzle. In that the nozzle changes the behavior and characteristics of the engine.

No. 12 — Builder

"Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it." We used the builder design pattern because it works perfectly to assemble the various components of the rocket within the composite design pattern.

No. 13 — Facade

"Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use." We used the Facade design pattern because it provides a higher-level interface that is used within our Main to ensure easier access to the subsystems and our other design patterns.

FUNCTIONALITY OF OUR SYSTEM

Our system is designed to stimulate the launch and testing of rockets. Our systems main functionality is to send rockets up to space. The system allows rocket creation mostly making use of dynamic capabilities to ensure that you are not locked within a specific structure which allows rockets to be built in a method other than the norm.

In this project we had to simulate the building, launching and working of the different types of space x rockets which we needed to differentiate between, namely the Falcon 9 and Falcon heavy. The Falcon 9 consists of 9 Merlin engines, which has a success record of 99.7%, which we simulated in our system on startup. We made use of observers on each engine to notice when something goes wrong and lets the rocket know what went wrong. Our team also created functionality to built up a rocket using the builder design pattern. This combines the nozzle with everything else a rocket needs together. Our team made use of factories, since these rockets have basic values that are shared across all of them, making it easier to make new rockets for future simulations.

The Command Center is being used as a Façade, as well as a singleton, this simplifies everything for the end user, as all the commands are included inside of this that is needed. We also made sure to use mementos to keep track of the states of rockets, incase we need to initialize them from the start again. We made sure to use states to test our system, and incase something went wrong with the launch/startup, we can instantly put our system into a state where we can fix the problem and then launch again if we want.

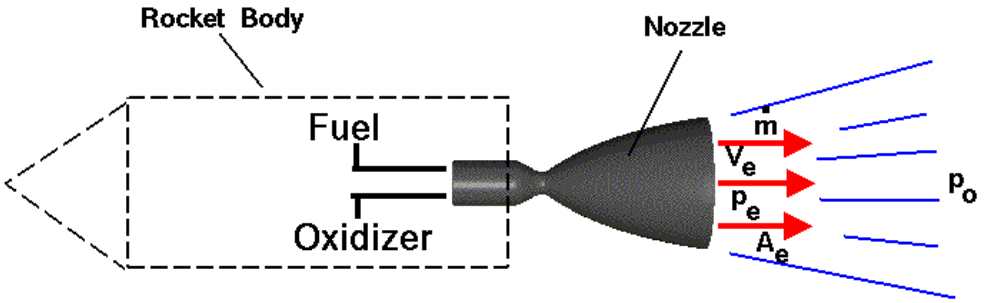
We have different types of payloads mainly the crew dragon, the starlink and the cargo dragon. The launch simulations are within our main. We have a test mode launch which pauses the rocket which allows the capability to change the rocket within certain manners. We have a launch simulation testing if the rocket launch or fails. And of course our actual launch where we launch the actual rocket. We provided functionality to save rockets for later use (create rockets and use them later).

The system is designed to be able to work with real world rocket calculations for both the engine and simulation of the actual rocket. The calculations used are presented below.

FUNCTIONALITY OF OUR SYSTEM

CONTINUES

Rocket Thrust



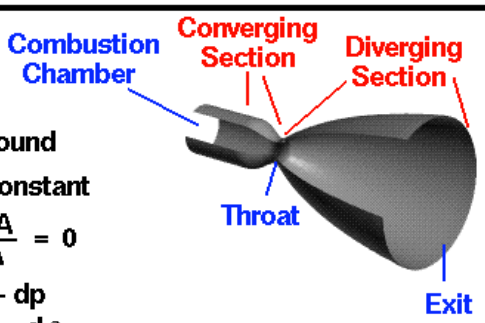
$\dot{m}$ = mass flow rate
p = pressure
V = Velocity
A = Area

Thrust = $F = \dot{m} V_e + (p_e - p_0) A_e$

Nozzle Design

Converging–Diverging (CD) Nozzle

$\dot{m}$ = mass flow rate
V = velocity
ρ = density
γ = specific heat ratio
A = area
p = pressure
M = Mach
a = speed of sound



Conservation of Mass: $\dot{m} = \rho V A = \text{constant}$

$$\frac{d\rho}{\rho} + \frac{dV}{V} + \frac{dA}{A} = 0$$

Conservation of Momentum: $\rho V \, dV = -dp$

Isentropic Flow: $\frac{dp}{p} = \gamma \frac{d\rho}{\rho}$ $\qquad dp = a^2 \, d\rho$

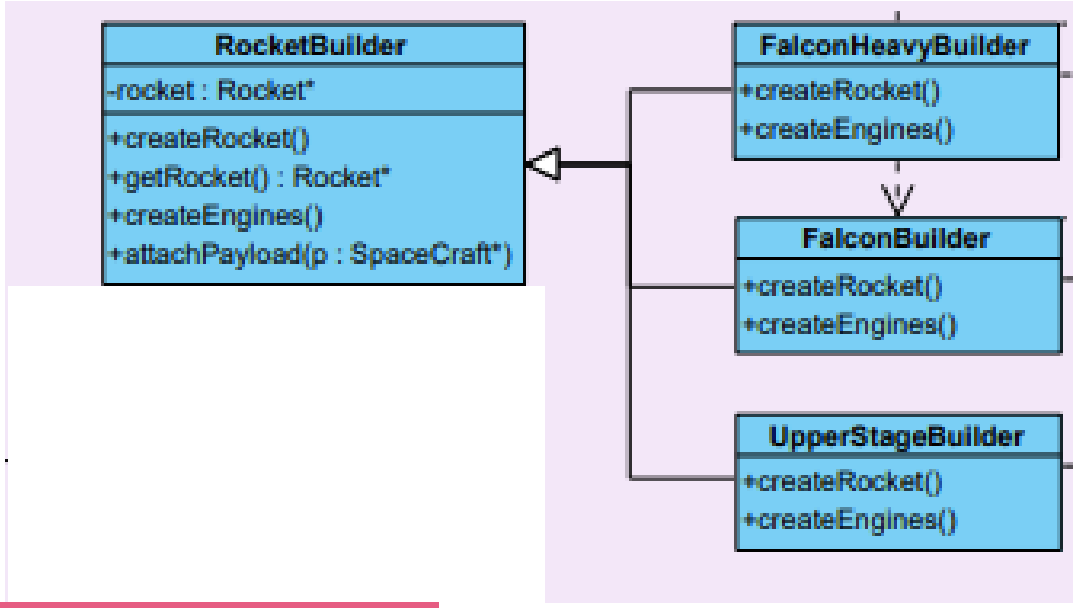
Combine with Momentum: $-M^2 \frac{dV}{V} = \frac{d\rho}{\rho}$

