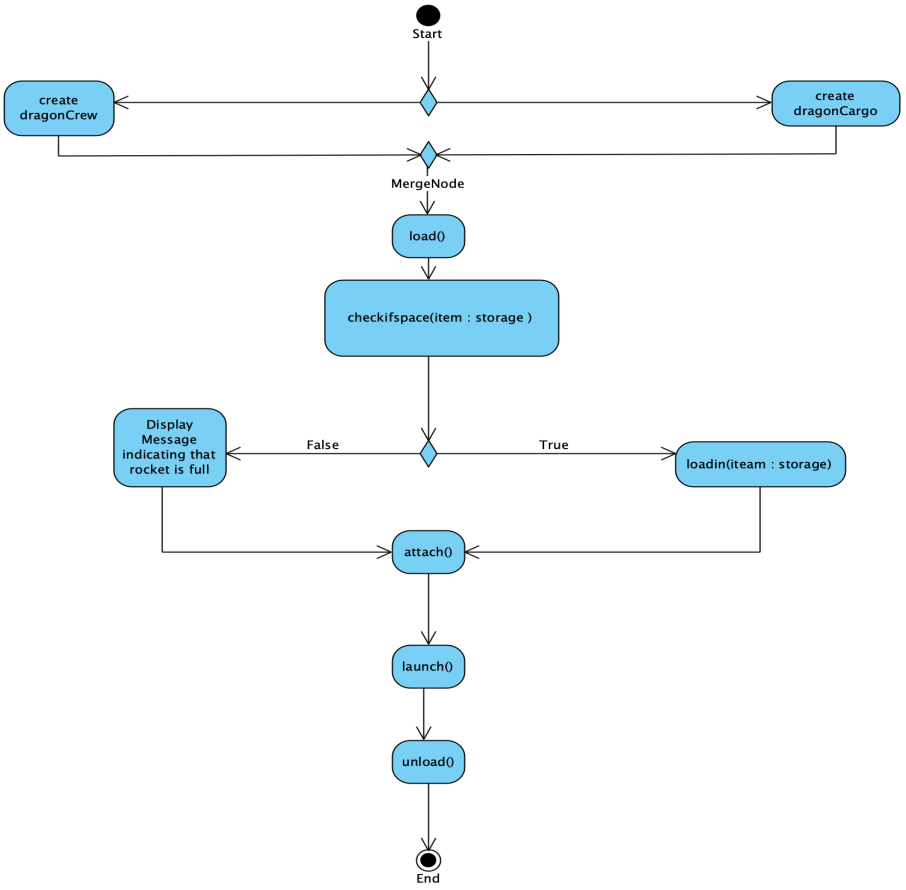
Task 1:

Task 1.1: The functional requirements

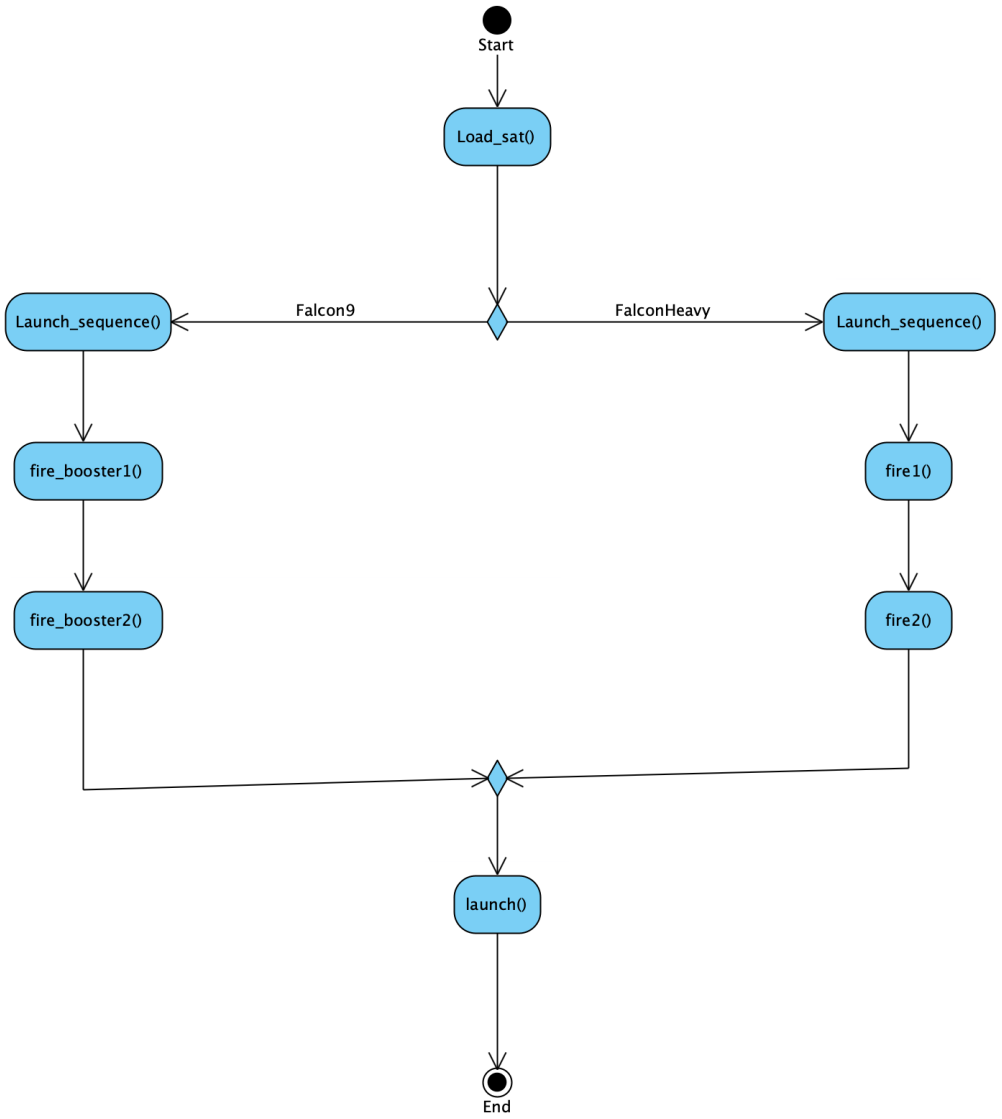
Functional Requirement No.	Function Requirement Description
FR 1	The user can create 2 types of rockets (Falcon and Dragon
FR 2	The user can create 2 sub-types of Falcon rockets with different launch sequences
FR 3	The user can create 2 sub-types of Dragon rockets with different load functions
FR 4	Give the Falcon rockets commands on what to do
FR 5	Give the Dragon rockets commands on what to do
FR 6	Check to see if the ISS is docked to a rocket or not
FR 7	Memento to create a saved state to use later and undo changes
FR 8	Store the state of the mediator for other simulations
FR 9	Observe the satellites to check if they are online
FR 10	Iterate through all the satellites
FR 11	Be able to let all the other satellites know that one has gone offline and thus the system will be offline.

