8 November 2022

WAR SIMULATOR REPORT

404 Not Found

COS 214 PROJECT

Shiluvelo Makhubele	u19086352
Christof Steyn	u17021074
Charl Volschenk	u17053928
Rebecca Oosthuizen	u20512008
Chris Mittendorf	u15092462
Akani Hlungwani	u19240202

Introduction

In this project, we were tasked to design and construct a war simulator software system using the design patterns identified by Gamma et al [1] .

Research Findings

We found the following information through research and pre-existing knowledge about warfare:

Warfare

Warfare concerns two or more opposing parties engaging in armed conflict with one another. Warfare occurs for one or more of a variety of reasons.

War Theatre

Inspired by von Clausewitz' definition of "theater of war" [3] , we define a war theatre as an area in which a battle between alliances takes place. The area type can be airspace, land, or sea, which affects the type of unit that can be deployed into the theatre. For example, airplanes can only be used in an airspace.

Entities

There are various entities belonging to countries that can be deployed into a war theatre to engage in warfare. The entities we will use in our implementation are as follows:

Person:
A person can be a soldier (actively participates in battle by causing and taking damage), a medic (heals wounded soldiers), or a mechanic (repairs damaged vehicles).

Vehicle:
A vehicle can be a tank (deployed in land war theatres), a plane (deployed in airspace war theatres), or a ship (deployed in sea war theatres).

Phases of War

A war itself can be broken into a variety of phases. We have decided to consider the following phases, inspired by Bloomfield [2], in our implementation:

Phase 1: Dispute
A dispute takes place between countries.

Phase 2: Conflict
The countries involved in the dispute form alliances.

Phase 3: Fighting
In this phase there is direct conflict between the alliances.

Phase 4: Post-Conflict
Direct fighting between alliances stops but the conflict has not yet ended. Negotiations between alliances take place in an attempt to end the conflict.

Phase 5: Diplomacy
This is the final phase of war where the dispute is settled.

The United Nations

The UN [4] is an international organisation which 193 member states form part of. It is committed to maintaining international peace and security and will thus foster negotiations in our system.

Changes in War

Countries and alliances may change their approach to warfare as time progresses. The changes that countries and alliances are able to make in our system are as follows:

Alliance Adjustment:
Countries may choose to join an alliance, leave an alliance, or change alliances.

Research and Development:
A country or an alliance may choose to spend additional resources on research and development in order to gain extra entities or to improve their existing entities.

War Treaties:
Countries or alliances may adopt war treaties, such as a no-fire treaty while negotiations are taking place.

Tactic Adjustment:
Countries may adjust how aggressively they respond to provocation or attack. For example, if a country has depleted resources, another alliance may choose not to attack it.

Application

The above research findings will be implemented and use design patterns [1] in the system in the following ways:

Warfare

Our system will model various elements of a turn-based war taking place between two opposing alliances or country groups. The reasons for war will not be taken into consideration in our simulation. We will only model the occurrences within actual warfare.

Simulator Mode

The user can run the program in two different modes:

Design mode: The program will accept all inputs from the user and will allow the user to interrupt the simulation, tweak it, and continue.

Real mode: The simulation will be set up and run uninterrupted.

The Memento Design pattern will be used to save the state of the entire simulation upon user interrupts in design mode. This is in case the user wishes to pause the program or if there is a system crash so that the saved state can be restored.