Combine with Mass: $(1 - M^2) \frac{dV}{V} = -\frac{dA}{A}$

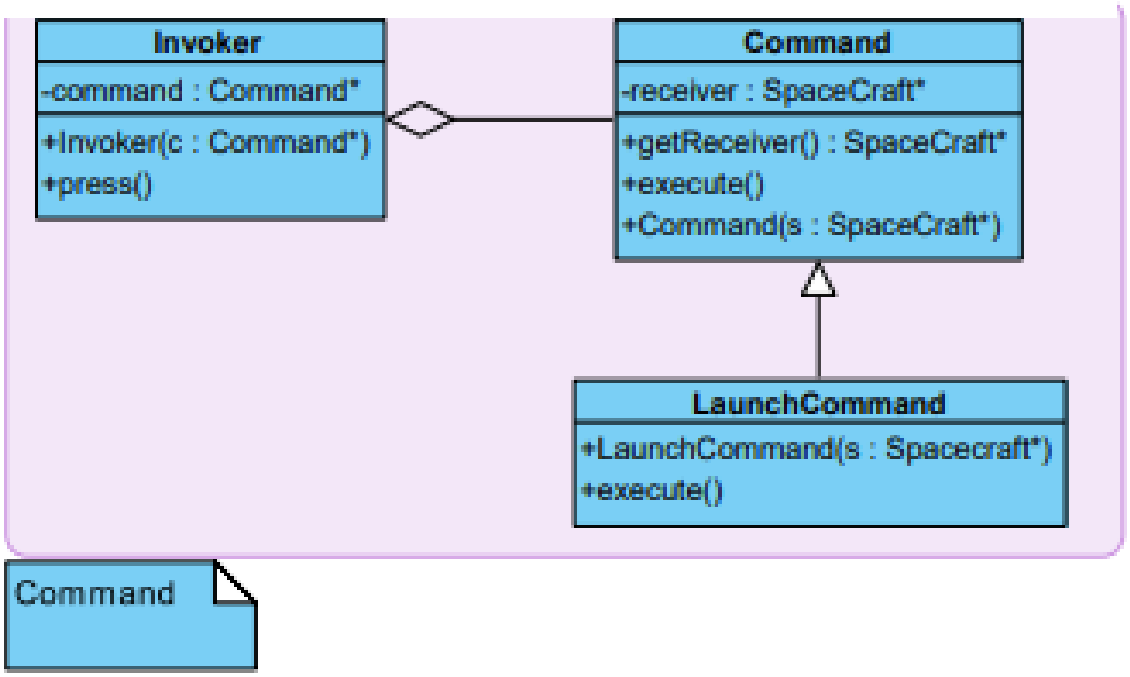
For subsonic flow ($M < 1$), increase in area ($dA > 0$) causes flow velocity to decrease ($dV < 0$)

For supersonic flow ($M > 1$), increase in area ($dA > 0$) causes flow velocity to increase ($dV > 0$)