Task 1.2: Activity diagrams

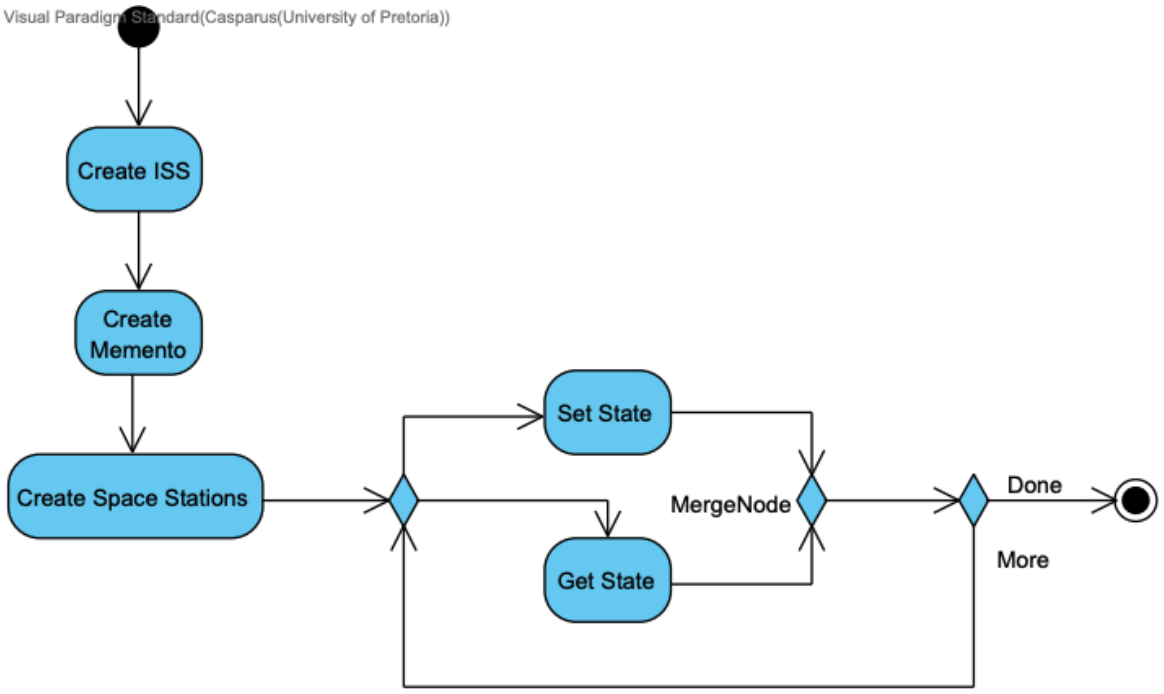
Template

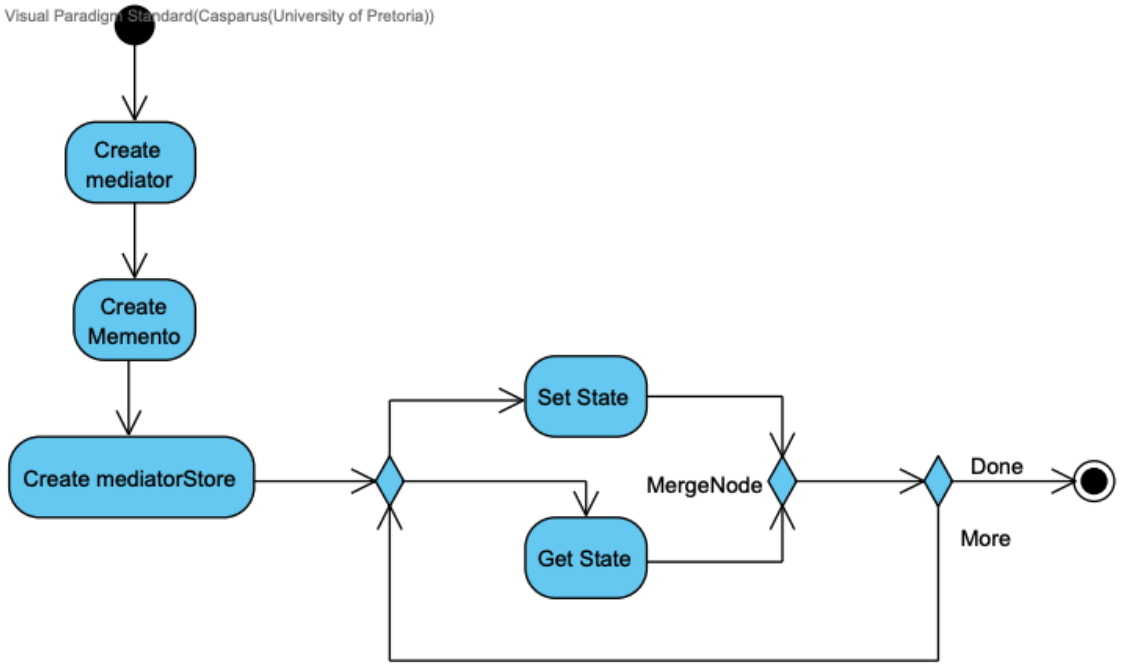


Strategy

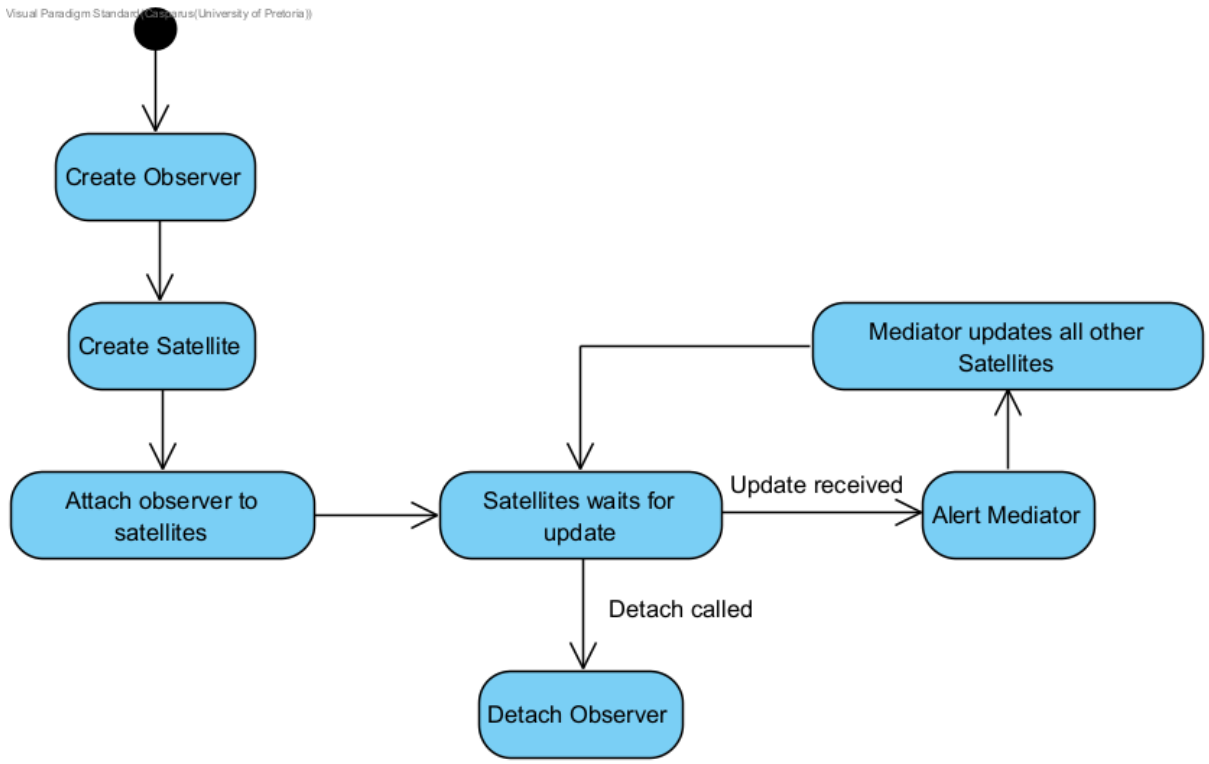

Start
Load_sat()
Launch_sequence()
Falcon9
FalconHeavy
Launch_sequence()
fire_booster1()
fire_booster2()
fire1()
fire2()
launch()
End

Memento

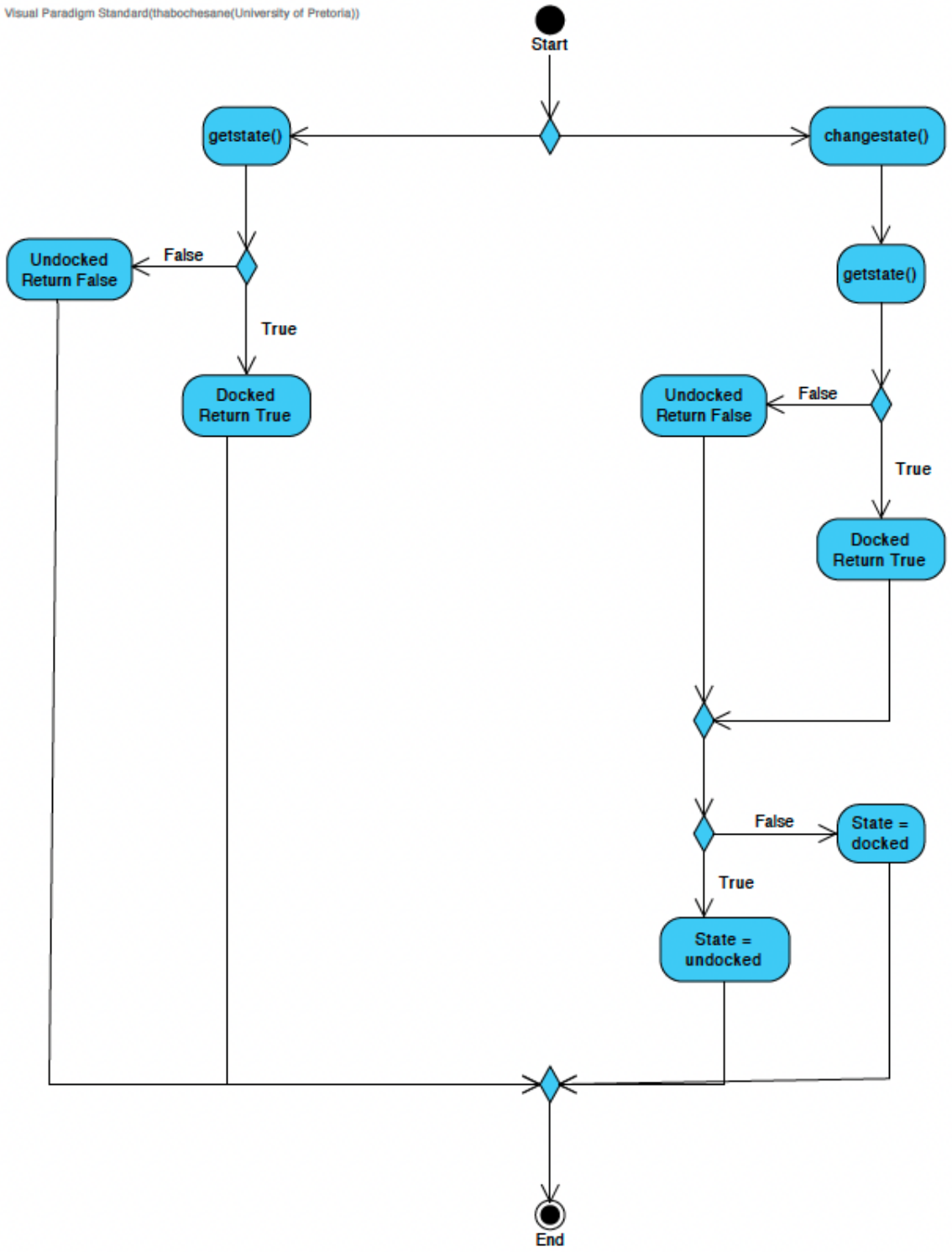

Visual Paradigm Standard(Casparus(University of Pretoria))
Create ISS
Create Memento
Create Space Stations
Set State
Get State
MergeNode
Done
More


