Distributed System "Drone Explorer" Process Report

Federico Foschini
Stefano Braggion
Luca Maestri
Riccardo Delvecchio

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy
federico.foschini4@studio.unibo.it
luca.maestri@studio.unibo.it
riccardo.delvecchio@studio.unibo.it
stefano.braggion2@studio.unibo.it

1 Introduction

This document represents the complete production process of the Drone Explorer, final task of the course of Distributed Systems.
In this work we want to show the acknowledgements and concepts acquired during the course of Distributed Systems.

2 Vision

Almost any computational system of today comes equipped with ICT technologies for interacting with other computational systems.
In this scenario Distributed Systems get a particular important role.

3 Goals

The main goal of this project is to create a complete and working distribuited system. The project must be also well-designed in order to accomplish all the concepts covered during the course.
The target of this project is also to show the procedure by which we arrive at this result, with a particular emphasis at the steps to get to the final product, showing the knowledge acquired during the course.

4 Requirements

The project consists of a distributed system composed of a master entity that takes care of driving drone entity whose navigation data are viewed by other passive entities. More precisely, the tasks of the master are:

- set the direction of the drone;
- receive the navigation data of the drone;
- create a map of the grid in which the drone travels through the data received.

The tasks of the drone are:

- receive master's commands;
- receive this commands about navigation;
- send data that contain navigation data of the grid explored (by receiving data from the sensors).

Finally, the role of the passive visualization entity are limited only to receive the navigation data of the drone and display it. Should be noted how is simulated the reception of data from drone's sensor, through a preset grid that the drone will check, to verify the presence of obstacles on the path (or rather a number of query on a collection of data present in the drone, in order to have a simulator). The communication is performed over the network using json for data formatting. All is communicated between the various entities through a web service on-line, that consist of some php pages and a dedicated database.

5 Requirement analysis

From what emerges from the requirements it can be seen that the system that is required to implement is composed of four macro parts, each of these provides additional functionality to the system, also it can be assumed that each of these corresponds to a stand-alone software system to implement in order to meet the requirements. The four main component parts of the system are:

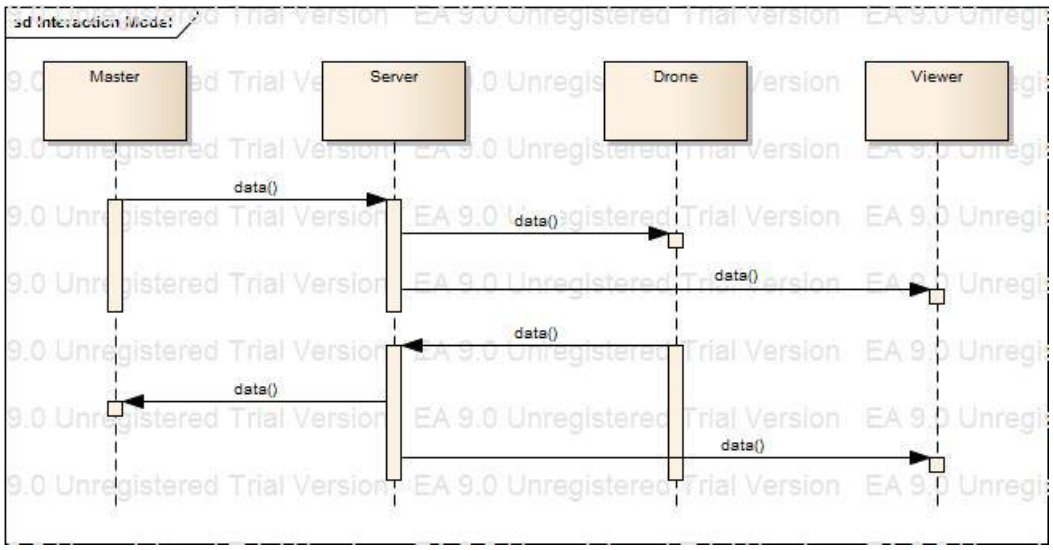
- *The Drone* is a software system that must simulate the behavior of the activity of a drone on a exploring mission;

– *The Master entity* this must first of all provide two basic functionality to the system, which are: to display the received data and messages from the drone and allow, through the command set, the possibility of controlling the drone;

– *The Viewer* have to display the data collected by the drone;

– *The Database* must collect the data received from drone and the command from master, and make this avaible to all entity of the distribuited system.

Also another important common entity is the *middleware* that allow communication between the variuos macro entity of the distribuited system. A first behavior of the required system is shown in the following picture:



5.1 Glossary

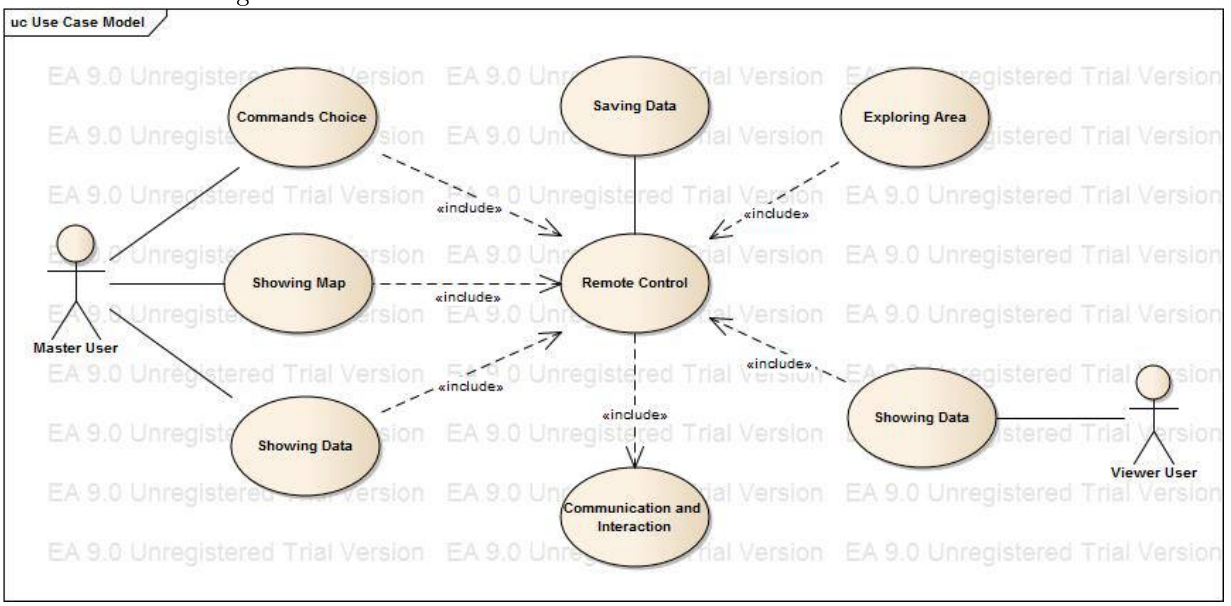
The table below shows the definition of terms that will be used several times, during the development process, to provide a elementary and understandable definition as possible in natural language.