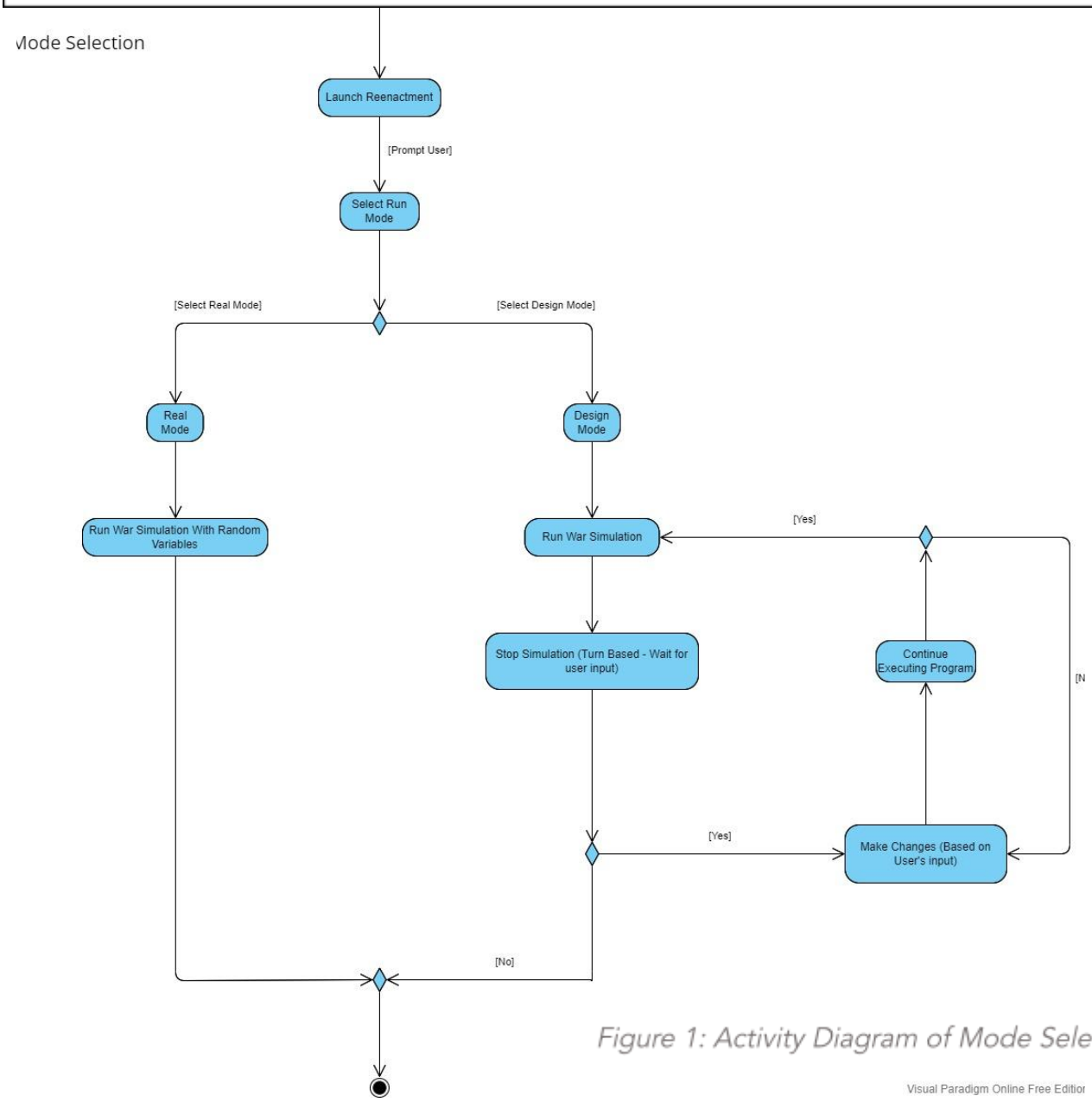


Figure 1: Activity Diagram of Mode Selection

War Engine

The main driver of our simulation will be an event loop referred to as the war engine. This loop contains a country group's 'turn' (consisting of attacking another country and resource management) and will continue until the diplomacy phase is reached.

> The Singleton design pattern will be used to create a single instance of the war engine as there should only be one instance of this class.

War Theatre

The war theatre will be classified as a land area, sea area, or mountain area, which will dictate the types of units that may enter the theatre. Land area will consist of air units and land units, sea area will consist of air units and navy units and lastly mountain area will consist only of air units. The war theatre will have a list of the alliances at battle, as well as an attacking squad and a defending squad. Each list of units in the squad will have different strategies on how they deploy.

> The attacking and defending squads will be built using the Builder design pattern, so that it is easy for the WarTheatre class to build squads for the various types of areas.

> The Strategy design pattern will be used by the WarTheatre class to implement the different deploy strategies for the different types of units.

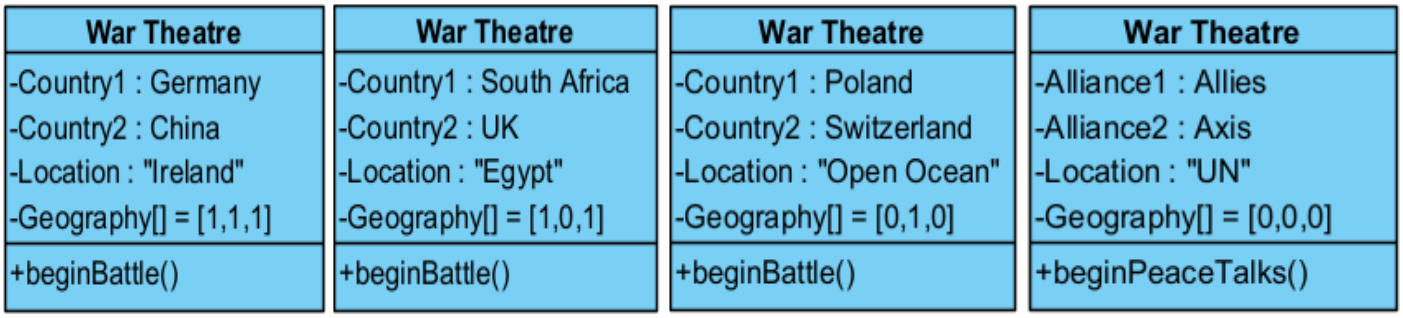
Participants for Builder design pattern

GOF participant	Our participant	Goal
Client	WarTheatre	Uses ConcreteSquadBuilder(ConcreteBuilder) and ArmyGeneral(Director) to build an appropriate Squad(Product), that is appropriate for the area('land','sea' or 'mountains')
Director	ArmyGeneral	Uses the SquadBuilder interface to direct a ConcreteSquadBuilder on how to build the different types of squads
Builder	SquadBuilder	Defines the base interface that can be used by the ConcreteSquadBuilders, to add ground and/or air and/or navy units to the squad, based on the area
ConcreteBuilder	ConcreteSquadBuilder	Owns a Squad Assembles parts of the squad by implementing the SquadBuilder interface Has a method for retrieving the Squad
Product	Squad	Defines and implements the squads that arrive at a WarTheatre. Has lists for all the types of units that arrive at the WarTheatre(Soldier, Tank, Ship, Plane, Medic, Engineer).

		Also has attributes that keep track of the squad's performance in the battle(eg: totalSoldiersKilled)

Participants for Strategy design pattern

GOF participant	Our participant(s)	Description
Context	DeployContext	Calls different strategy implementations based on the ConcreteStrategy that it uses
Strategy	DeployType	Defines the interface that ConcreteStrategies should implement
ConcreteStrategy	SoldierDeploy, TankDeploy, ShipDeploy, AirDeploy, MedicDeploy, MechanicDeploy	Each differently implements what the units do that they deploy. Soldiers can only fight other soldiers Tanks choose between fighting other tanks or taking out multiple soldiers at once Ships will fight other ships Planes can fight any type of unit, if it chooses soldiers, then it carpet bombs multiple soldier at once Medics will remove heavily wounded soldiers from the fight and heal the wounded soldiers Mechanics will repair damaged tanks