Visual Paradigm Standard(Casparus(University of Pretoria))
Create mediator
Create Memento
Create mediatorStore
Set State
Get State
MergeNode
Done
More

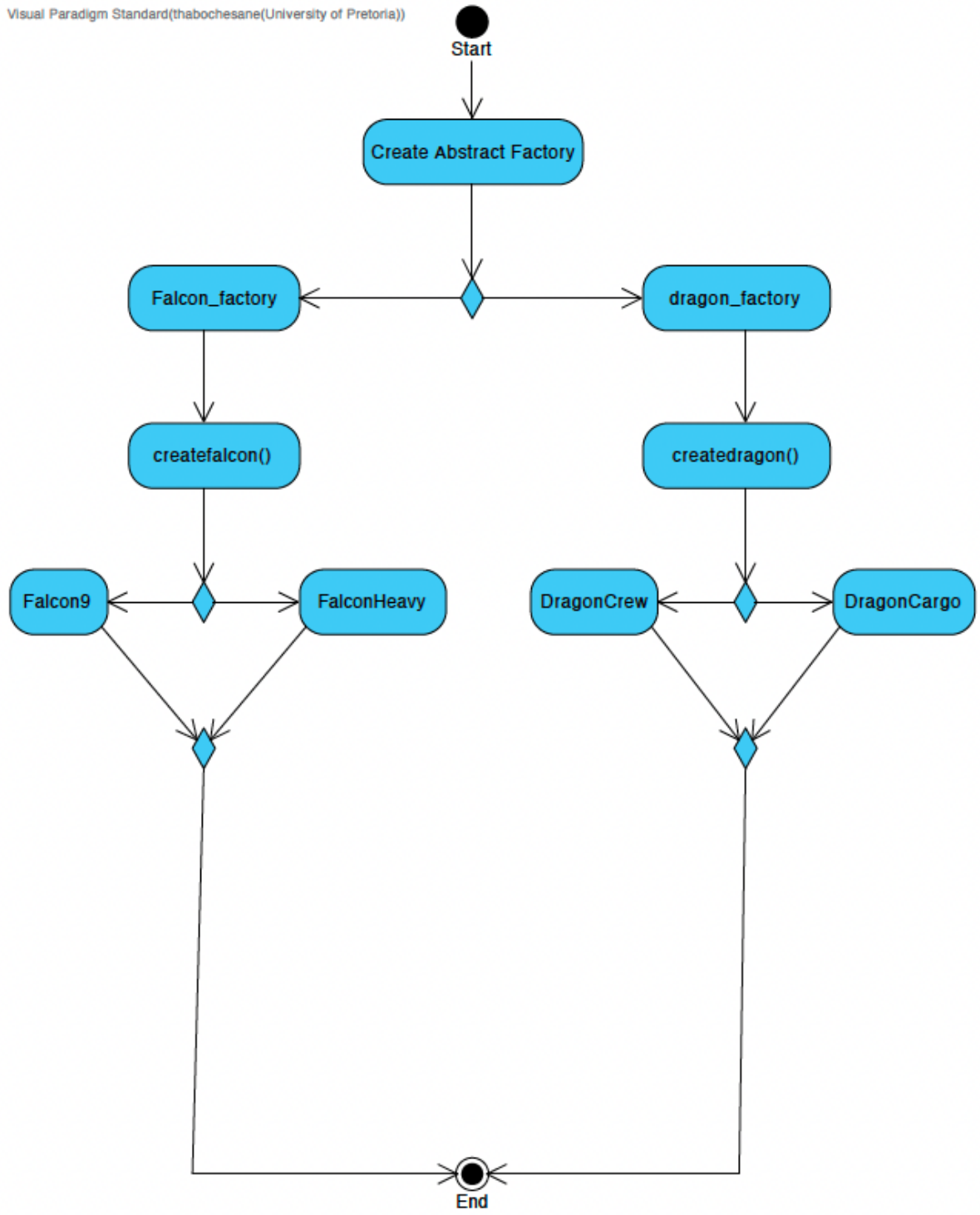
Observer


Visual Paradigm Standard(Casperus(University of Pretoria))
Create Observer
Create Satellite
Attach observer to satellites
Satellites waits for update
Update received
Alert Mediator
Mediator updates all other Satellites
Detach called
Detach Observer

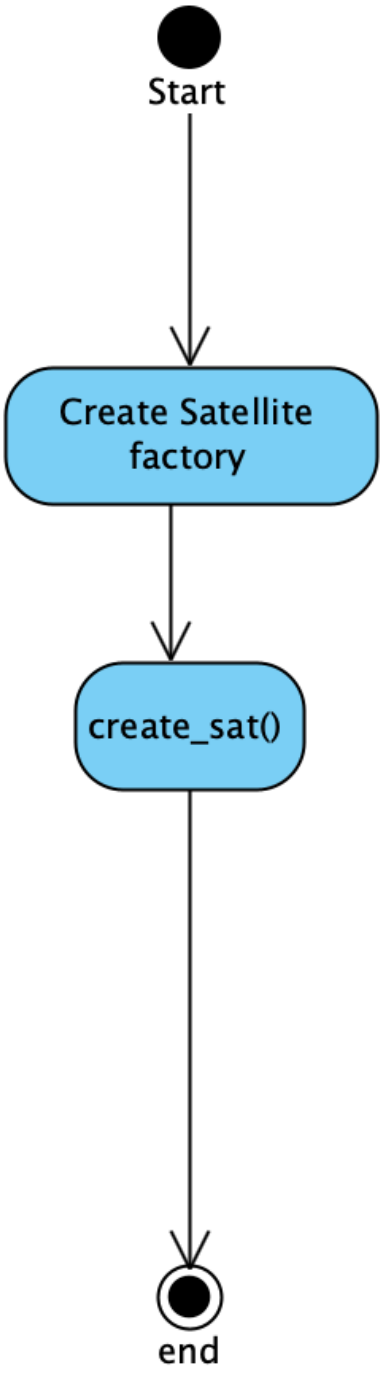
State


Visual Paradigm Standard(thabochesane(University of Pretoria))
Start
getstate()
changestate()
Undocked
Return False
False
True
Docked
Return True
getstate()
Undocked
Return False
False
True
Docked
Return True
False
State =
docked
True
State =
undocked
End