Glossario

Name	Definition
Drone	It's an vehicle characterized by the absence of the human pilot on board. Its behavior is controlled by the computer on board, it is under the remote control of a navigator or pilot on the ground or in another vehicle.
Sensor	Device that measures a physical quantity in input and provides an output signal for the purpose of measurement or control of the system in which it is used.
Viewer	Tipically a remote device that is expected to show information in a Display. These informations are about an object, event, person or location that is hidden from physical view and separeted at some distance.
Display	Entitiy that shows the data given in input.

5.2 Use cases

After a first reading of the requirements, is clear the following Use Case represented from the image below:



5.3 Scenarios

The following scenarios show the behavior of the system.

Scenario: Showing Data

Field	Content
ID (Nome)	UC1 - Showing Data
Description	Shows the message received by the drone.
Primary Actors	Master User and Viewer User.
Main course	Each time that a message is received it must be displayed.
Success Condition	The message is shown.
Failure Condition	Invariants are not respected.

Scenario: Showing Map

Field	Content
ID (Nome)	UC2 - Showing Map
Description	Shows the map explored by the drone.
Primary Actors	Master User.
Preconditions	There are no malfunctions on the drone or in the means of communication between drone and Master.
Main course	Each time the system performs the updating of its data in memory these must be shown.
Success Condition	The map is displayed correctly.
Failure Condition	The map is not displayed.

Scenario: Saving Data

Field	Content
ID (Nome)	UC3 - Saving Data
Description	Collect the data sent.
Primary Actors	Master User, Viewer User.
Preconditions	There are no malfunctions.
Main course	Take a backup of the data received.
Success Condition	The information is collected successfully.
Failure Condition	The information is not collected.

Scenario: Exploring Area

Field	Content
ID (Nome)	UC4 - Exploring Area
Description	Explore the required area.
Preconditions	There are no malfunctions.
Main course	There aren't malfunctions on the drone.
Success Condition	The area is correctly discovered.
Failure Condition	Invariants are not respected.

Scenario: Remote Control

Field	Content
ID (Nome)	UC5 - Remote Control
Description	Allow the control between remote components of the system: drone, master, viewer.
Primary Actors	Master User, Viewer User.
Preconditions	There are no malfunctions.
Main course	Allow remote control of the components of the system.
Success Condition	The components are able to communicate.
Failure Condition	The exchange of messages does not happen or reach corrupted information.

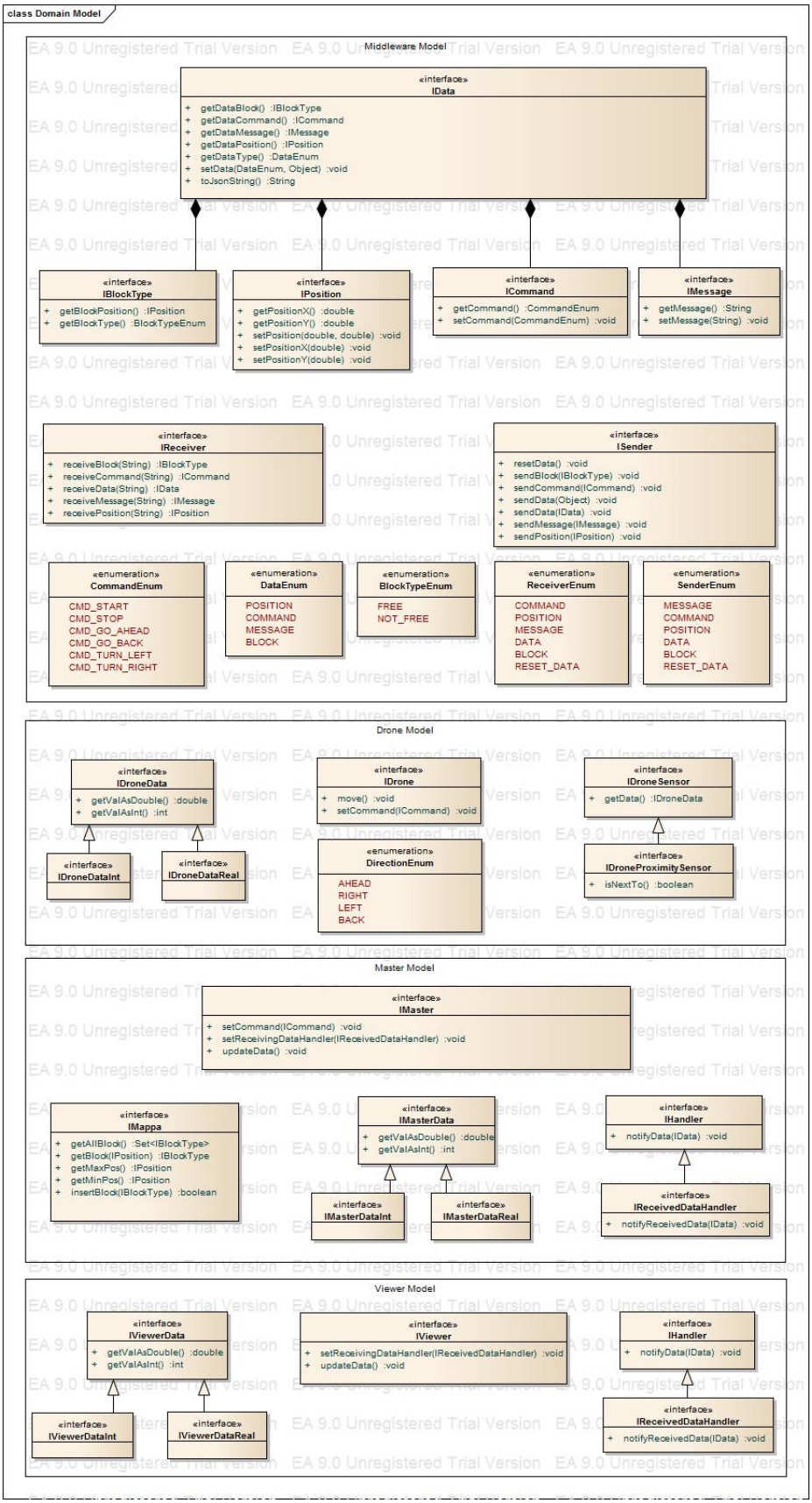
Scenario: Command Choise

Field	Content
ID (Nome)	UC6 - Command Choise
Description	Collect the commands to be sent to the drone.
Primary Actors	Master User.
Preconditions	There are no malfunctions.
Main course	Gather input data.
Success Condition	The information is collected successfully.
Failure Condition	The information is not collected.

Scenario: Communication and Interaction

Field	Content
ID (Nome)	UC7 - Communication and Interaction
Description	Manage the communication between remote components of the system: drone, master, viewer.
Primary Actors	Master User, Viewer User.
Preconditions	There are no malfunctions.
Main course	Manage the exchange of information between the components of the system.
Success Condition	The components are able to communicate.
Failure Condition	The exchange of messages does not happen or reach corrupted information.