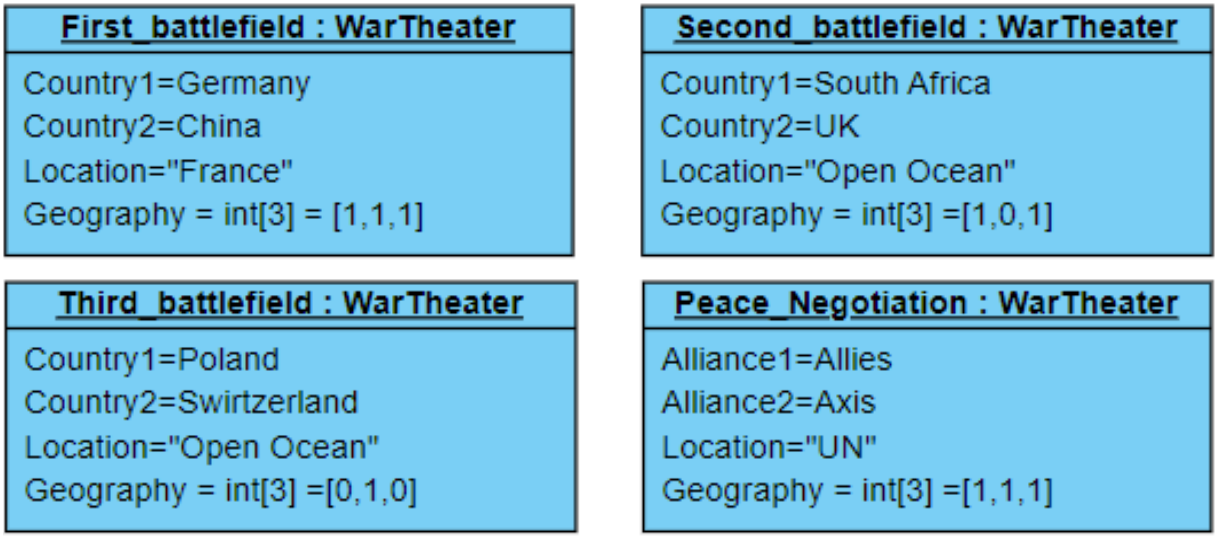


Figure 2: War Theatre Object Diagram During Phase 3 and Phase 4

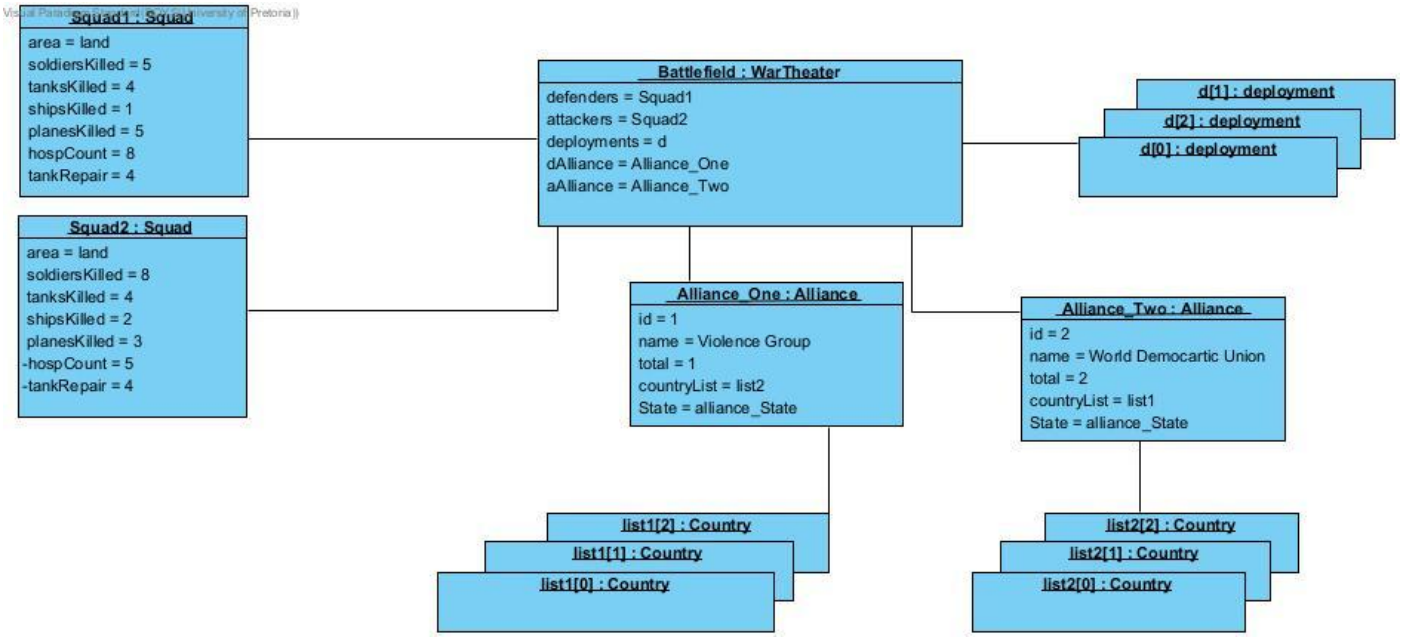


Figure 3: War Theatre Object Diagram

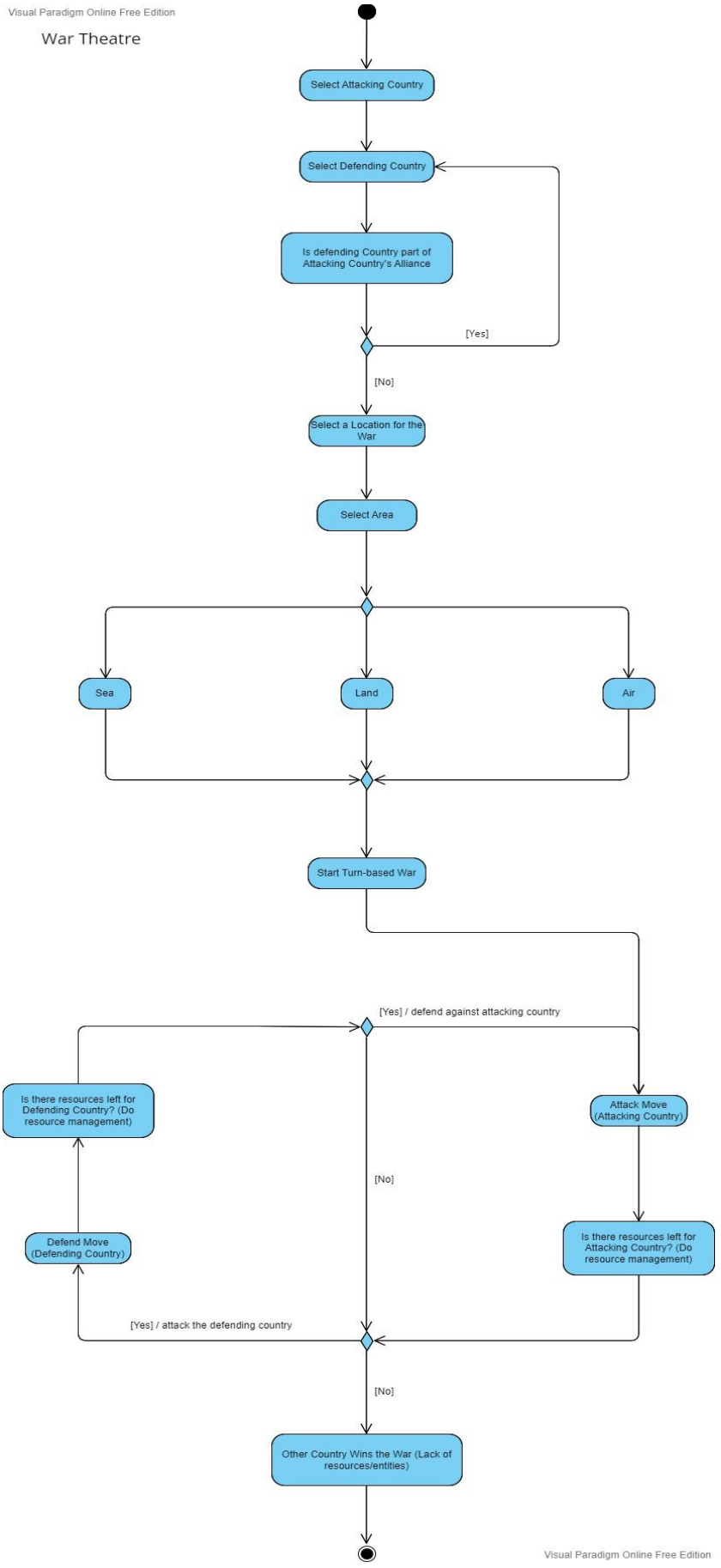


Figure 4: War Theatre Activity Diagram

Entities

Each country will have its own set of entities, including people (soldiers, mechanics, and medics) and vehicles (tanks, ships, and planes). In a war theatre, the entities of opposing groups compete against one another. Vehicles can only cause damage to other vehicles, and soldiers can only cause damage to other soldiers. If a unit falls below a certain health level, it is considered damaged/wounded and will need to be repaired/healed by a mechanic/medic. If a unit loses all of its health, it will be eliminated.

The Template design pattern will be used to define an interface for the entities.

The Abstract Factory design pattern will be used to produce the prototypes that are used in the creation of units.

The Prototype design pattern will be used by the AbstractFactory to create multiple instances of concrete entities.

The Memento design pattern will be used restore the original state of damaged entities once they have been attended to by a medic or mechanic.