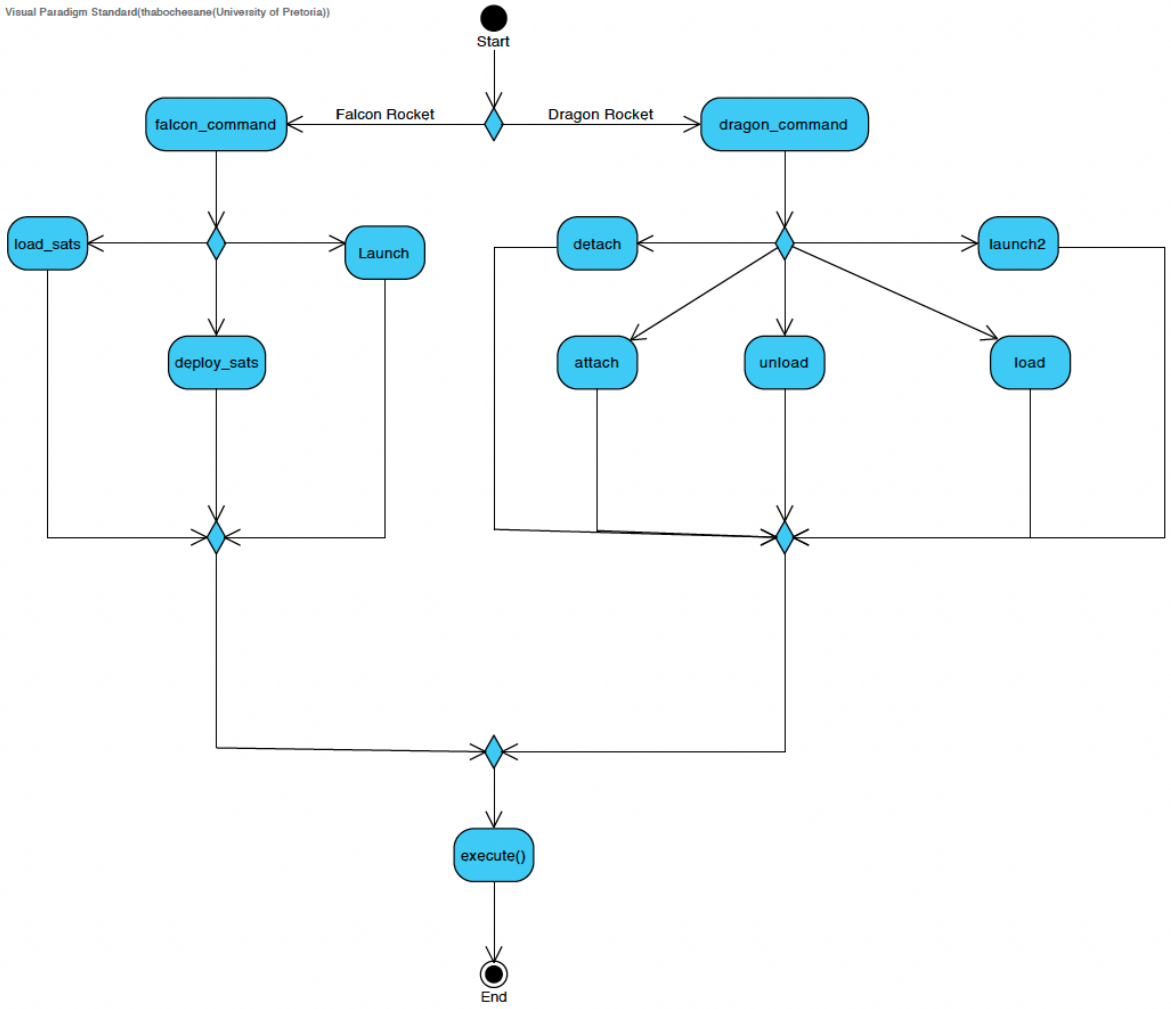
Abstract Factory



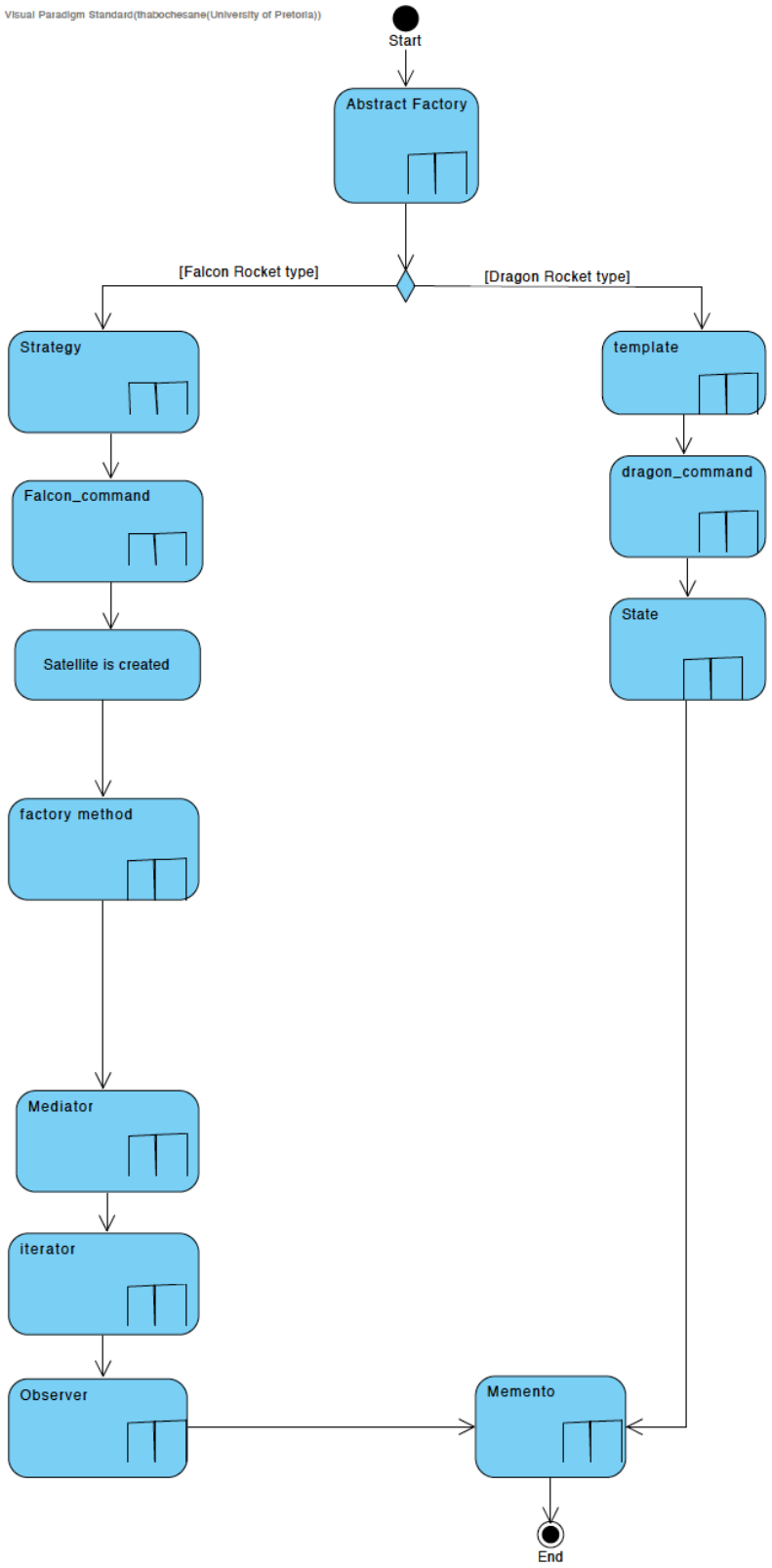
Factory



Command


Visual Paradigm Standard(thabochesane(University of Pretoria))
Start
falcon_command
Falcon Rocket
Dragon Rocket
dragon_command
load_sats
Launch
detach
launch2
deploy_sats
attach
unload
load
execute()
End

Final Activity diagram



Task 1.3: Patterns to address the functionality defined by the functional requirements and processes

- Abstract factory: The ability to create 2 types of rockets(falcon and dragon).
- Strategy: The ability to create 2 subtypes of Falcon rockets with different launch sequences.
- Template: The ability to create 2 subtypes of Dragon rockets with different load functions.
- Command: The ability to give the Falcon rockets and Dragon rockets commands on which sequence to follow.
- State: The ability to check if the ISS is docked or undocked to a spaceship.
- Memento: Be able to store the ISS as well as being able to store the state of the rockets for other simulations.
- Prototype: Need to have a default satellite prototype.
- Factory: Use the prototype to create additional satellites.
- Observer: We need to observe the state of the satellites to check if they are online and functional.
- Iterator: Be able to iterate through all of the satellites.
- Mediator: Be able to have a mediator which lets all the other satellites know. that one has gone offline and thus the system will be offline.
- Memento: Be able to save the state of the mediator for other simulations.

Task 1.4: Classes for each of the identified patterns taking their interrelationships into account

STRATEGY: strategy, Falcon9, FalconHeavy.

TEMPLATE: template, DragonCrew, DragonCargo, storage, cargo, crew.

MEMENTO: ISSMemento, ISS, space_stations.

MEMENTO: mediatorMemento, mediator, mediators.

MEMENTO: rocketMemento, rockets, rocket.

STATE: ISS, docked_state, docked, undocked.

PROTOTYPE: prototype, satellite factory, satellite.

ITERATOR: iterator, aggregate, satellite.

FACTORY: satellite factory, satellite, prototype.

ABSTRACTFACTORY: falcon_factory, dragon_factory, Falcon9, FalconHeavy, DragonCrew, DragonCargo.