5.4 Domain Model



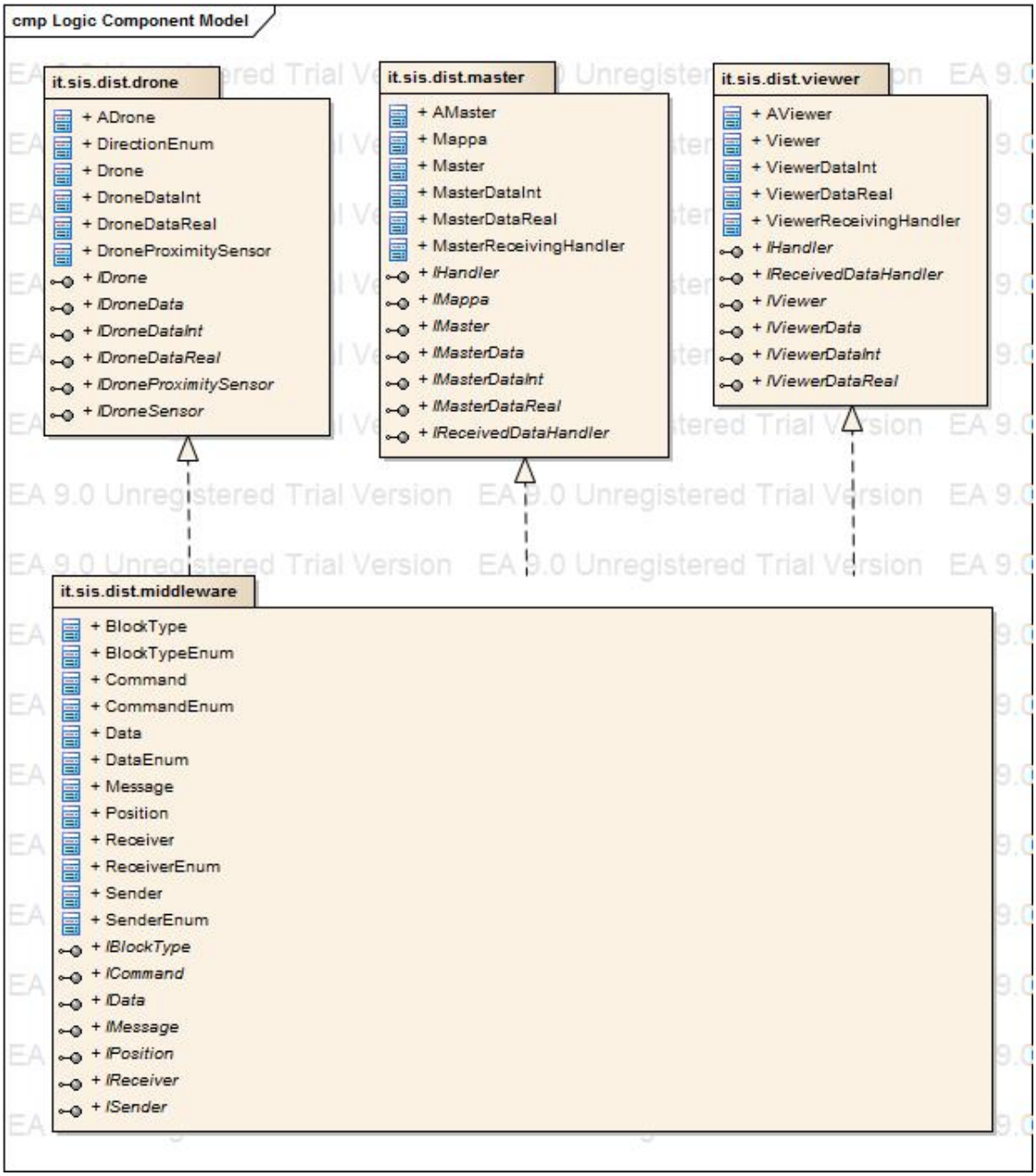
The image shows the fundamental entities that compose the system. By showing all every macro-entity that define the system.
The *Middleware Model* is composed by the entity that allow interaction between the macro-entities of the distribuited system, *IReceiver* and *ISender* have the function to receive and send *IData* through the network, *IData* is the base type of data that is specialized by *IMessage*, *ICommand*, *IPosition*, *IBlockType*, which have a different semantic meaning.
The *Drone Model* shows the entity that compose the drone simulator, in particular the entity *IDroneSensor* rappresent a generic sensor, *IDroneProximitySensor* is the component who provides the property to scout the area near the drone by notify the presens of blocks.
The *Master Model* and the *Viewer Model* shows the entity that compose the master and the viewer macro-entity of the distribuited system.

6 Problem analysis

According to what defined in the Requirements Analysis we proceed by defining the entities identified, by showing their structure, their behavior and their interactions, through the graphs in the following paragraphs. Subsequently are then analyzed the middleware, master, drone and viewer by providing a description of how they should perform their duties free from the issues of implementation, not required at this stage of analysis.