CLASS DIAGRAMS



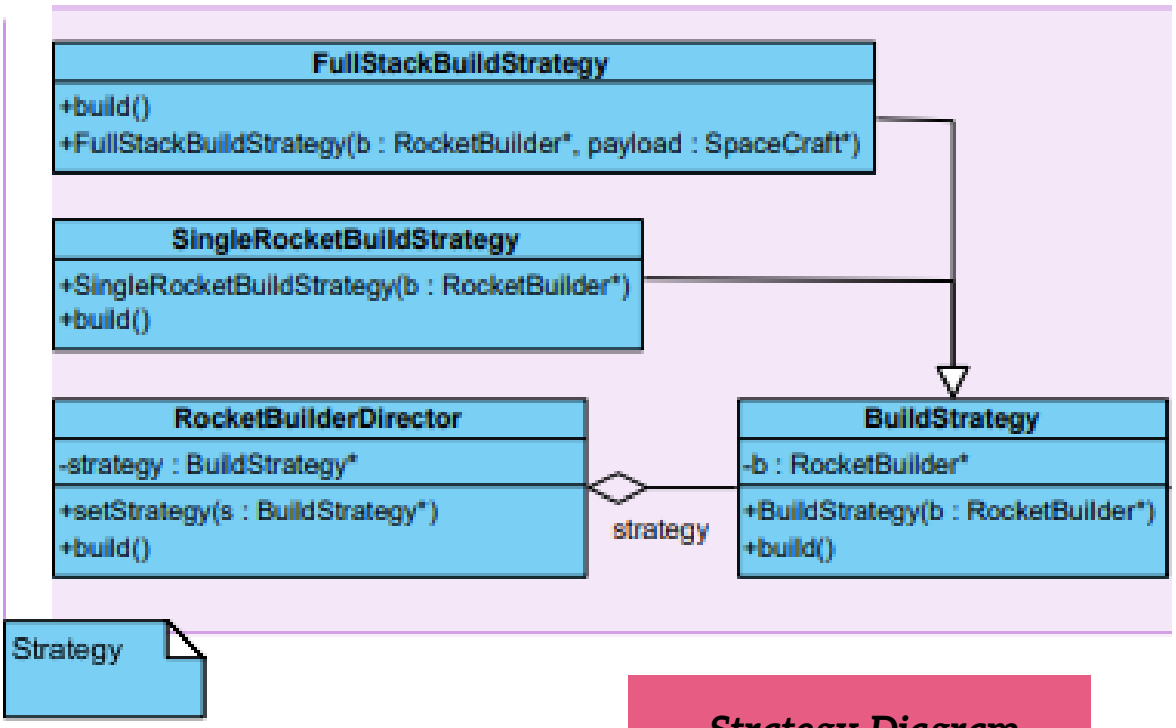
Builder Diagram



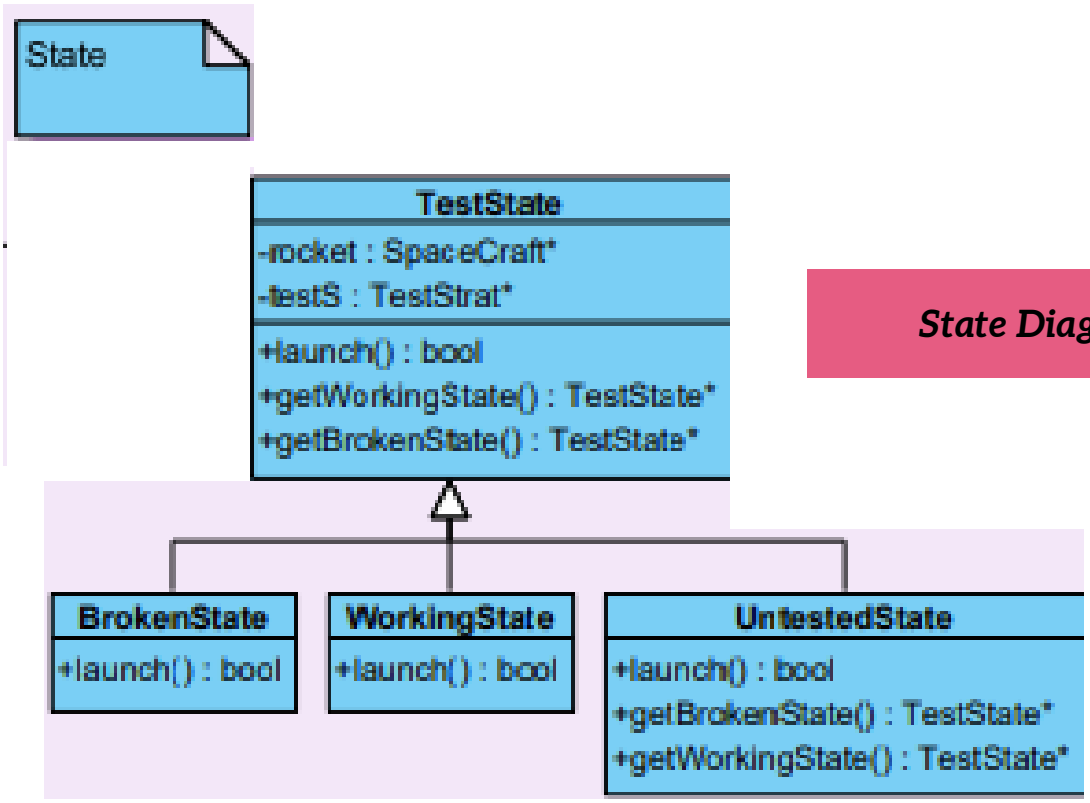
Command Diagram

CLASS DIAGRAMS

CONTINUES



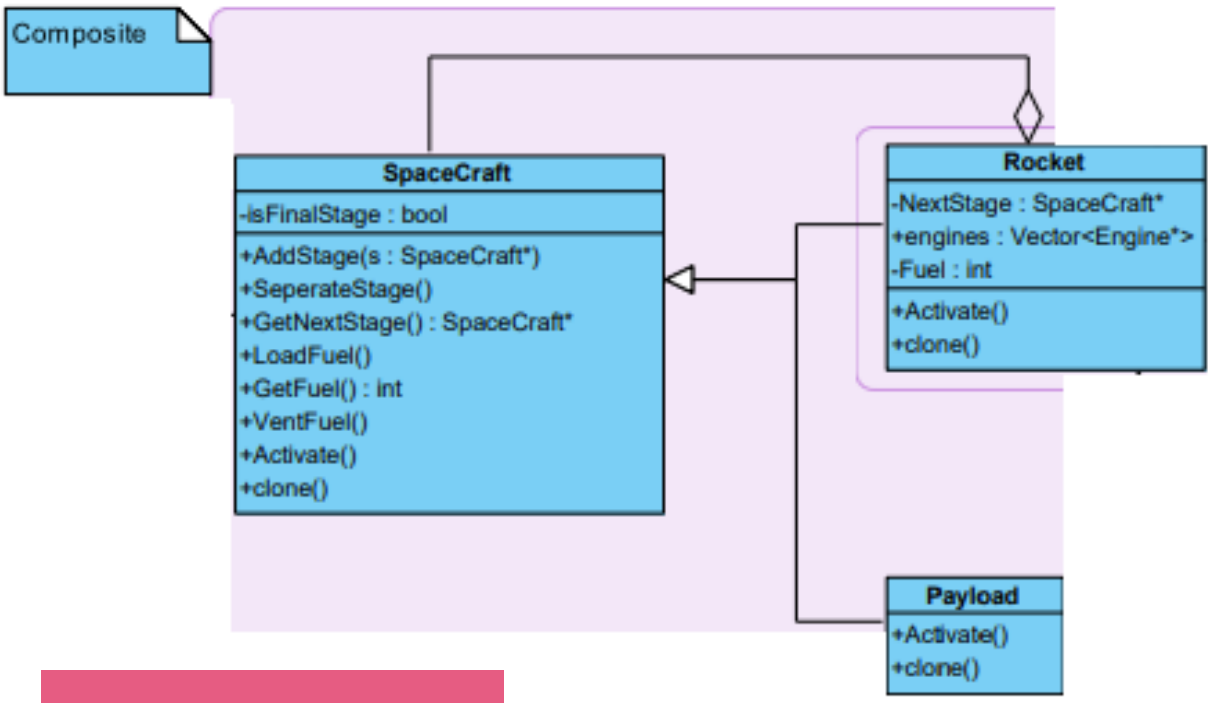
Strategy Diagram



State Diagram

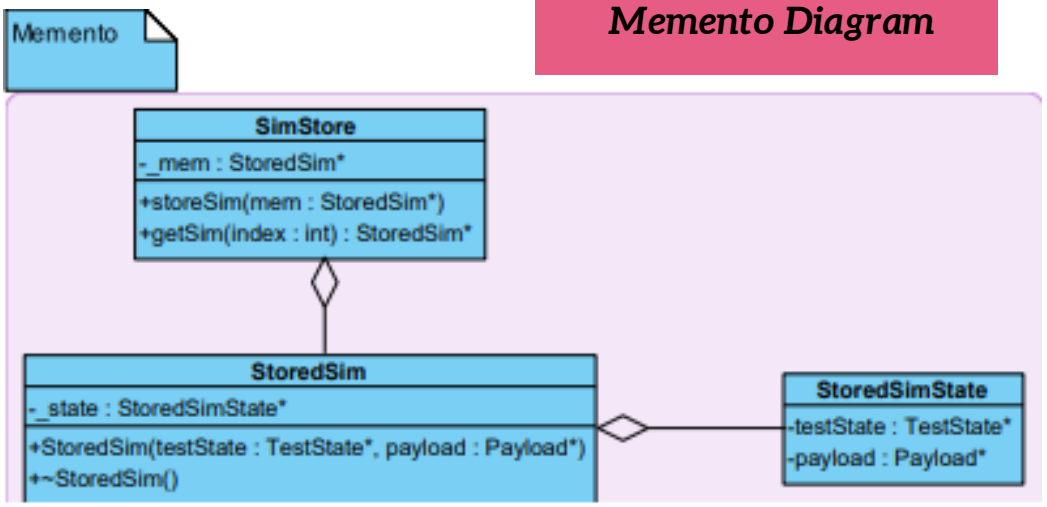
CLASS DIAGRAMS

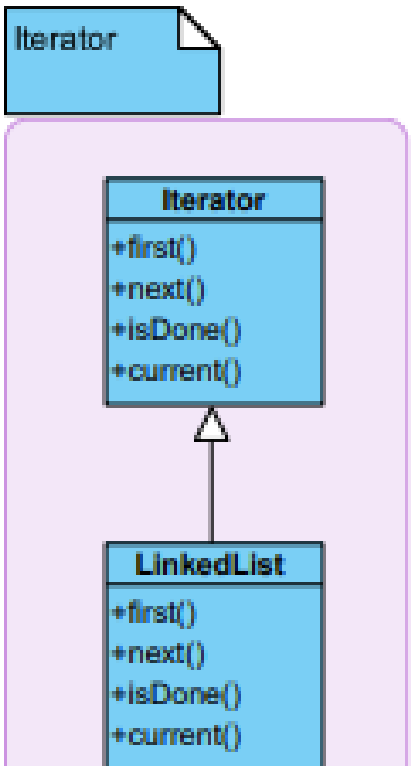
CONTINUES



Composite Diagram

Memento Diagram

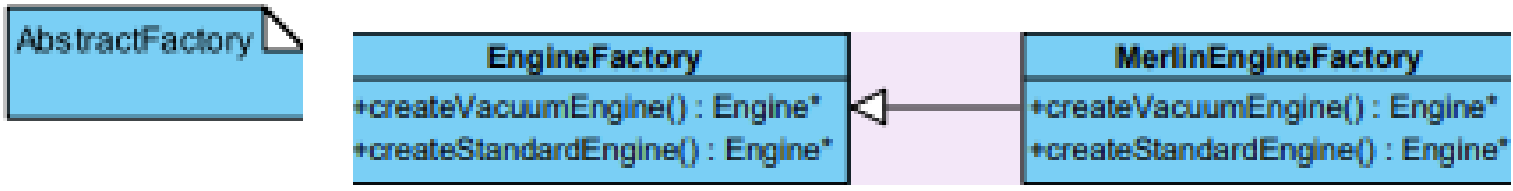




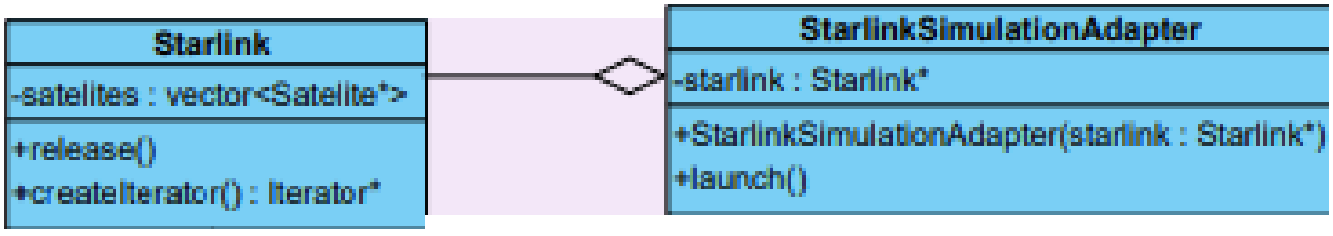
Iterator Diagram

CLASS DIAGRAMS