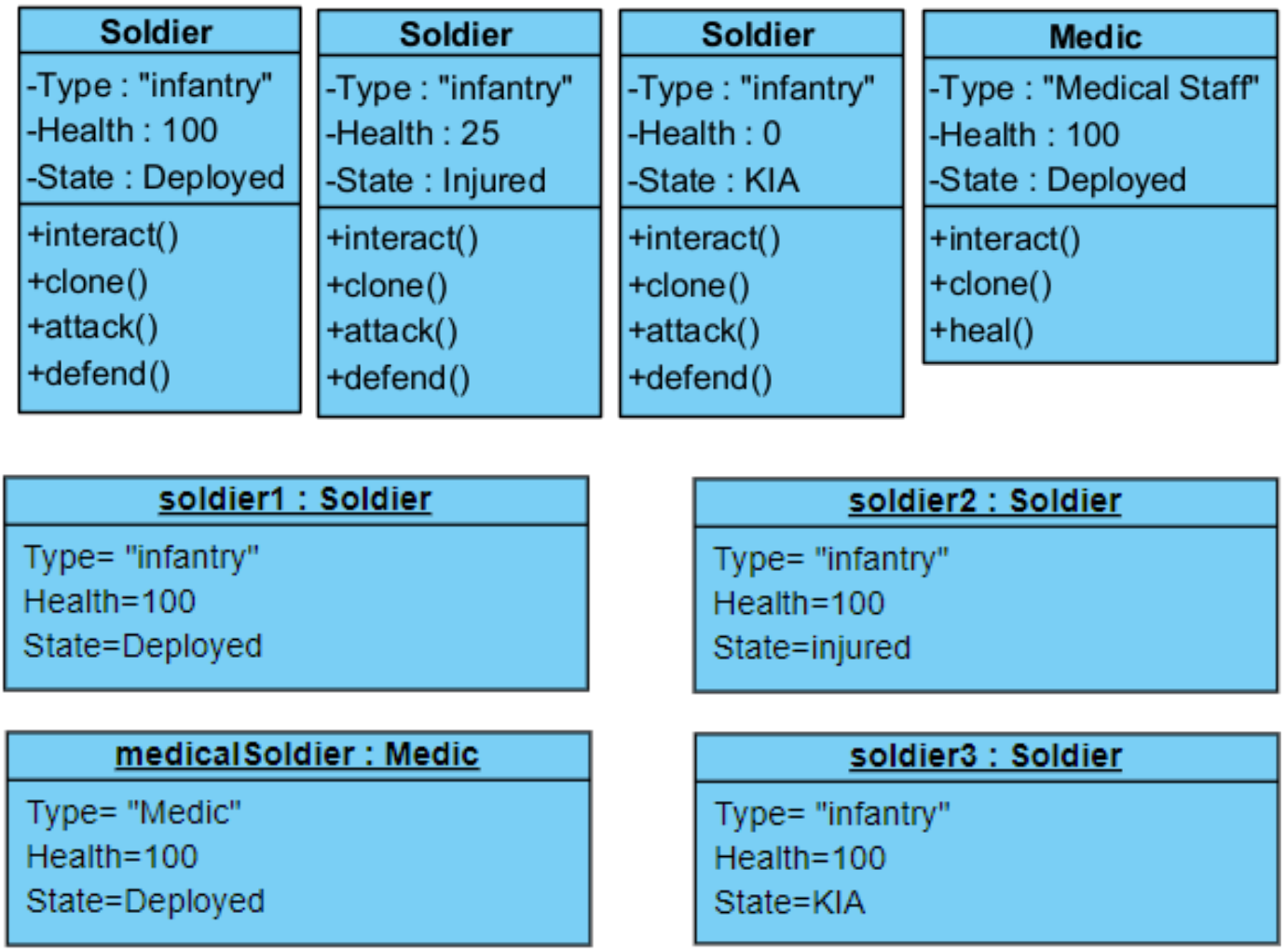


Figure 5: Entity Object Diagram of a Soldier and a Medic

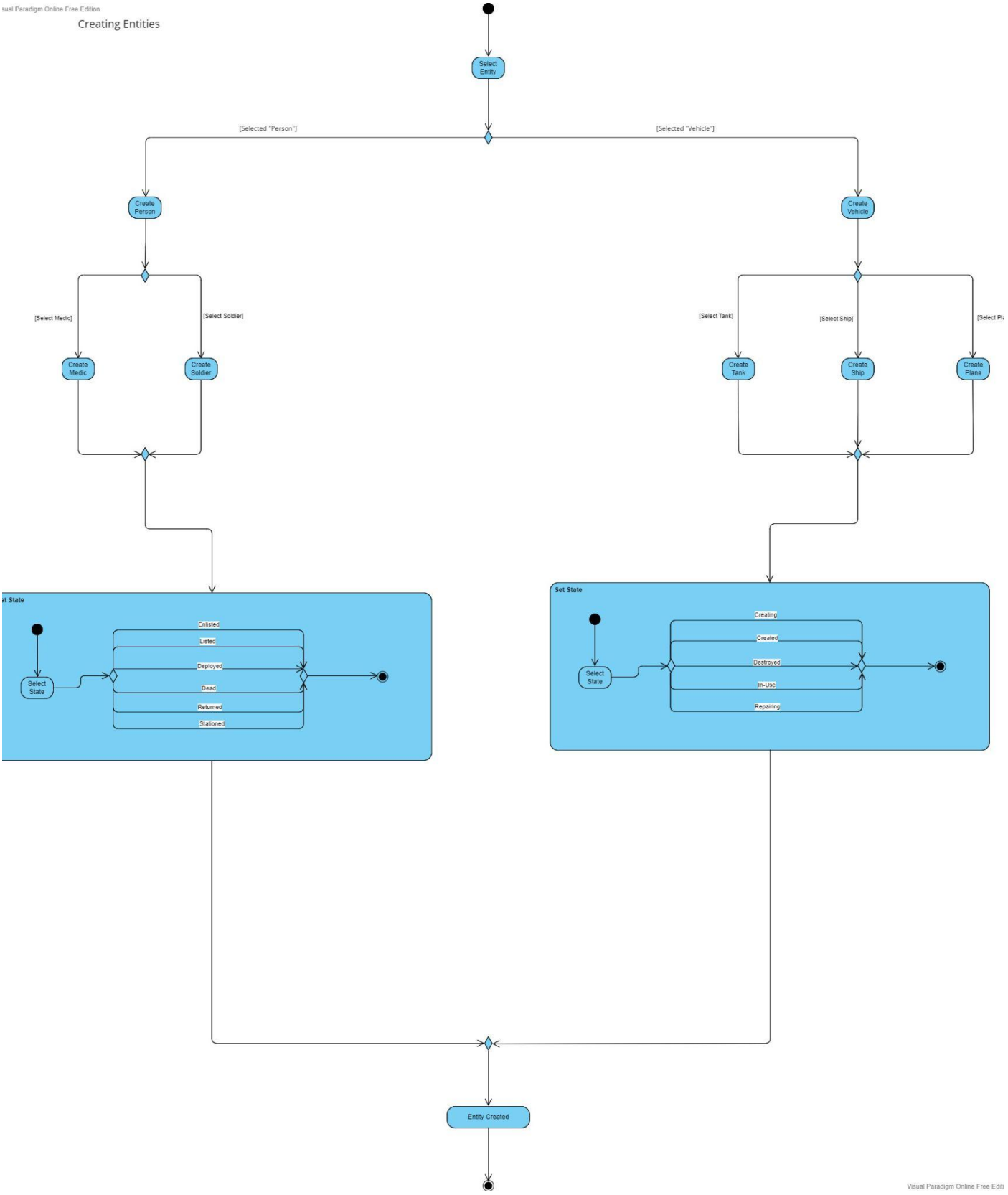


Figure 6: Entity Creation Activity Diagram

Phases of War

Phase 1: Dispute
The user specifies how many countries are involved in the conflict and how many alliances can be formed. The starting state of each country is also decided here, for example, military size and available resources.

Phase 2: Conflict
Alliances can be manually created by the user, or they can be randomly created by the program.

Phase 3: Fighting
In turn, alliances will attack and deal damage to one another. Resources can be managed before a turn ends. This entails maintenance of troops and vehicles, delivery of additional units and ammunition, and evacuation of refugees and prisoners of war.

Phase 4: Post-Conflict
Direct fighting ceases, but resource management still takes place and negotiations begin. This stage is entered if an alliance's units and/or resources are depleted or if they surrender. If negotiations are successful, the war progresses to Phase 5. If negotiations are unsuccessful, the war goes back to stage 3.