6.1 Logic architecture

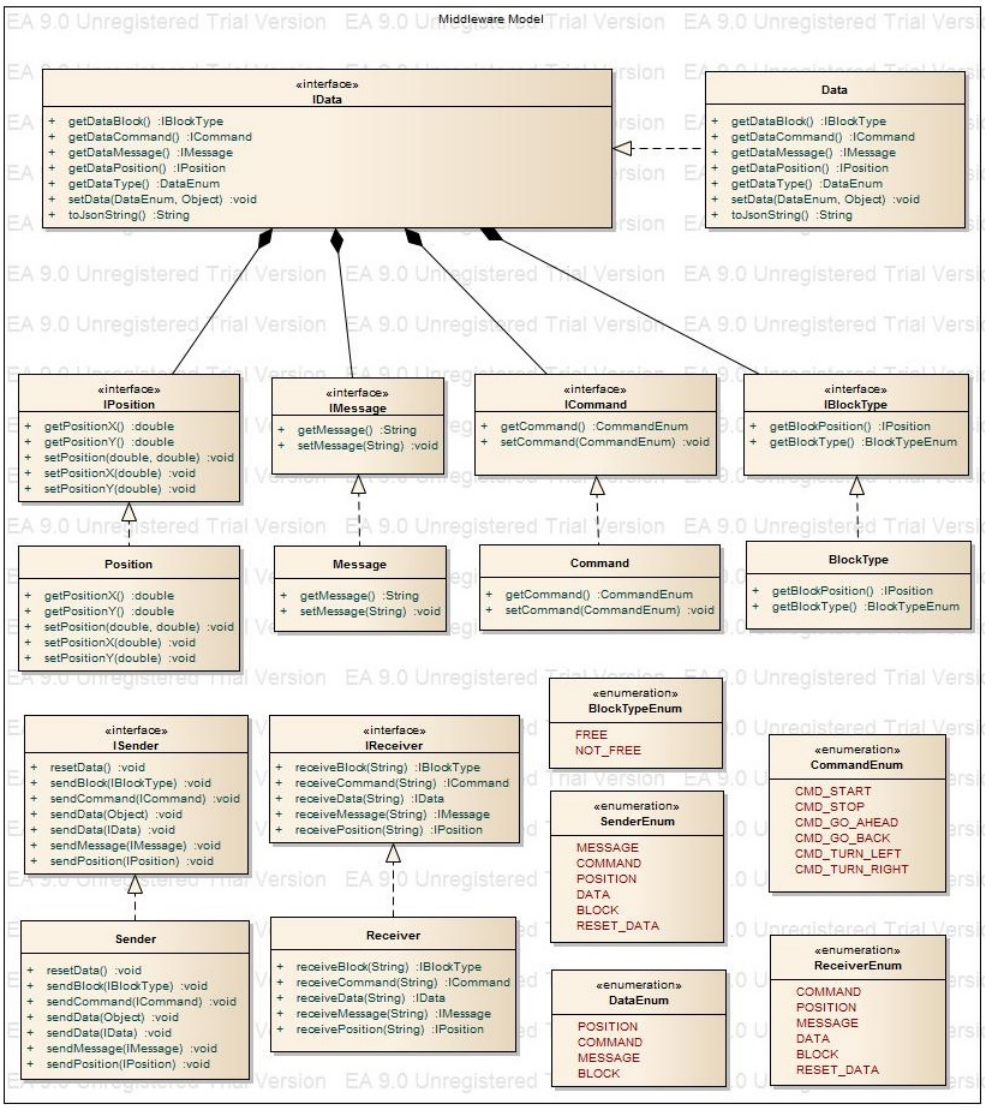
The following immage shows the logic architecture of the system. It describes the various entities, roles, and relationships that govern the system.



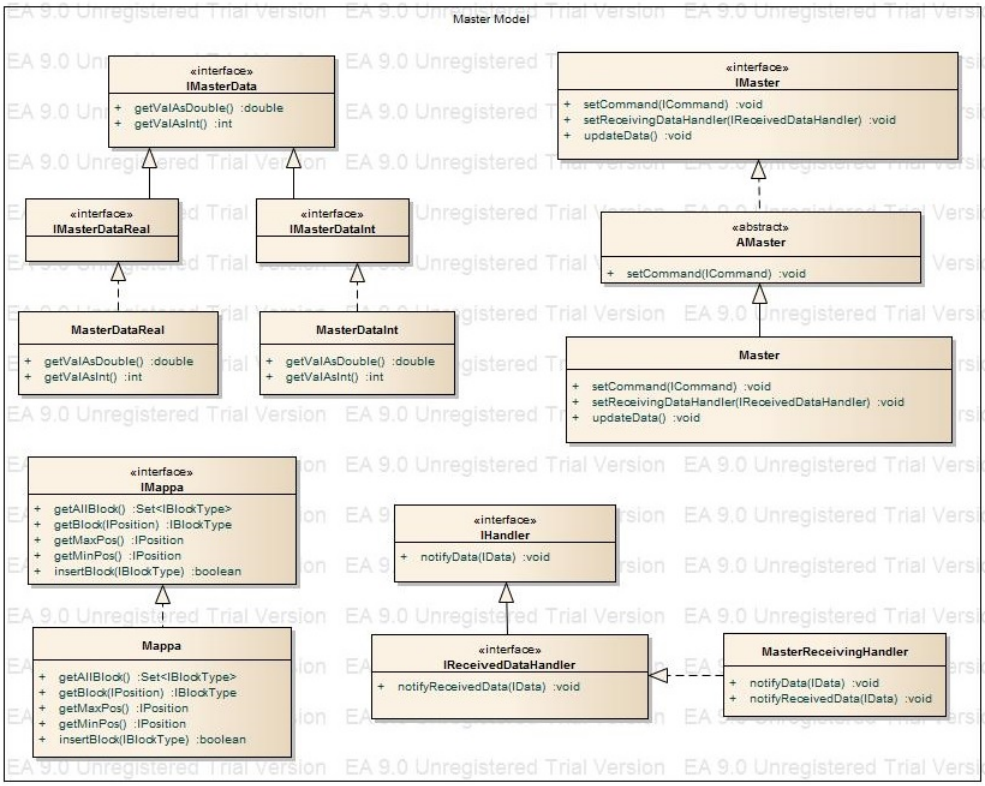
6.2 Structure

The considerations presented in this section show the structure of the entities defined in the requirement analisys in detail, to provide a better understanding of the internal organization of the primary components:

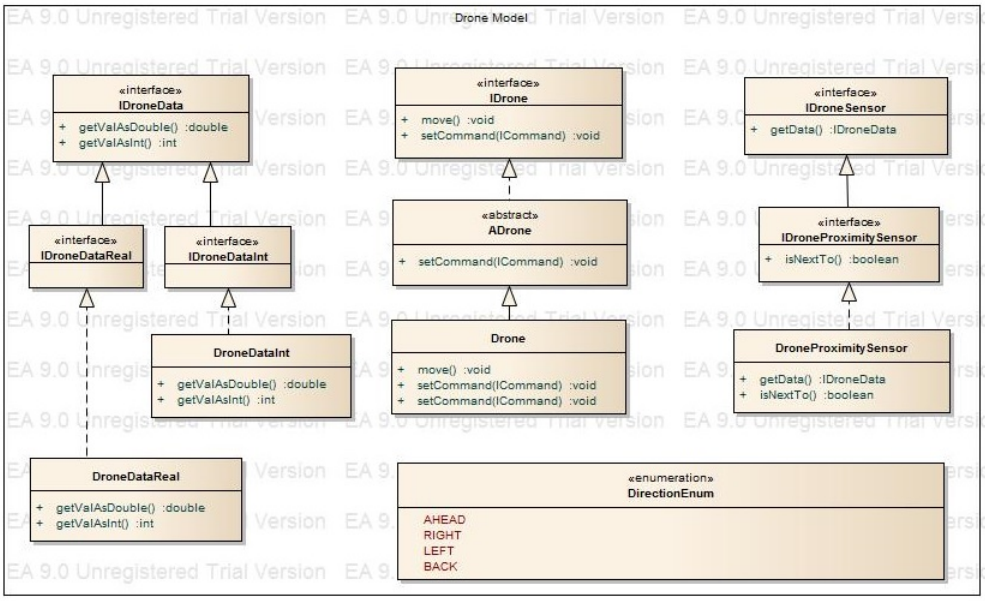
The following is the subsystem relating to the middleware.