CONTINUES



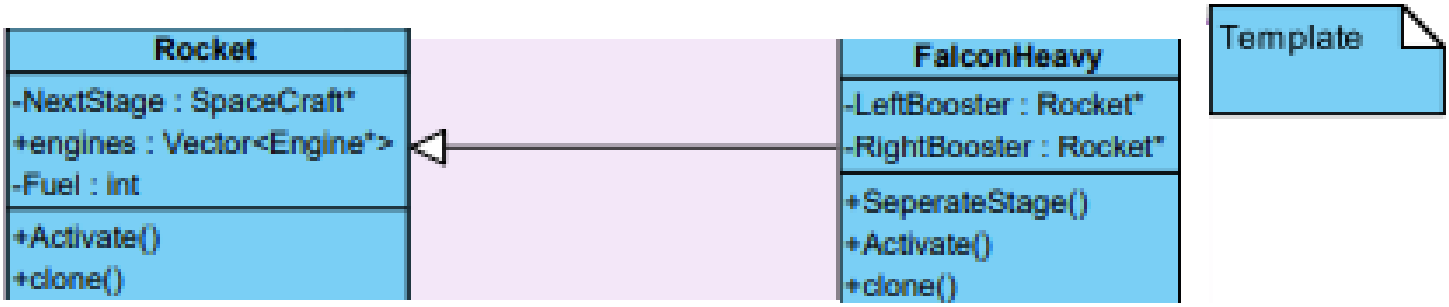
Abstract Factory Diagram



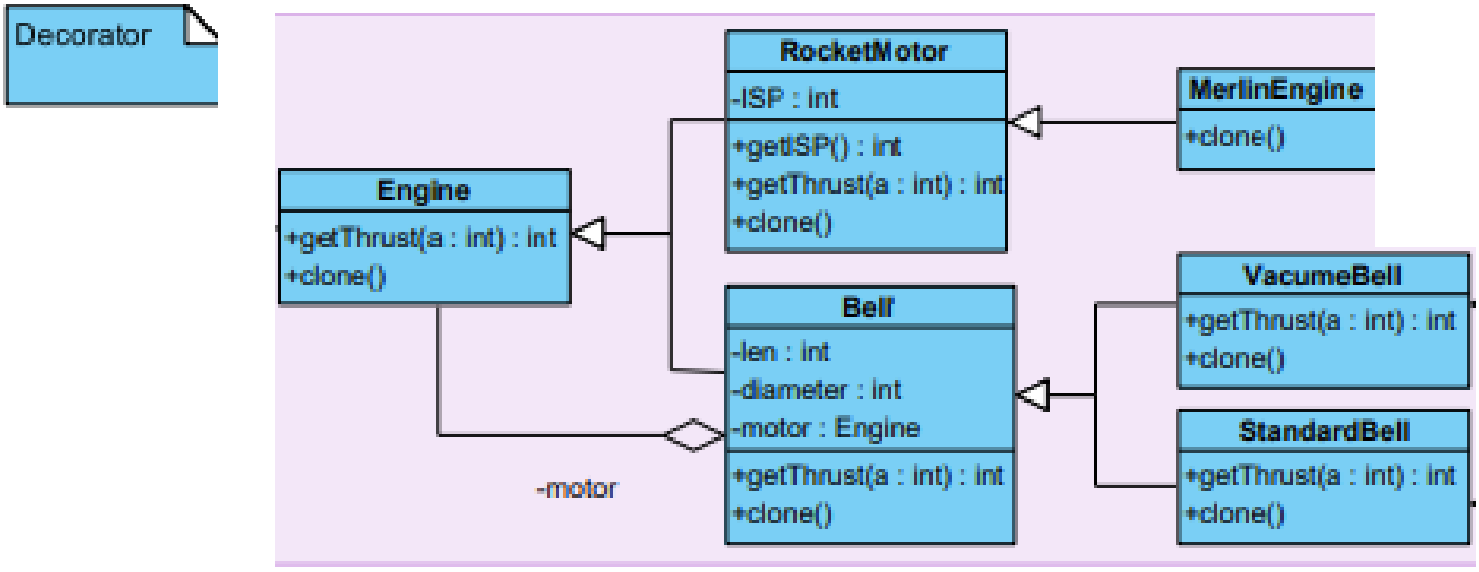
Adapter Diagram

CLASS DIAGRAMS

CONTINUES

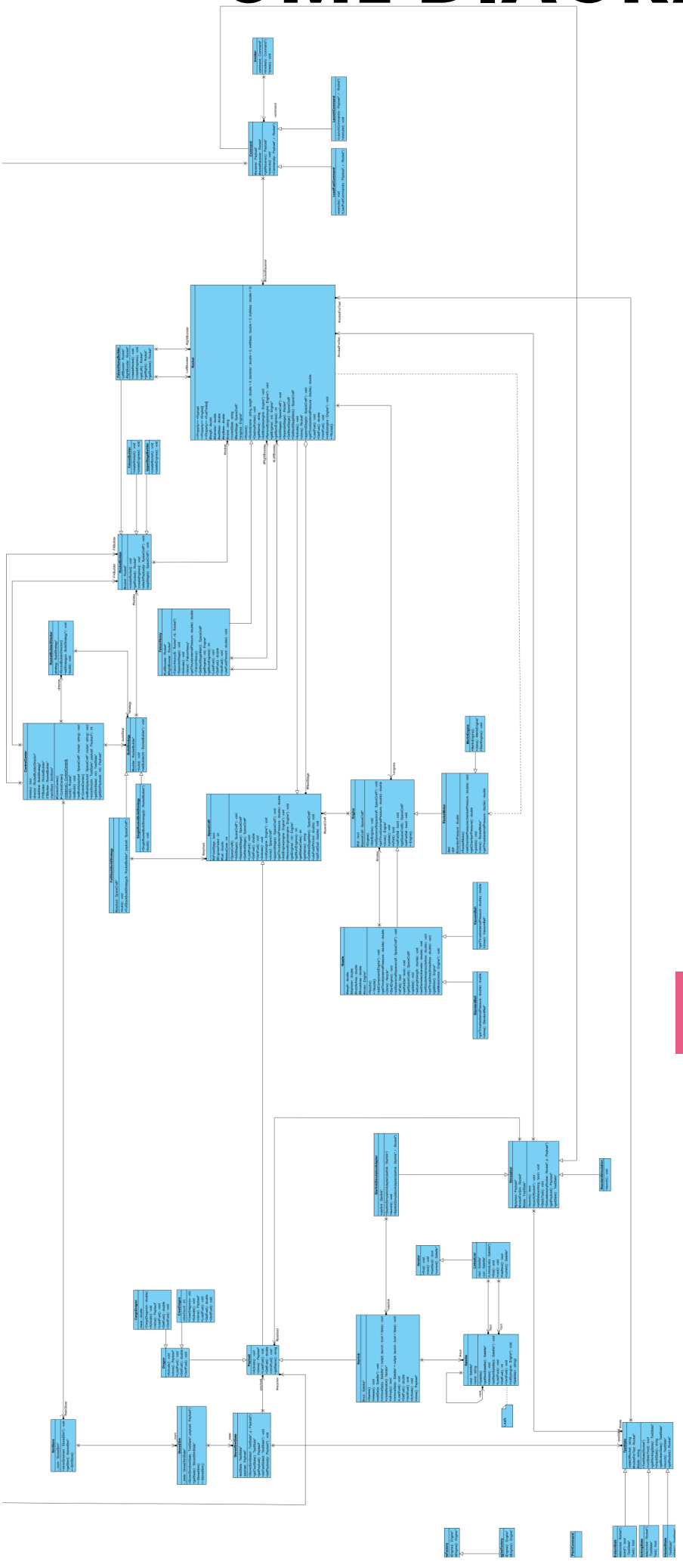


Template Diagram



Decorator Diagram

UML DIAGRAMS

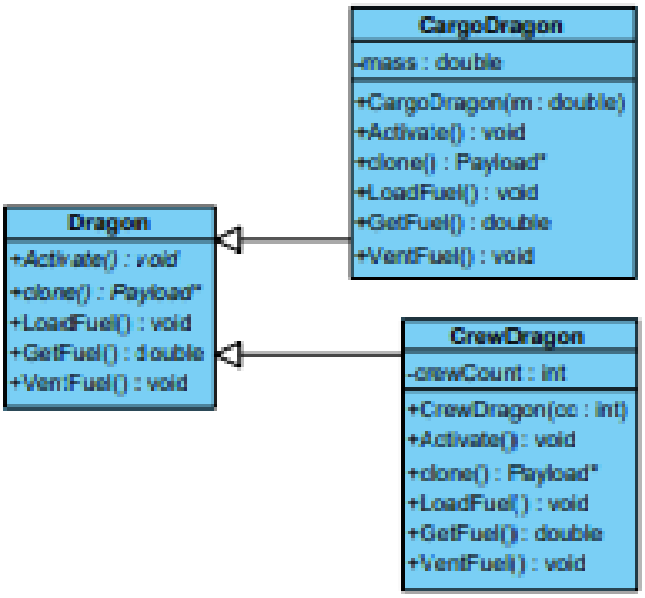


Main System

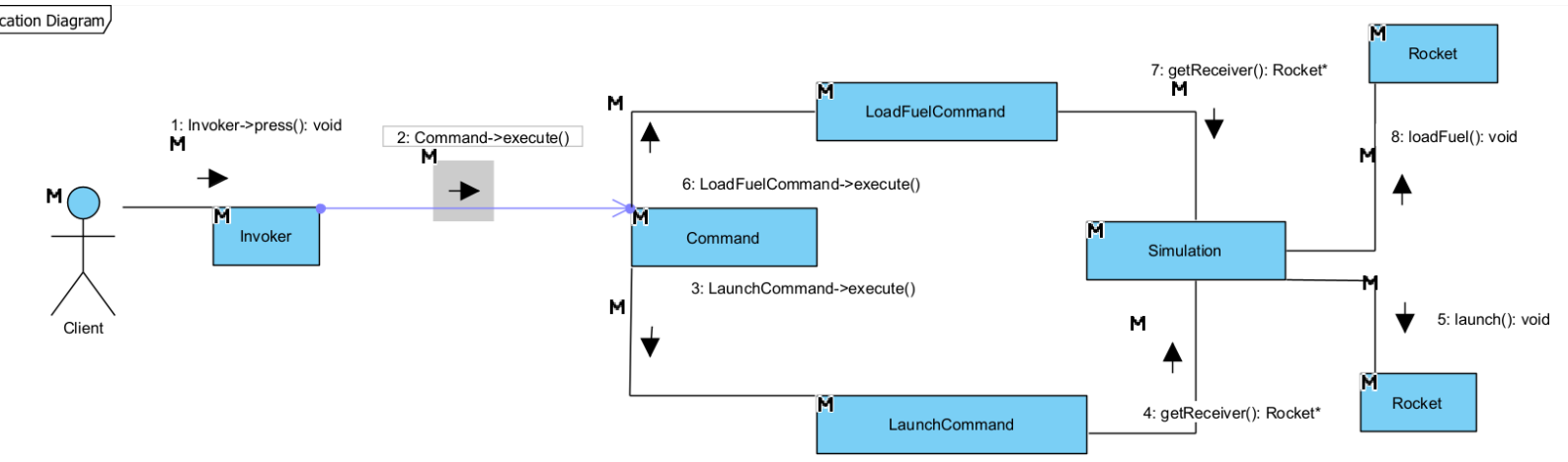
Magnificent 7
Report

UML DIAGRAMS

CONTINUES



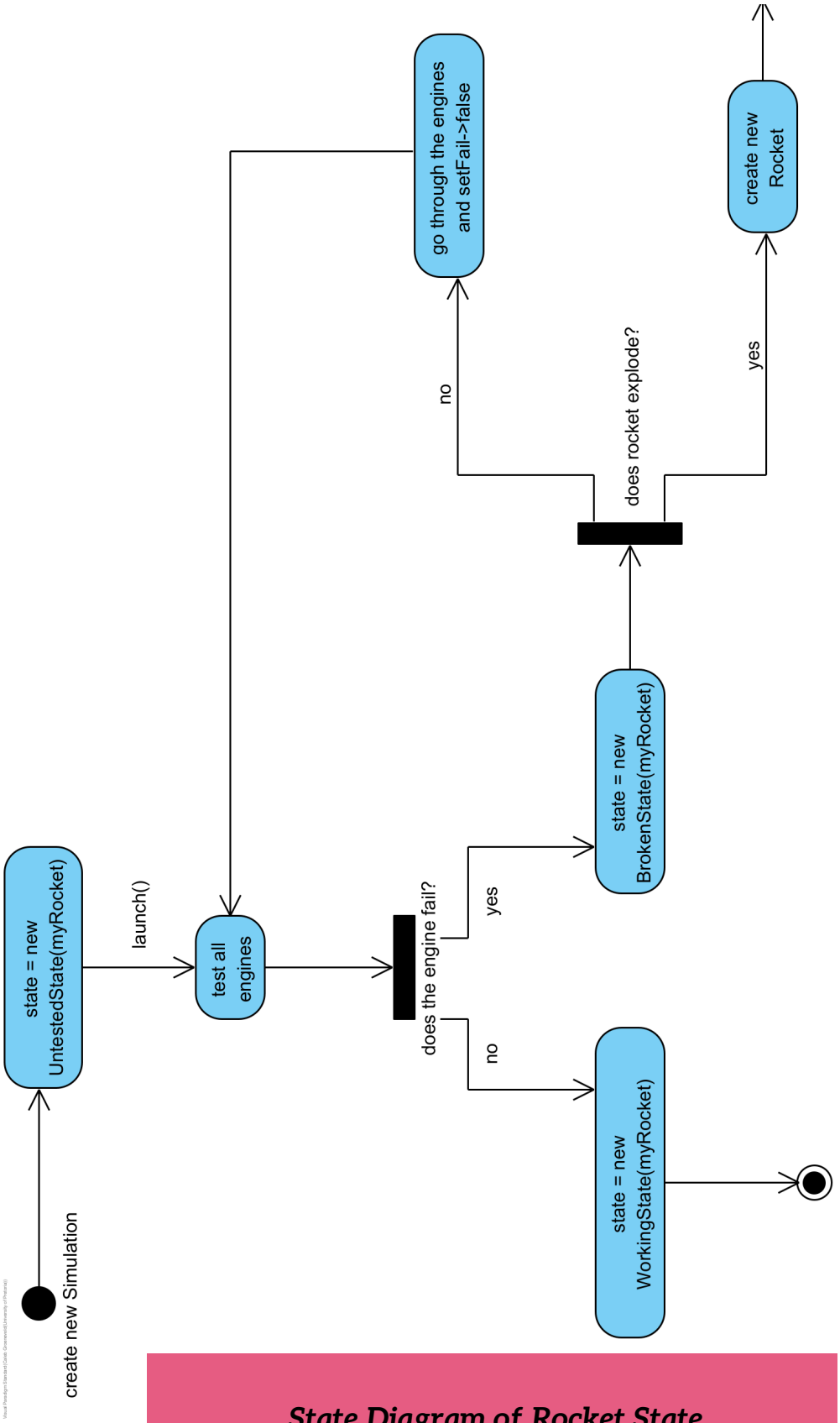
Object Diagram of Dragon object