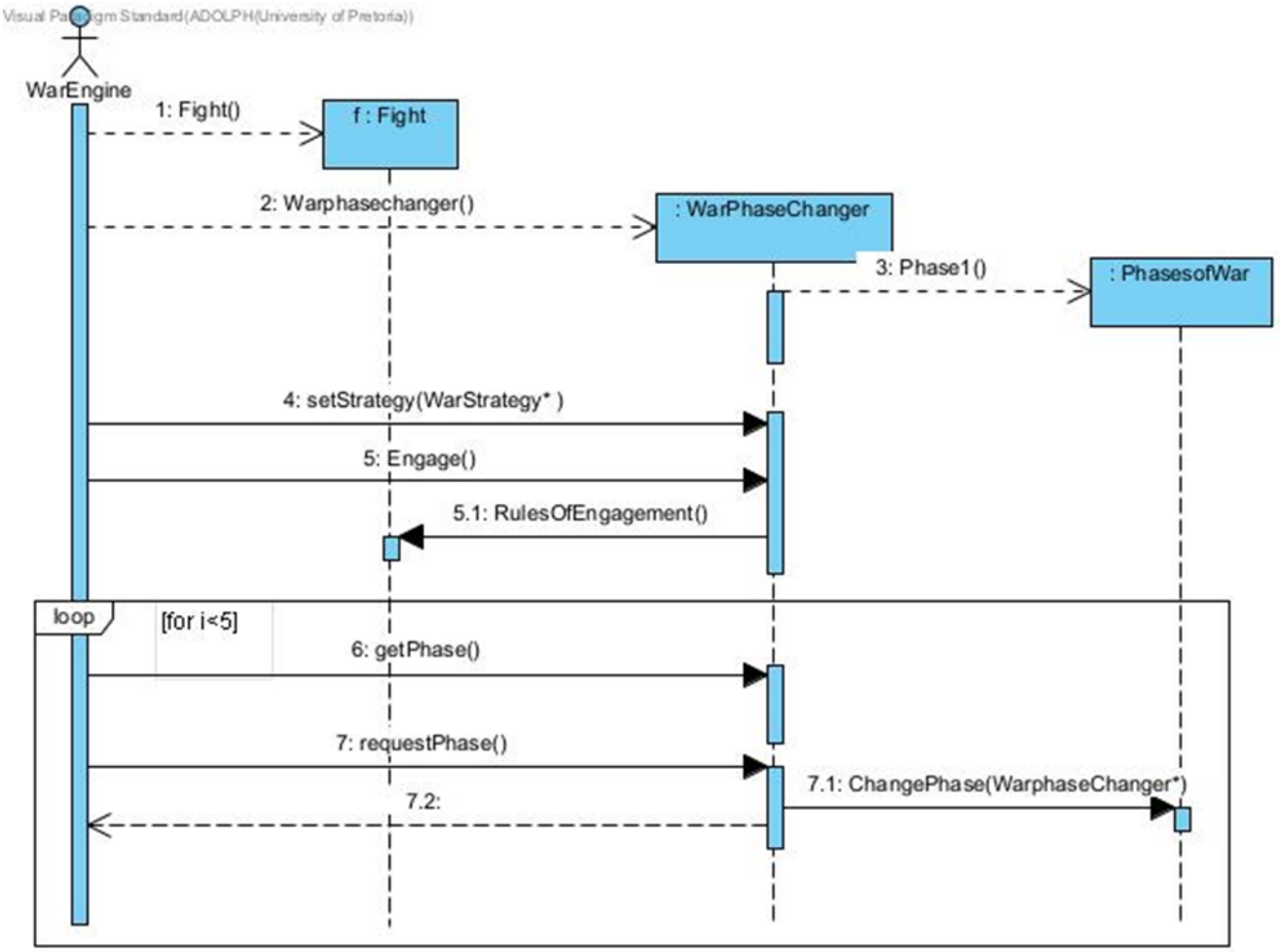
Phase 5: Diplomacy
If diplomacy is successful, the war ends. If it is unsuccessful and a new dispute arises, the war goes back to Phase 2 where alliances will be reformed, and the war will continue.

The State design pattern will be used to control the phases of war that the simulator is currently in. The states will change how the program runs and how all the entities behave depending on the state currently being used.

The Strategy design pattern will be used to define the algorithms that units use depending on the overall state of the program.

The Iterator design pattern will be used to progress through a turn in Phase 3. The iterator will step through the methods in the war theatre to produce an outcome.

The Command design pattern will be used in negotiations and diplomacy.