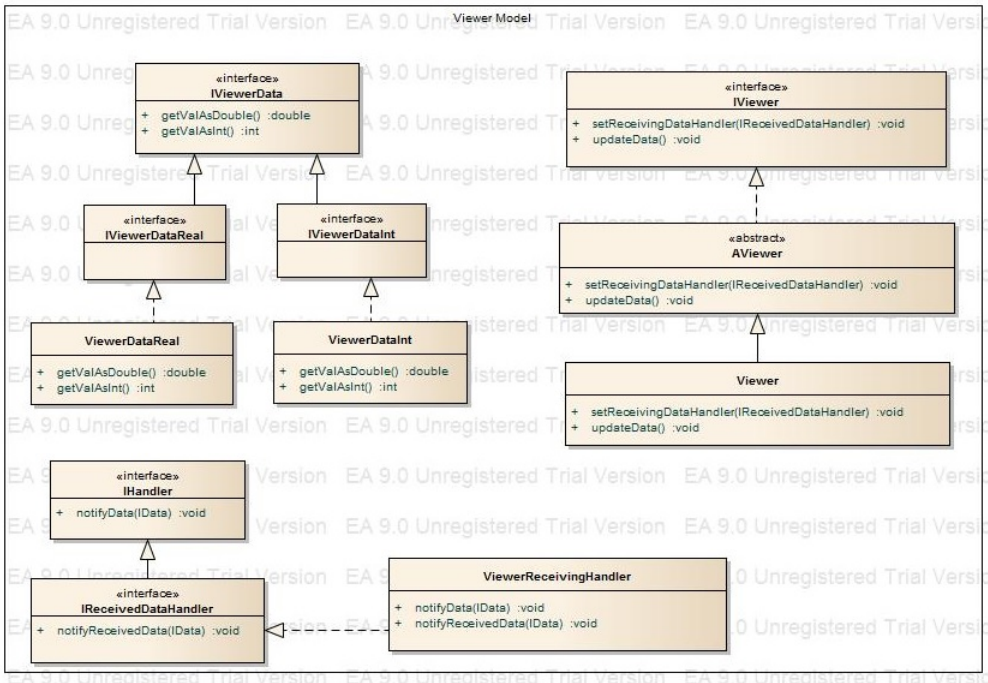
The following is the subsystem relating to the master entity.



The following is the subsystem relating to the drone entity.

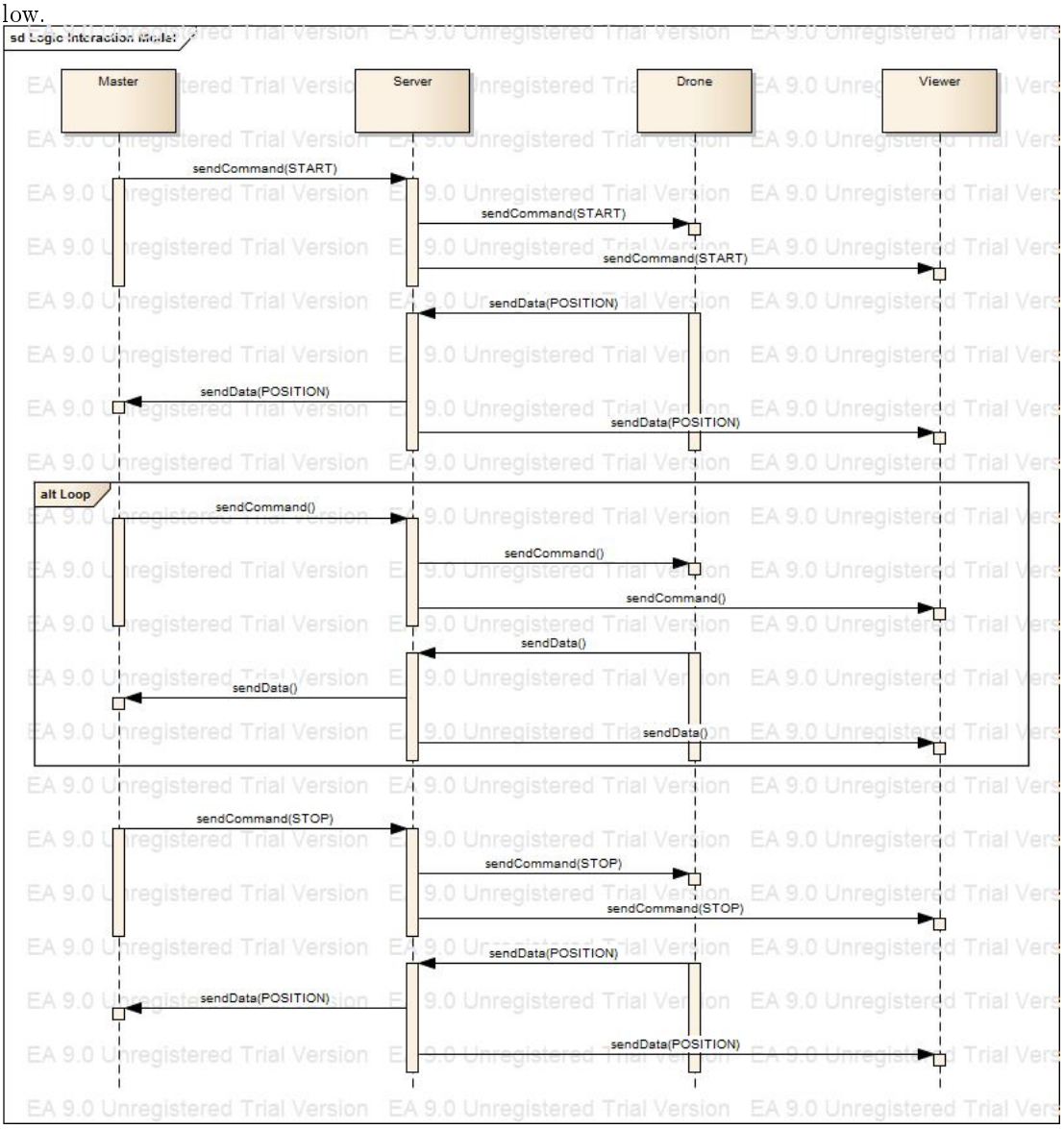


The following is the subsystem relating to the viewer entity.