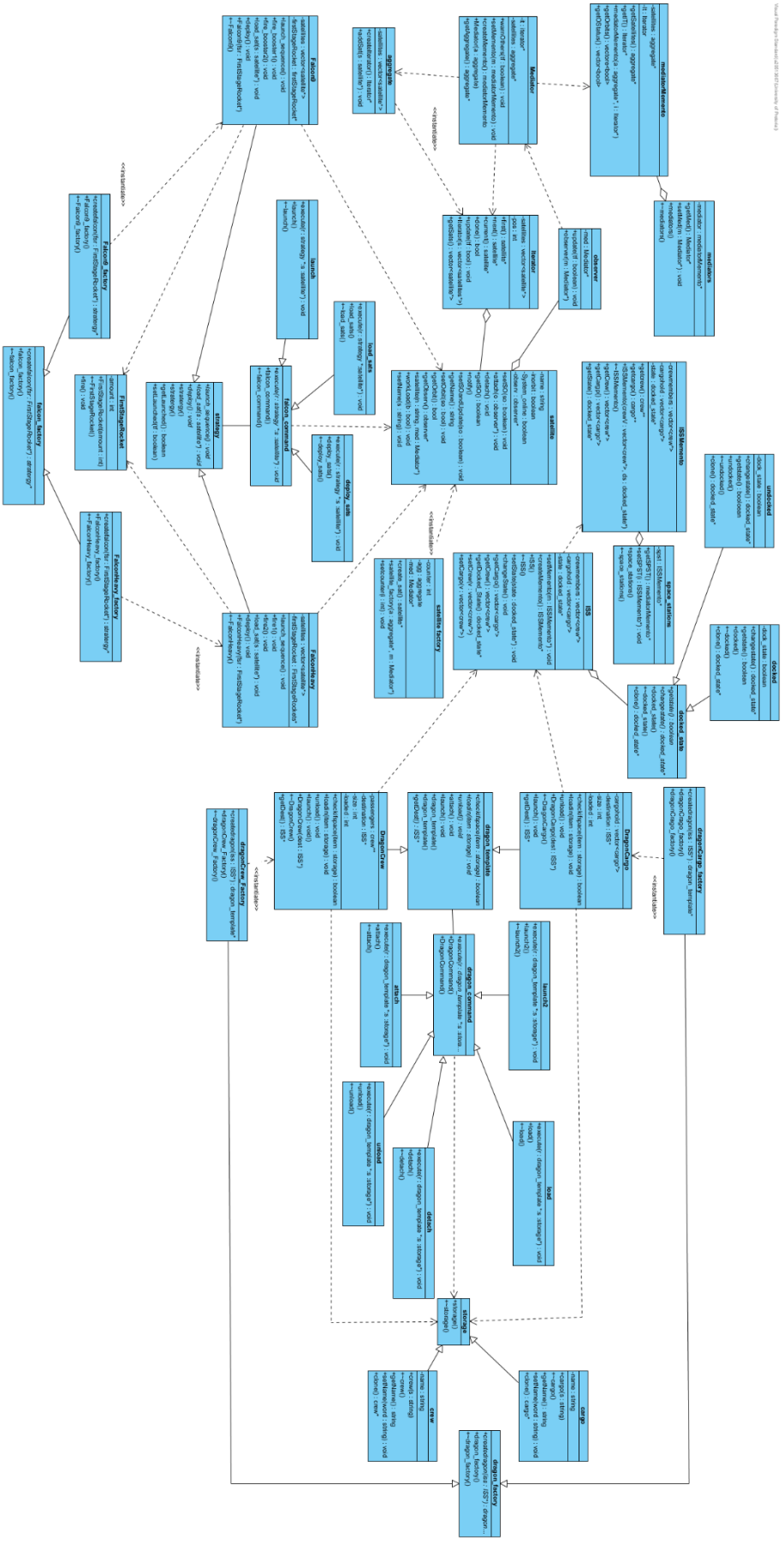
MEDIATOR: mediator, iterator, mediatorMemento, observer.

OBSERVER: observer, satellite, mediator.

COMMAND: dragon_command, launch2, load, attach, unload, strategy, storage.

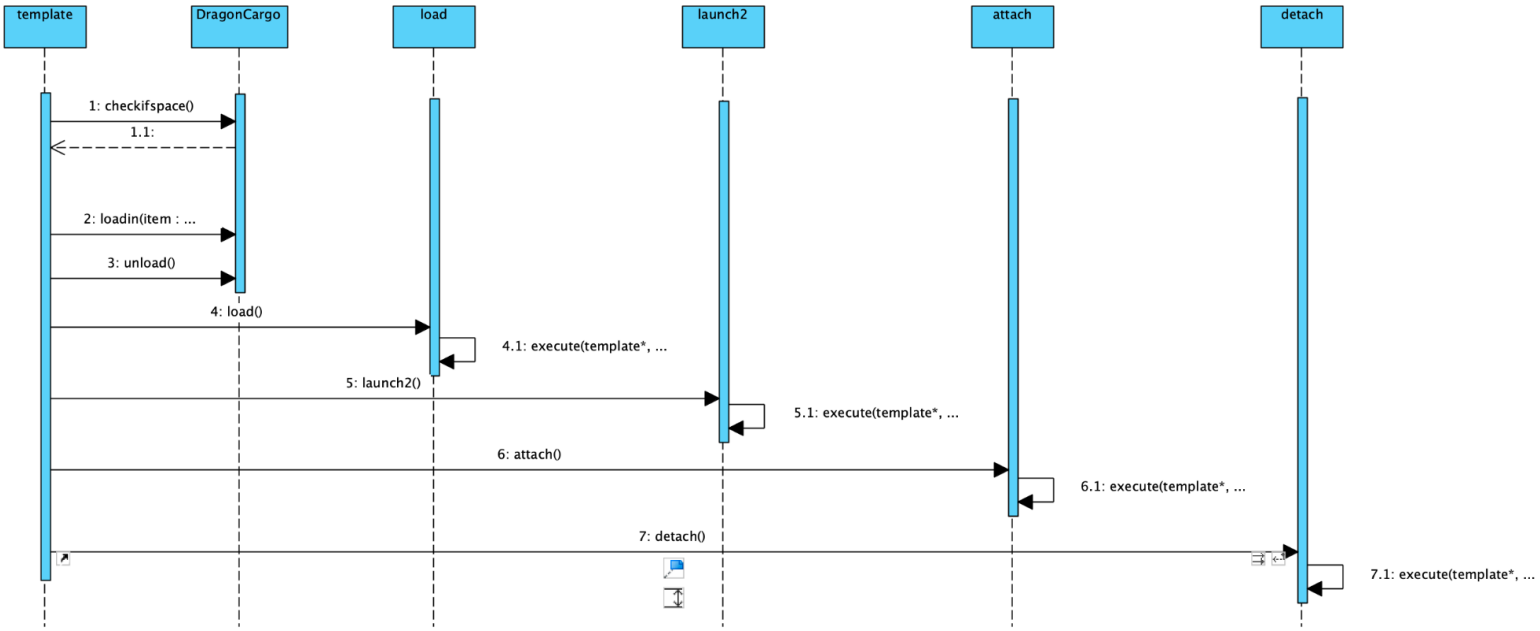
COMMAND: falcon_command, launch, load_sats, deploy_sats, template, satellite.

Task 1.5: Class diagram of our system

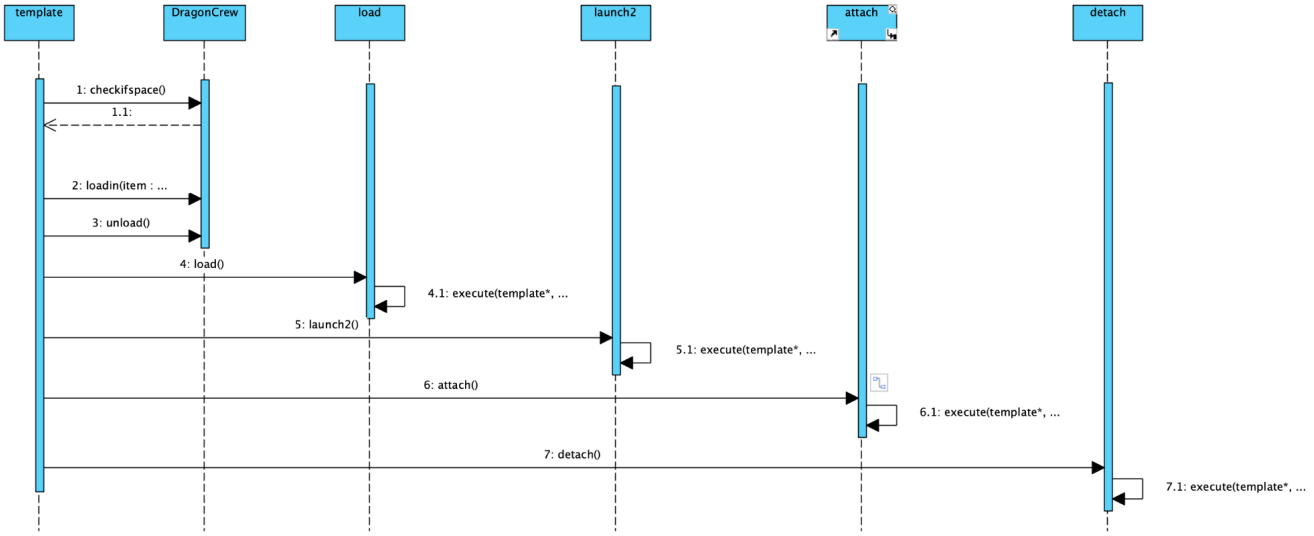


Task 1.6: Sequence and communication diagrams(Not complete)