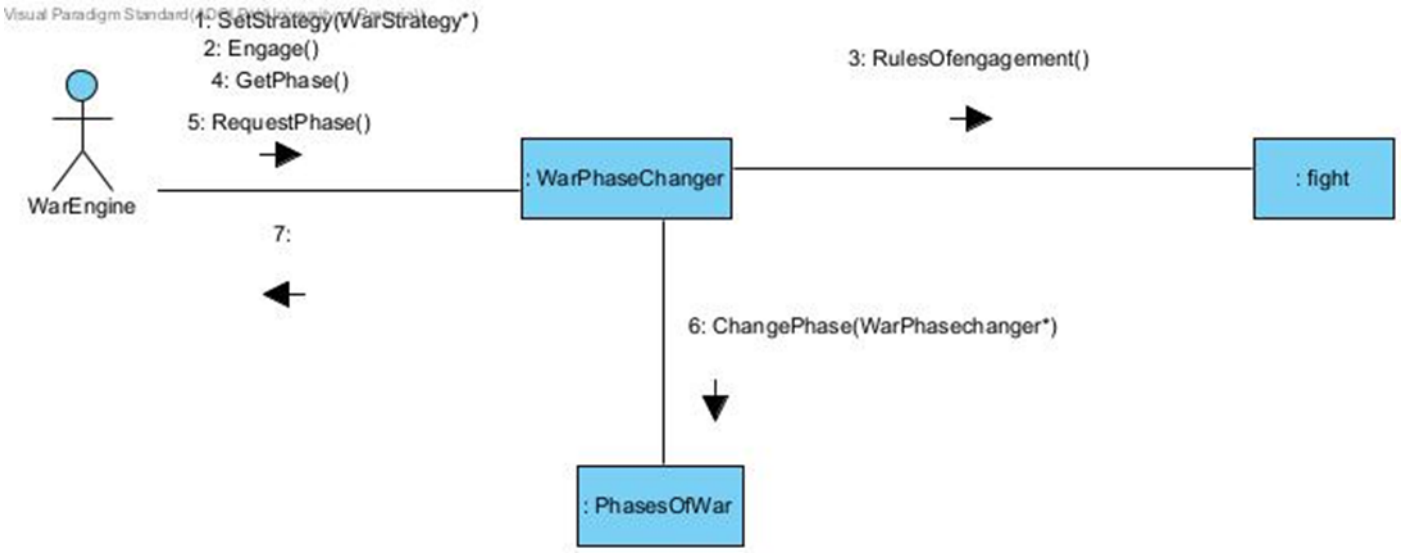


Figure 7: Phase and Strategy Sequence and Communication Diagrams

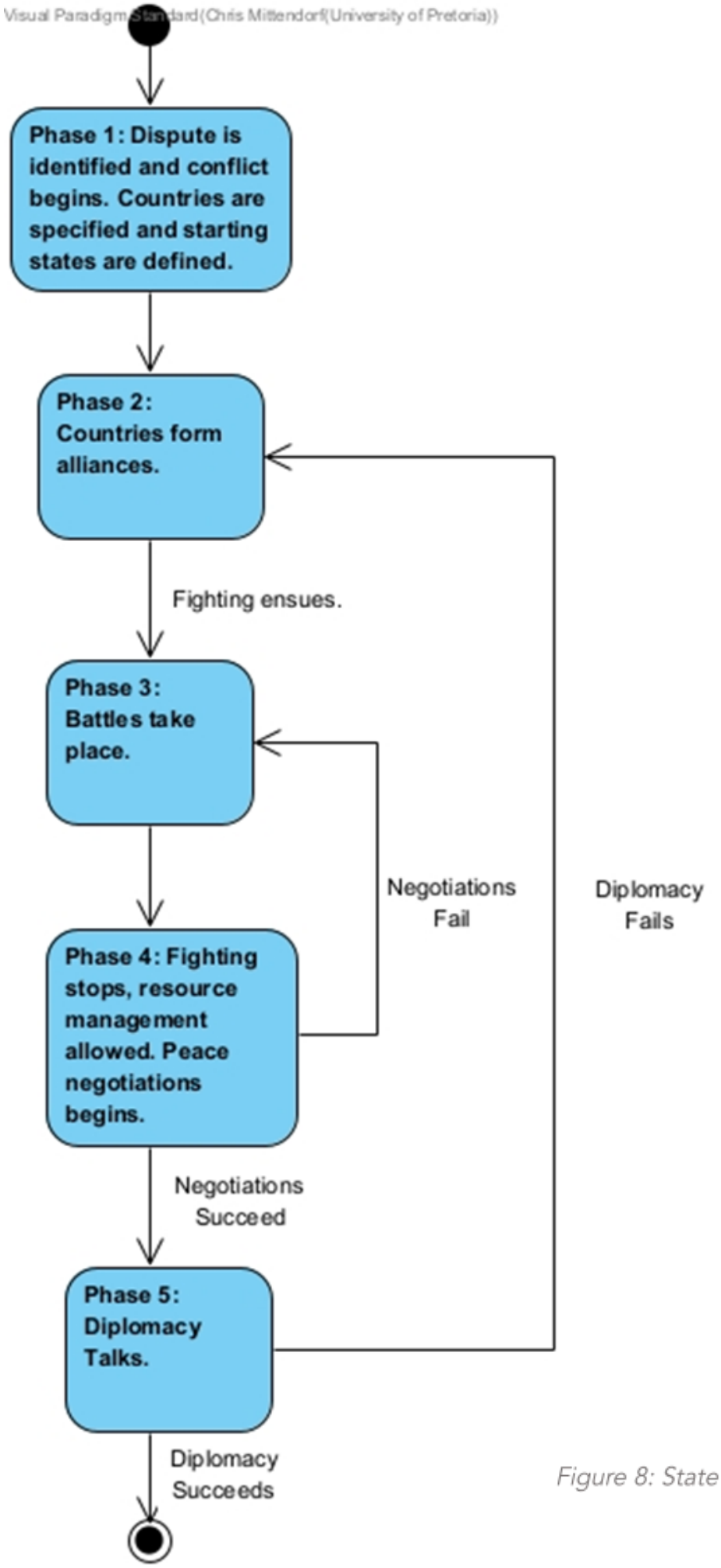


Figure 8: State of War State Diagram

The United Nations

In our simulation, the United Nations will be notified when war is waged and of the outcome of negotiations.

> The Observer design pattern will be used to notify the UN of wars and negotiation outcomes as well as to notify and update countries when the state of their alliance changes.

Changes in War

Countries will be able to join an alliance, leave an alliance, or change alliances while the simulation is running, as well as be able to upgrade the abilities of their entities.

> The Decorator design pattern will be used to add new methods and attributes to entities in order to upgrade them.

