

COS 214 PROJECT

REPORT

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

2.1 Identify the functional requirements.

Main points

- Allow the user to decide how the program runs (Either as Design Mode or Real Mode)
- Allow the user to specify how many countries and alliances will be formed and the initial state of the countries.
- The program will need to run through the phases of the program by either accepting inputs from the user(design mode) or producing its own inputs to continue the program(real mode)
- Each country should be able to build their own army that will be used in a war. This includes flags, weapons, troops, soldiers and infantry.
- Countries should only be allowed to attack countries that are not in their group.
- Countries should have the choice to be able to form an alliance with other countries.
- For any party to engage in warfare, their entities need to be entered into the war theatre.

Introduction

We were tasked to implement a war simulator that will model the various elements of war. This program should be able to simulate a turn based style war that can take place between two countries. The war must take place within a specified location that can be defined as the war theatre. Each country will have its own assets, like the amount of soldiers, weapons, medics, infantry etc. Below is a detailed description of the requirements for the simulator:

In Depth Description

Countries can be alone or part of an alliance(data structure)

Countries have:

- Soldier/armies/troops
- Weapons
- Flags

There are multiple alliances

5.1 War engine

Countries take turns **ordering attacks** on countries from other alliances, until **one alliance**(super power country group) **remains**

5.2 Components (can add RnD to “upgrade” units)(Construction of new units)

5.2.1 War theatre

- List of defending countries(in same alliance)
- List of attacking countries(in same alliance)
- Location
- Area: Air space / land / sea

5.2.2 Transportation

- Transport **goods**(weapons), **people**(troops) and **services**(medics, engineers)
- Should be able to destroy transport

5.2.3 Entities

- Keep track of time elapsed
- Entities will all go through a timed process(eg: construction of a vehicle), after which the entity will be added to the corresponding country
- How aggressively countries respond to provocation and attack

5.2.4 Phases of war

- Phase 1 (Dispute) : A dispute takes place between countries. The user specifies how many countries are involved in the conflict and how many alliances can be formed. The starting states of each country is also decided here, for example the military size and resources available.
- Phase 2(Conflict) : The countries form alliances against one another. This can be done by the User or done randomly in the program.
- Phase 3(Fighting) : In this phase there is direct conflict between the alliances. In turns the alliances will attack one another and damage will be caused to the defender. Resources will also be able to be managed before the end of a turn, this entails the maintenance of troops or vehicles, delivery for more units and ammunition and then the evacuation of refugees/POWs.
- Phase 4(Post-Conflict) : In this phase direct fighting stops but resource management still takes place. This stage is entered if an alliance's units and/or resources are depleted or if they surrender. Negotiations between alliances take place at this stage in order to end the conflict. If the negotiations are successful the war progresses to Phase 5. If negotiations are unsuccessful then the war goes back to stage 3.
- Phase 5(Diplomacy) : This is the final phase of the war where the dispute is settled. If diplomacy is successful then the war ends. If diplomacy is unsuccessful and a new dispute arises then the war goes back to Phase 2 where alliances are reformed and the war continues.

Influences on the above phases were drawn from the following article:

<https://web.mit.edu/cascon/warend.html>

5.2.5 Changes to war engine

As **time progresses** and according to **other factors** countries should **behave differently**:

- On how they **spend money**(eg: they spend a lot of money in the middle of the war)
- Adopting **war treaties**(eg: they decide to adopt peace treaties towards the end of the war)
- On how aggressively they **respond to provocation or attack**(eg: if they know another country is weak, they might not respond on provocation)
- Countries can also **change or leave alliance**(eg: if they are losing, or if an has performed too many invasions)

5.3 War reenactment

Interface where a user can run the war simulation

Should have 2 modes:

- **Design mode**: changes can be made on the fly
- **Real mode**: simulations are set up and run

2.2 Design the processes using Activity diagrams.

2.3 Decide on the patterns to address the functionality defined by the functional requirements and processes.

- Memento - used to restore the state of entities(units/vehicles) after they have been helped by a medic/mechanic. Also used to save the state of the entire system on the interrupts in the design mode in case the user wishes to pause the game or there is a system crash.
- Template - used to define the rough layout of an entity (units/vehicles).
- AbstractFactory - used to produce the prototypes that are used in the creation of units.
- Prototype - used by the AbstractFactory to create multiple instances of concrete products.
- Strategy - used to define the algorithms that units use depending on the overall state of the war engine program.
- State - used to control the phases of war that the war engine program is currently in.
- Decorator - used to add new methods/attributes to a unit as the war progresses.
- Iterator - can be used to progress through a turn in Phase 3 (eg: attack->manage resources->evacuate personnel, etc.)
- Singleton - used to keep track of the the time of the battles (turns taken)
- Command - might be able to use this in diplomacy/negotiations?
- Observer - used to notify the alliances of the change in a country once a battle has completed
- Composite - The lists of attacking/defending countries in a war theatre will be a composite, so that we can call each of their functions at the same time.