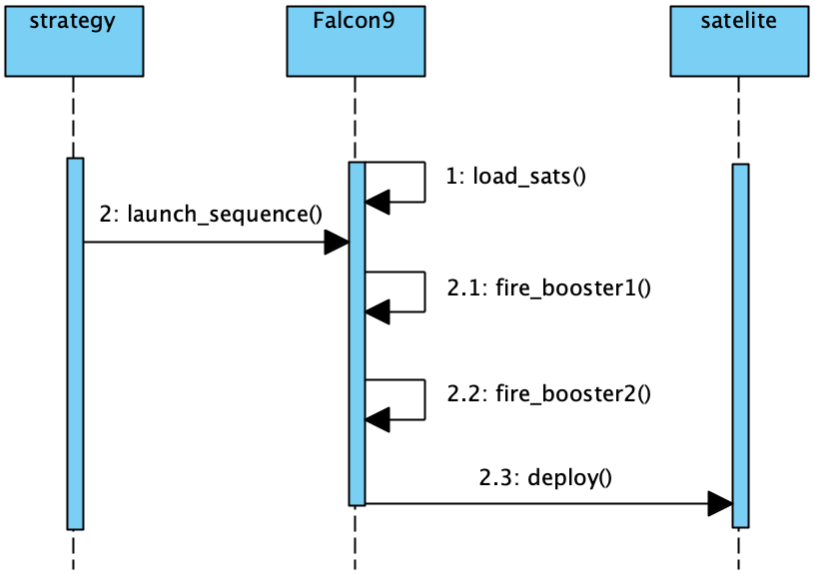
DragonCargo



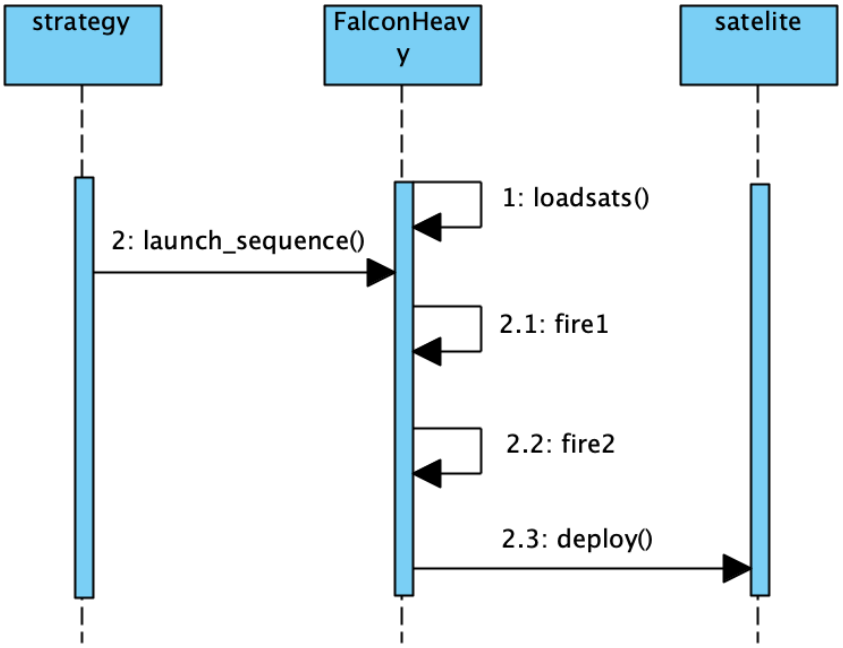
DragonCrew

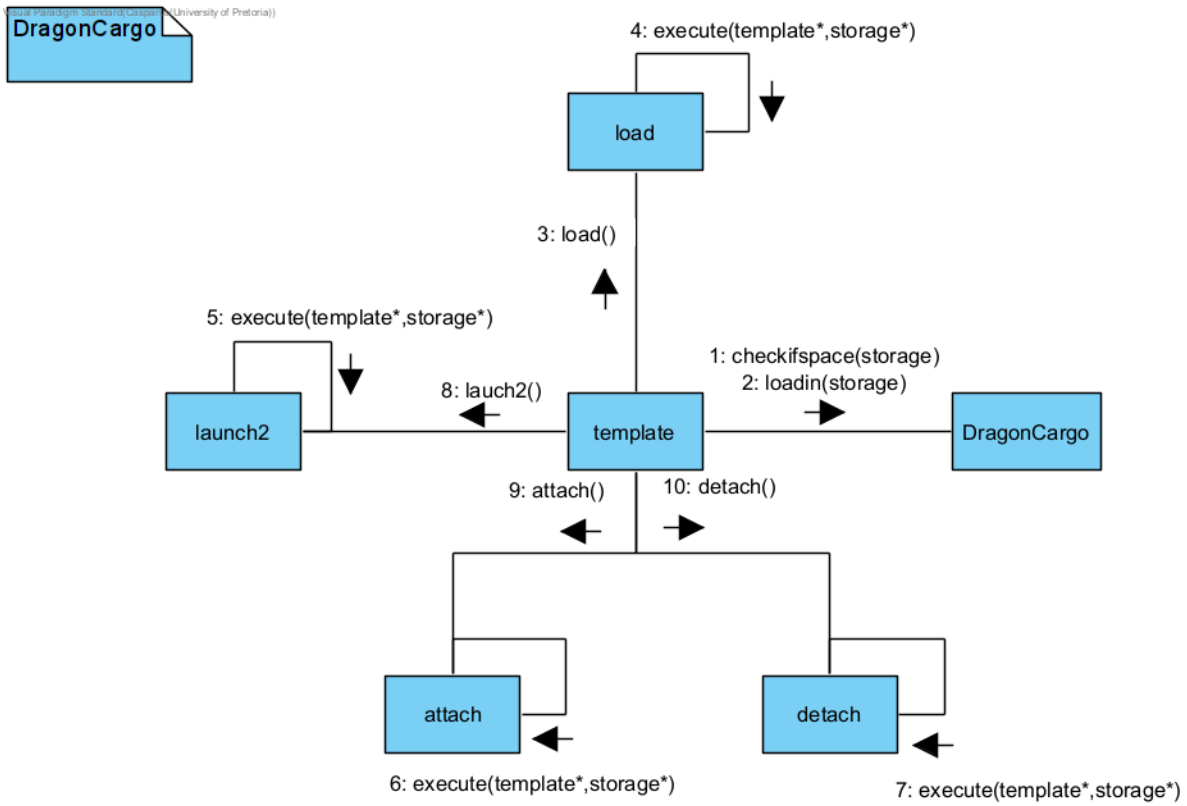


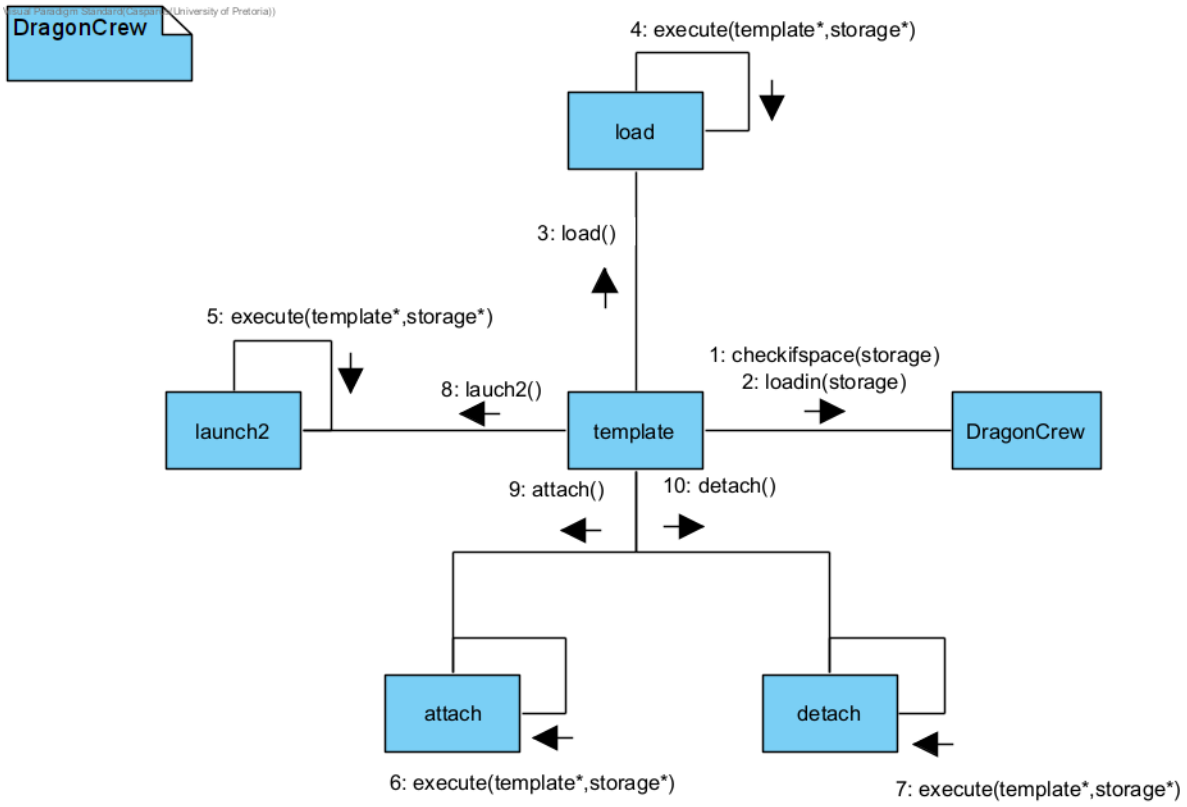
Falcon9

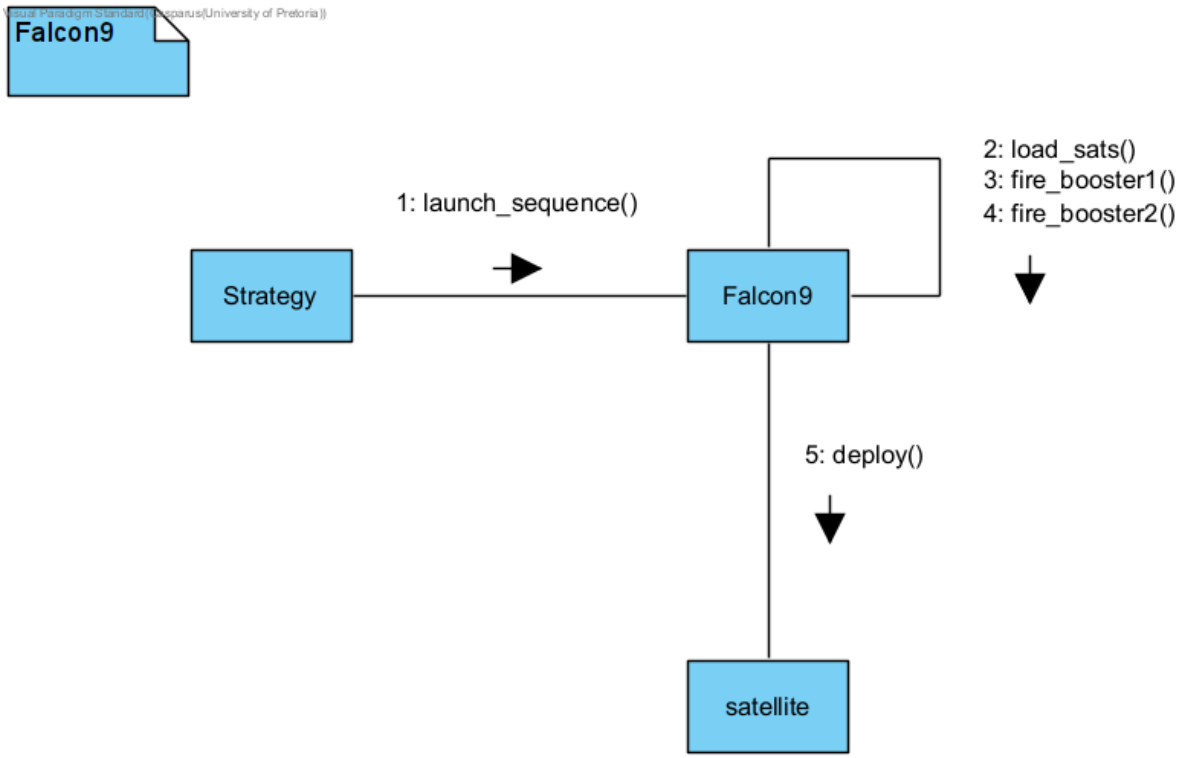


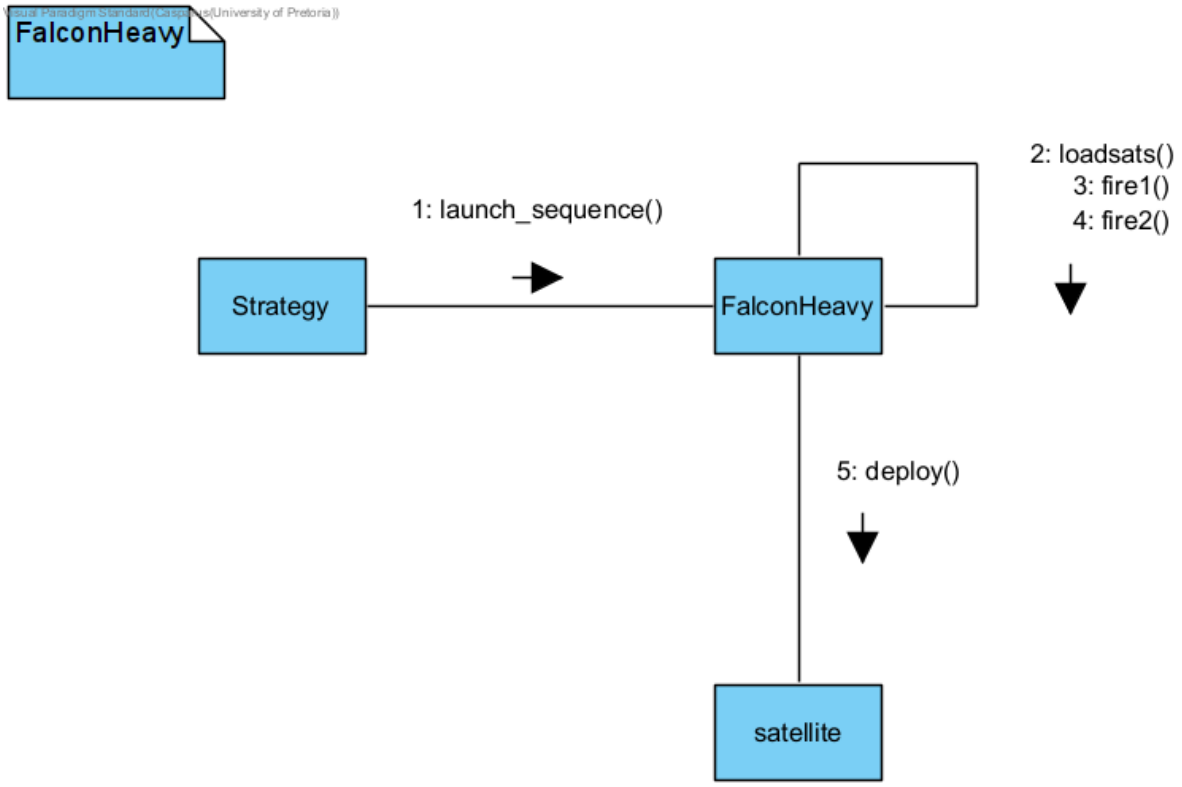
FalconHeavy