Class Diagram

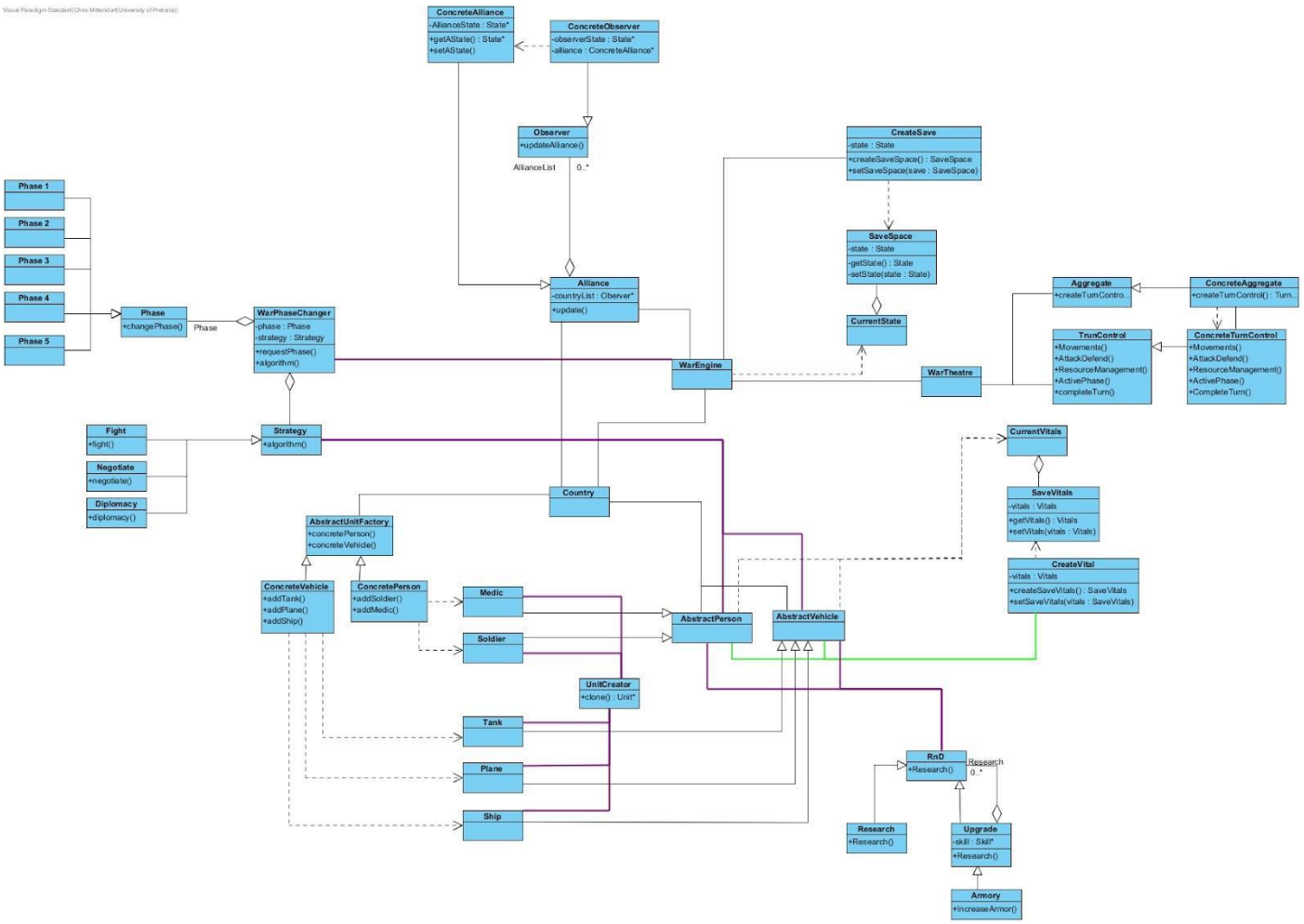


Figure 9: Simulation Class Diagram

Classes and their Relationships

Class1	Class2	Relationship Type
WarEngine	Country	association
WarEngine	Snapshot	association
WarEngine	Alliance	association
Country	UnitFactory	association
Country	Person	association
Country	Vehicle	association
WarEngine	CurrentState	dependency
WarEngine	WarTheatre	association
WarEngine	CreateSave	association
WarEngine	WarPhaseChanger	association
Alliance	Observer	aggregation
Alliance	ConcreteAlliance	generalization
Alliance	Country	association
WarPhaseChanger	Strategy	aggregation
WarPhaseChanger	Phase	aggregation
Phase	Phase1	generalization
Phase	Phase2	generalization
Phase	Phase3	generalization
Phase	Phase4	generalization
Phase	Phase5	generalization
Strategy	Fight	generalization
Strategy	Negotiate	generalization
Strategy	Diplomacy	generalization
Strategy	Person	association
Strategy	Vehicle	association

UnitFactory	VehicleFactory	generalization
UnitFactory	PersonFactory	generalization
PersonFactory	Mechanic	dependency
PersonFactory	Medic	dependency
PersonFactory	Soldier	dependency
VehicleFactory	Tank	dependency
VehicleFactory	Ship	dependency
VehicleFactory	Plane	dependency
UnitCreator	Mechanic	association
UnitCreator	Medic	association
UnitCreator	Soldier	association
UnitCreator	Tank	association
UnitCreator	Ship	association
UnitCreator	Plane	association
Person	RnD	association
Vehicle	RnD	association
Person	Mechanic	generalization
Person	Medic	generalization
Person	Soldier	generalization
Vehicle	Tank	generalization
Vehicle	Ship	generalization
Vehicle	Plane	generalization
RnD	Upgrade	generalization
RnD	Research	aggregation, generalization
Armory	Upgrade	generalization
CurrentVitals	Person	dependency
CurrentVitals	Vehicle	dependency
CurrentVitals	SaveVitals	aggregation

CreateVital	SaveVitals	dependency
CreateVital	Person	association
CreateVital	Vehicle	association
WarTheatre	Aggregate	association
WarTheatre	TurnControl	association
Aggregate	ConcreteAggregate	generalization
TurnControl	ConcreteTurnControl	generalization
ConcreteAggregate	ConcreteTurnControl	dependency, association
CreateSave	SaveSpace	dependency
CurrentState	SaveSpace	aggregation

References

[1] Gamma, E. *et al.* (1998) *Design patterns elements of Reusable Object Oriented Software*.

[2] Bloomfield, L.P. (1997) *Why Wars End: CASCON's Answers from History*. Available at: https://web.mit.edu/cascon/warend.html

[3] von Clausewitz, C. (1918) *On War*

[4] MasterClass (2022) *United Nations Explained: What Does the United Nations Do?* Available at: https://www.masterclass.com/articles/what-is-the-united-nations

Link to DOCX

REPORT.docx