6.3 Interaction

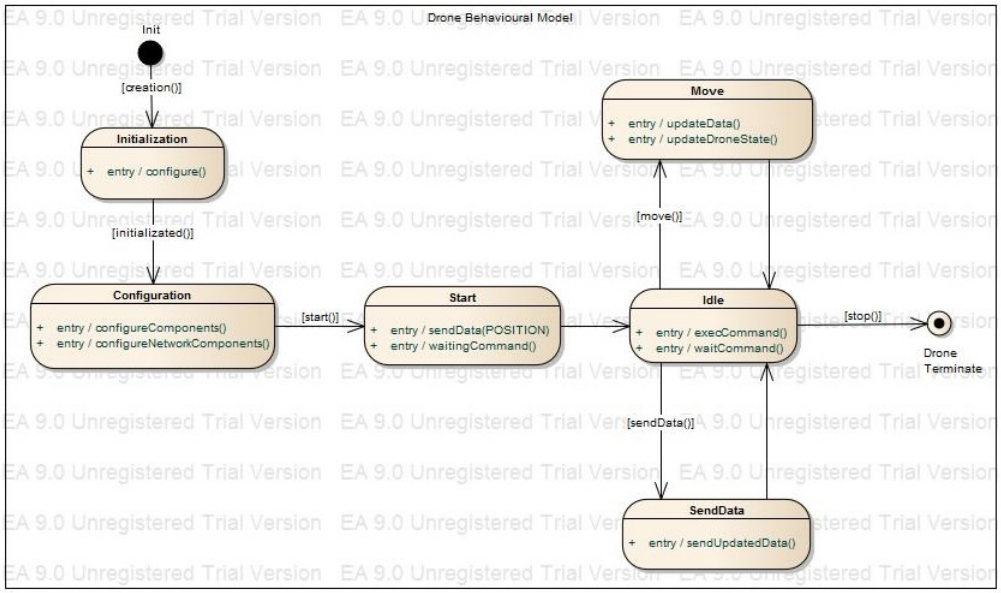
The interaction between the components of the system can be explained as follow.



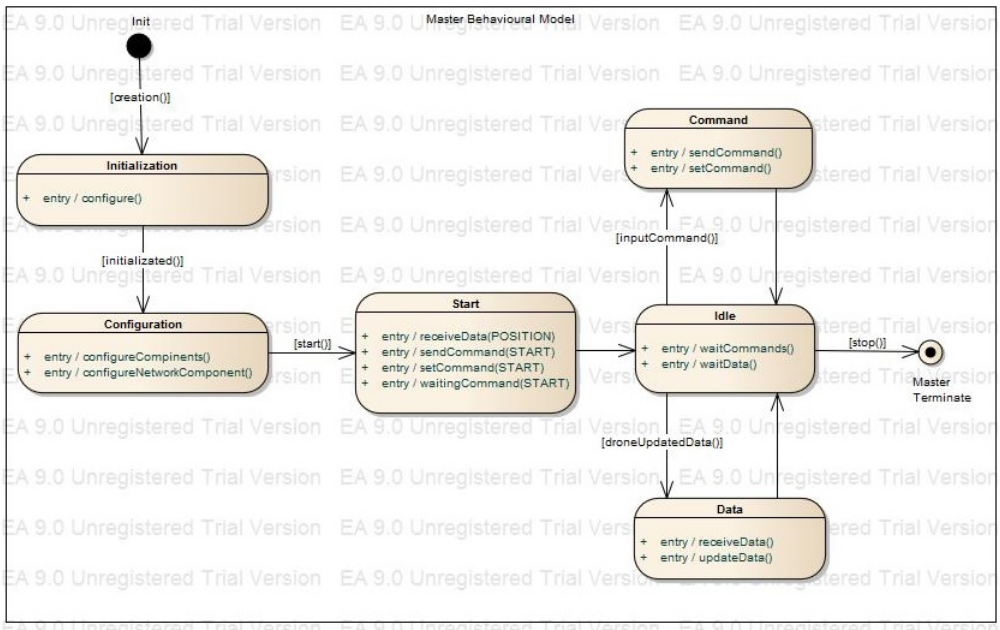
6.4 Behavior

Below is shown the behavior of the elements of the system, through the *UML State Machine Diagrams*, diagrams showing states and transitions that involve changes in them.

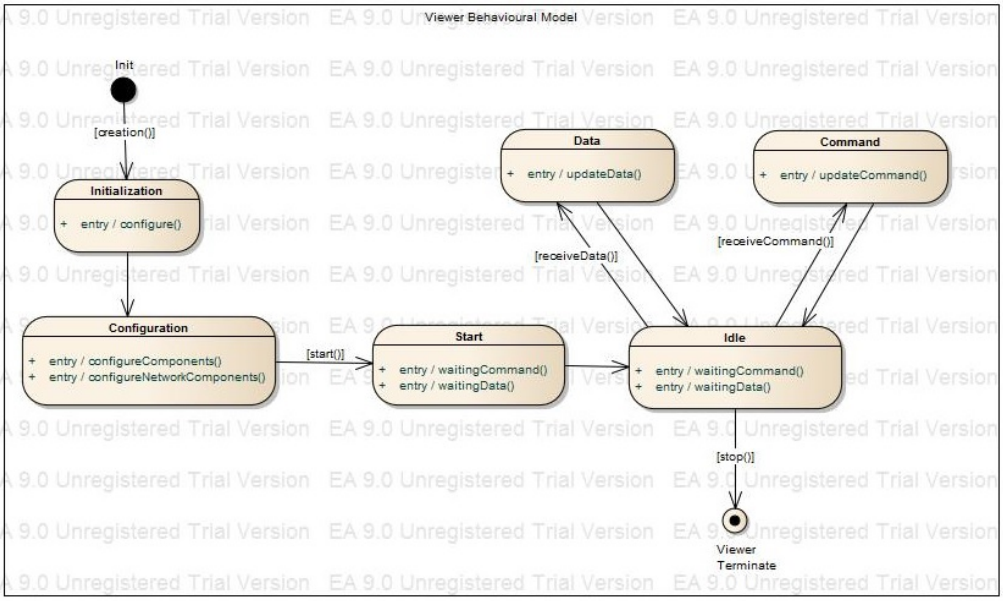
The following is the behaviour model relating to the Drone.



The following is the behaviour model relating to the Master.



The following is the behaviour model relating to the Viewer.



6.5 Abstraction gap

The abstraction gap is not very high because the system is completely achievable through the use of technologies that have been available.

6.6 Risk analysis

The development of the overall system can be viewed as a medium risk, since the basic parts of the system can be implemented in almost all the technologies because they are based on a model completely independent from the technology development. In addition there is an excellent structure for subsystems as they can be reused thus reducing the weight of any changes so that will not cause any radical change in the overall structure. Finally, you can see how the middleware has high openess since it is structured so as to support any additions of new entities in the future.