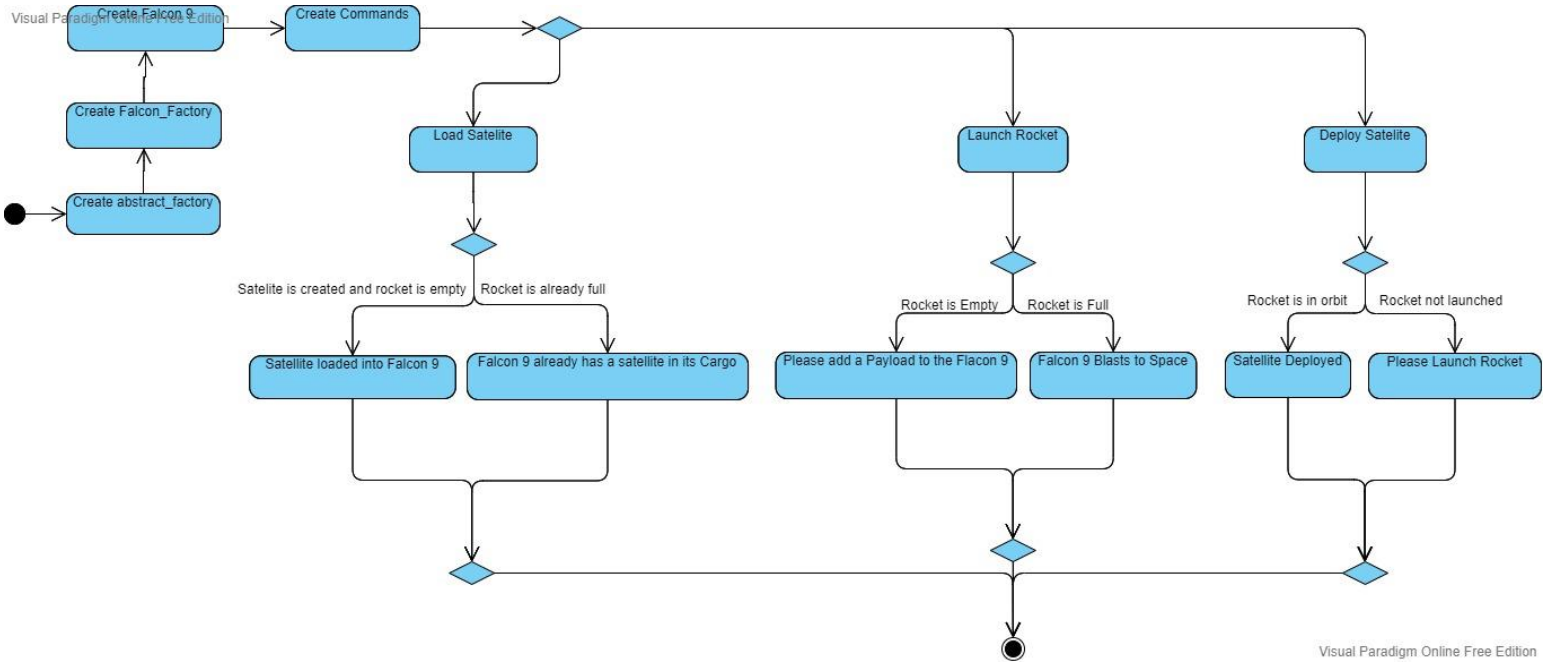

DragonCargo
4: execute(template*,storage*)
load
3: load()
5: execute(template*,storage*)
1: checkifspace(storage)
2: loadin(storage)
8: lauch2()
launch2
template
DragonCargo
9: attach()
10: detach()
attach
detach
6: execute(template*,storage*)
7: execute(template*,storage*)