Communication Diagram of Command object

UML DIAGRAMS

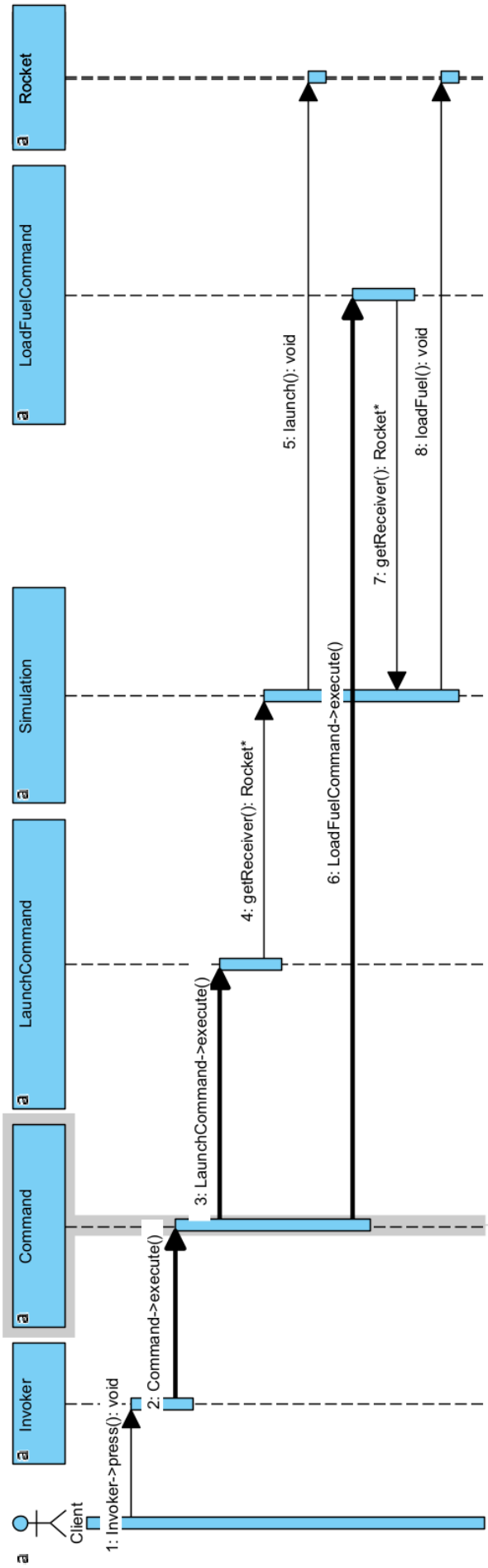
CONTINUES



State Diagram of Rocket State

UML DIAGRAMS

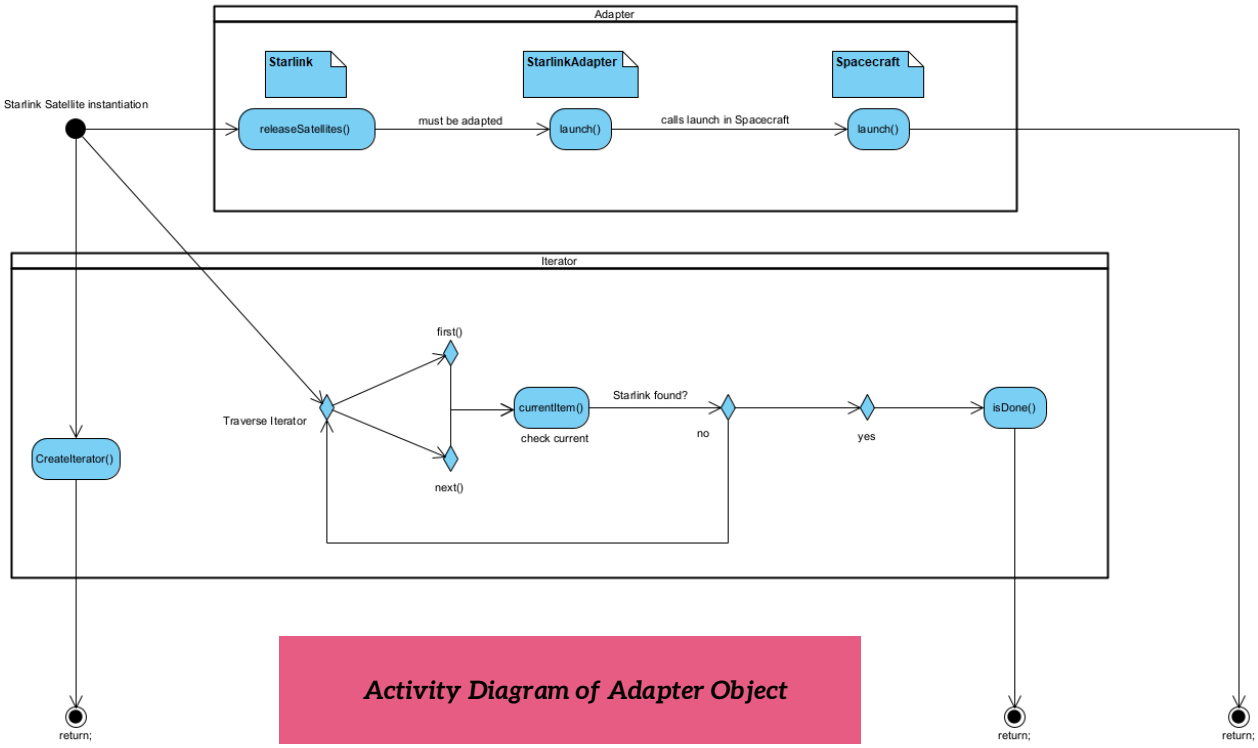
CONTINUES



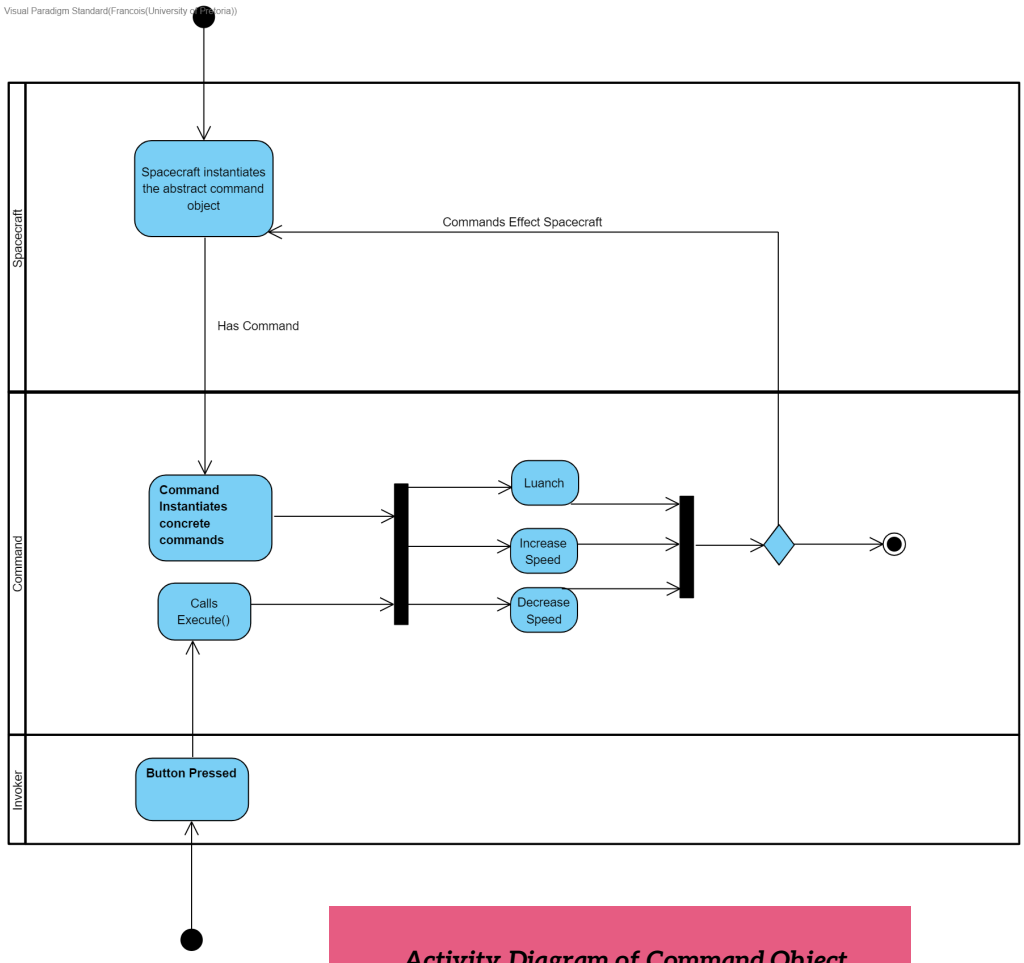
Sequence Diagram of Command Object

UML DIAGRAMS

CONTINUES



Activity Diagram of Adapter Object



Activity Diagram of Command Object

CONCLUSION

The point of the system is to demonstrate how design patterns can be used together to make a big system. Our