7 Work plan

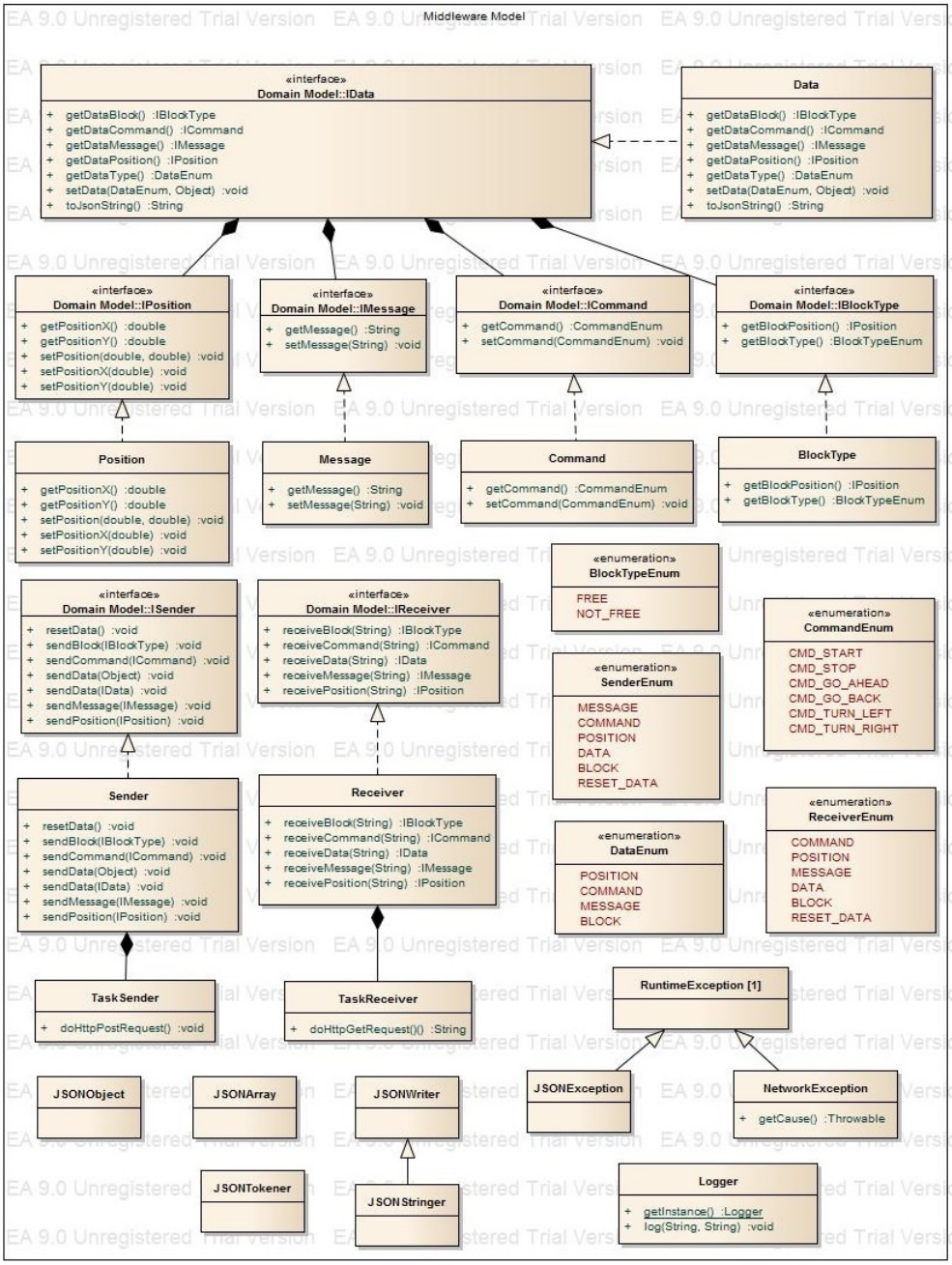
As the project related to the theme of the course all the team members became involved in all stages of the production process. The only real division of work was found in the implementation phase since it was decided to operate under the strategy of *divide et impera* in order to parallelize as much as possible.

8 Project

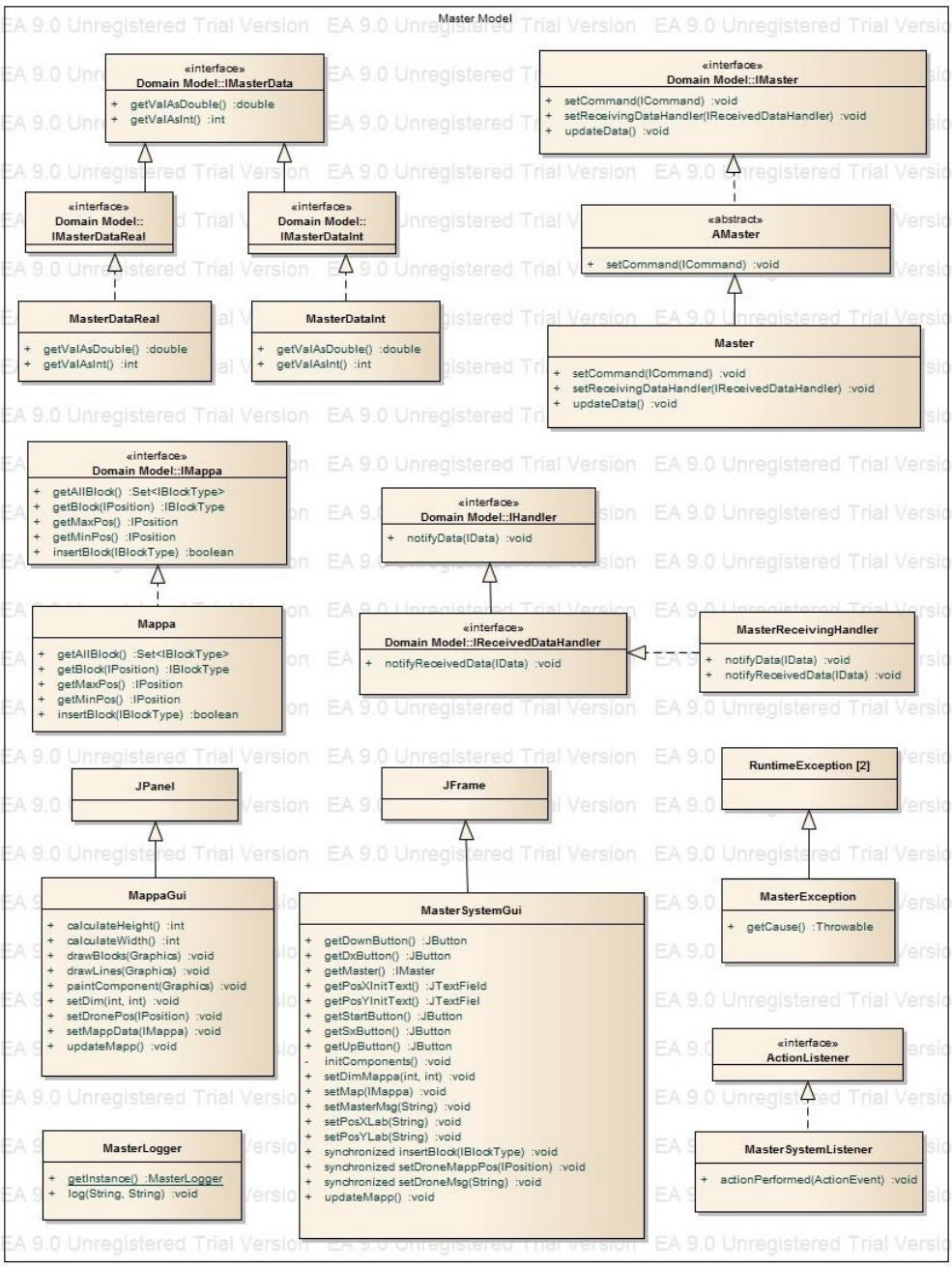
In this phase is shown the evolution of the final system as defined in the phase of Problem Analysis, by showing its structure, its interaction and its behavior.