2.4 Design the classes for each of the identified patterns taking their interrelationships into account.

V1.0

1. WarEngine - Country(association), Snapshot(association), Alliance()
2. Alliance - Country(Template/Iterator-Generalization)
3. Country - EntityFactory(association), ProductEntity(association)
4. WarTheatre - Country(Not sure about the relationship?), Alliance(Not sure about the relationship?)?
5. EntityFactory - Country(association), PersonFactory(generalisation), VehicleFactory(generalisation), WeaponsFactory(generalisation)
6. PersonFactory - ProductPerson(dependency)
7. VehicleFactory - ProductVehicle(dependency)
8. WeaponsFactory - ProductWeapons(dependency)
9. ProductEntity - Country(association), ProductPerson(generalisation), ProductVehicle(generalisation), ProductWeapons(generalisation)
10. ProductPerson
11. ProductVehicle
12. ProductWeapons
13. State - ProductPerson(aggregation), ProductVehicles(aggregation), ProductWeapons(Aggregation)
14. Concrete States for ProductPerson - Listed, Enlisted, Deployed, Dead, Returned, Stationed, etc.
15. Concrete States for ProductVehicles - Creating, Created, Destroyed, In-Use, Repairing, etc.
16. Concrete States for ProductWeapons - Same as Vehicles?
17. Snapshot(Memento) <- SaveState(Association) - WarEngine

V2.0

Classes:

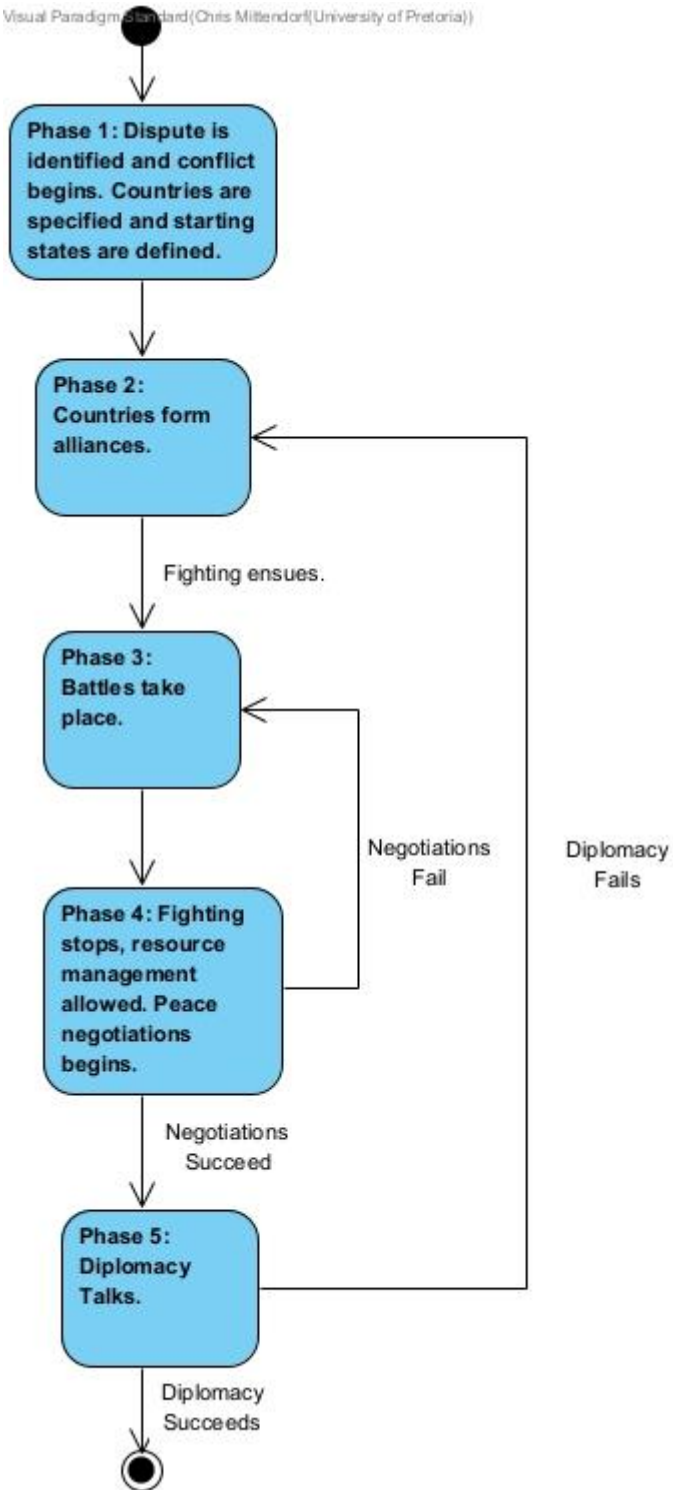
1. WarEngine
2. Alliance
3. Country
4. WarTheatre
5. EntityFactory
- 6.

2.5 Draw a class diagram of your system.

2.6 Draw Sequence and communication diagrams showing the message passing between objects.

2.7 Design state diagrams showing how an object (which could also be a composite) changes state.

Visual Paradigm Standard (Chris Milledore (University of Pretoria))



2.8 Provide at least two object diagrams showing the state of the objects active in the war simulation at a specific point in time.