system provides functionality to build a variety of pre-existing rockets as well as the capability of making custom and modular rockets. It caters for cargo, crew and even Starlink satellites. It offers post creational services such as a static fire testing system with repair states included. It comes pre-equipped with a state of the art launch facility and computer guided rockets that'll deliver your cargo at express speeds straight to the ISS at a modes to competitive fee of absolutely nothing!

ACKNOWLEDGEMENT

- Nozzle Design (2021). Available at: https://www.grc.nasa.gov/www/k-12/rocket/nozzle.html (Accessed: 22 November 2021).

- TECH SUMMARY (2021). Available at: http://www.rasaero.com/TECH_SUMMARY.htm (Accessed: 22 November 2021).

- Anon (2021) Web.stanford.edu. Available at: https://web.stanford.edu/~cantwell/AA284A_Course_Material/Karabeyoglu%20AA%20284A%20Lectures/AA284a_Lecture2.pdf (Accessed: 22 November 2021).

- Evolution of the SpaceX Merlin engine (2021). Available at: https://www.b14643.de/Spacerockets_2/United_States_1/Falcon-9/Merlin/index.htm (Accessed: 22 November 2021).

- Ideal Rocket Equation (2021). Available at: https://www.grc.nasa.gov/WWW/k-12/rocket/rktpow.html (Accessed: 22 November 2021).

- Specific Impulse (2021). Available at: https://www.grc.nasa.gov/WWW/k-12/airplane/specimp.html (Accessed: 22 November 2021).

- Rocket Thrust Equations (2021). Available at: https://www.grc.nasa.gov/WWW/k-12/airplane/rktthsum.html (Accessed: 22 November 2021).