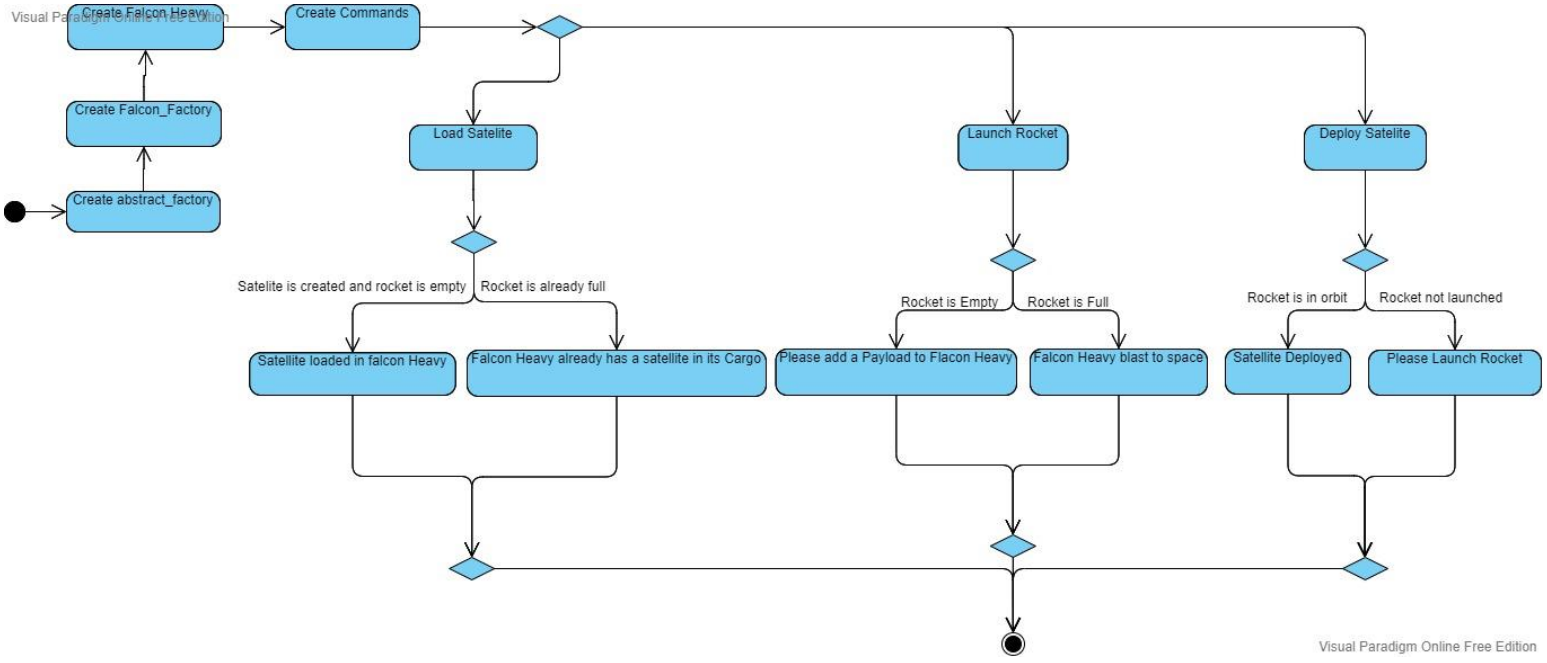

DragonCrew
4: execute(template*,storage*)
load
3: load()
5: execute(template*,storage*)
1: checkifspace(storage)
2: loadin(storage)
8: lauch2()
launch2
template
DragonCrew
9: attach()
10: detach()
attach
detach
6: execute(template*,storage*)
7: execute(template*,storage*)


Falcon9
1: launch_sequence()
2: load_sats()
3: fire_booster1()
4: fire_booster2()
Strategy
Falcon9
5: deploy()
satellite

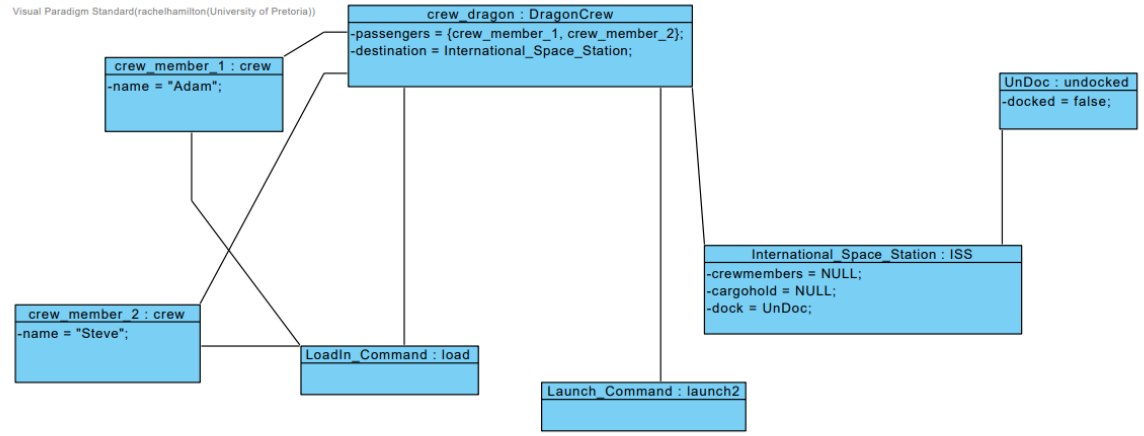

FalconHeavy
1: launch_sequence()
2: loadsats()
3: fire1()
4: fire2()
Strategy
FalconHeavy
5: deploy()
satellite

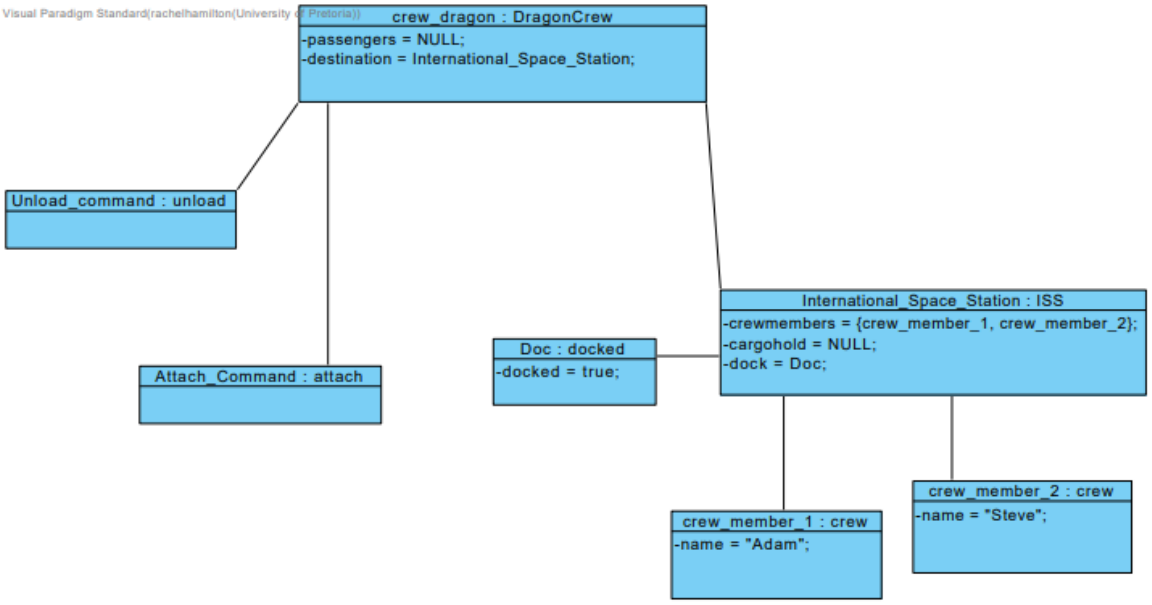
Task 1.7: State diagram to show how an object changes state





Task 1.8: Two object diagrams showing the state of the objects active in the simulation just before launch and then again when docked at the International Space Station





Link to this document:
https://docs.google.com/document/d/1_2gOOdovvoOTH74Ds7AR3uAke4EfVRdMjXNMN5mqTvA/edit?usp=sharing