8.1 Structure

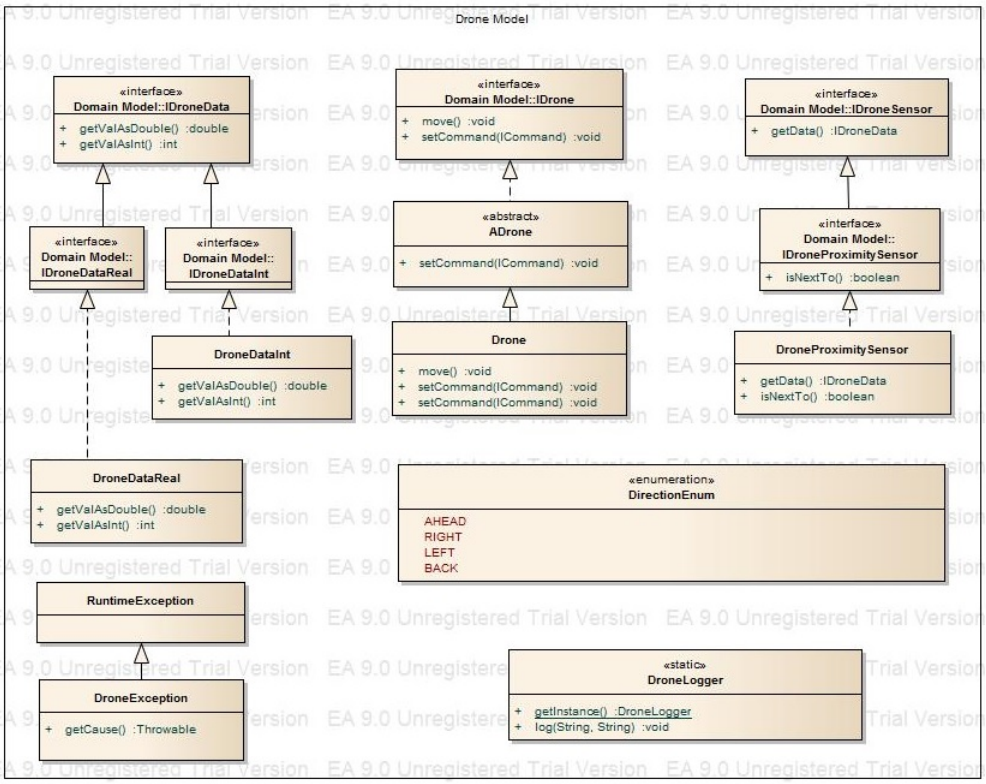
As follows is reported the structure of the middleware subsystem.



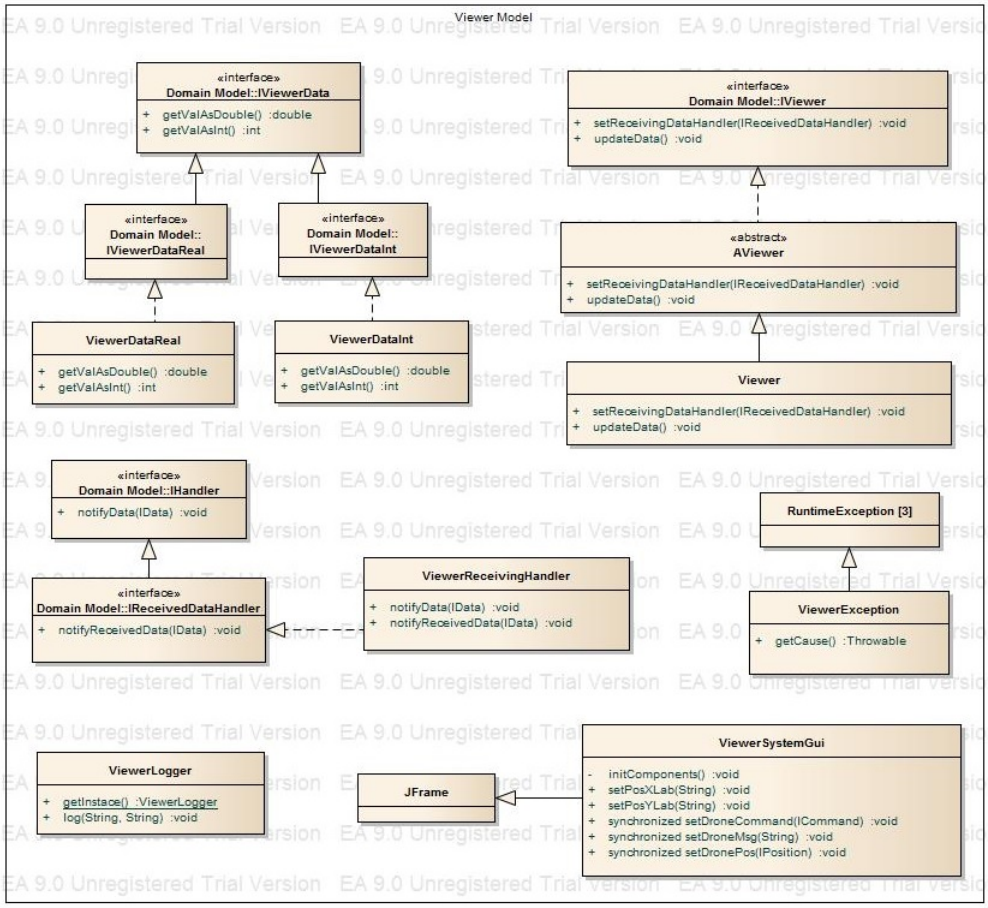
As follows is reported the structure of the masetr subsystem.



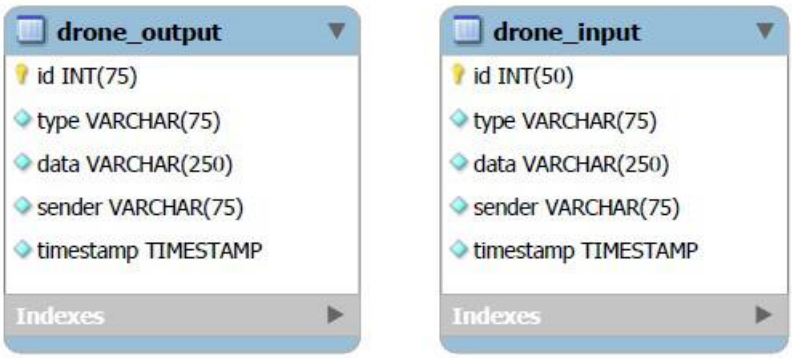
As follows is reported the structure of the drone subsystem.



As follows is reported the structure of the viewer subsystem.



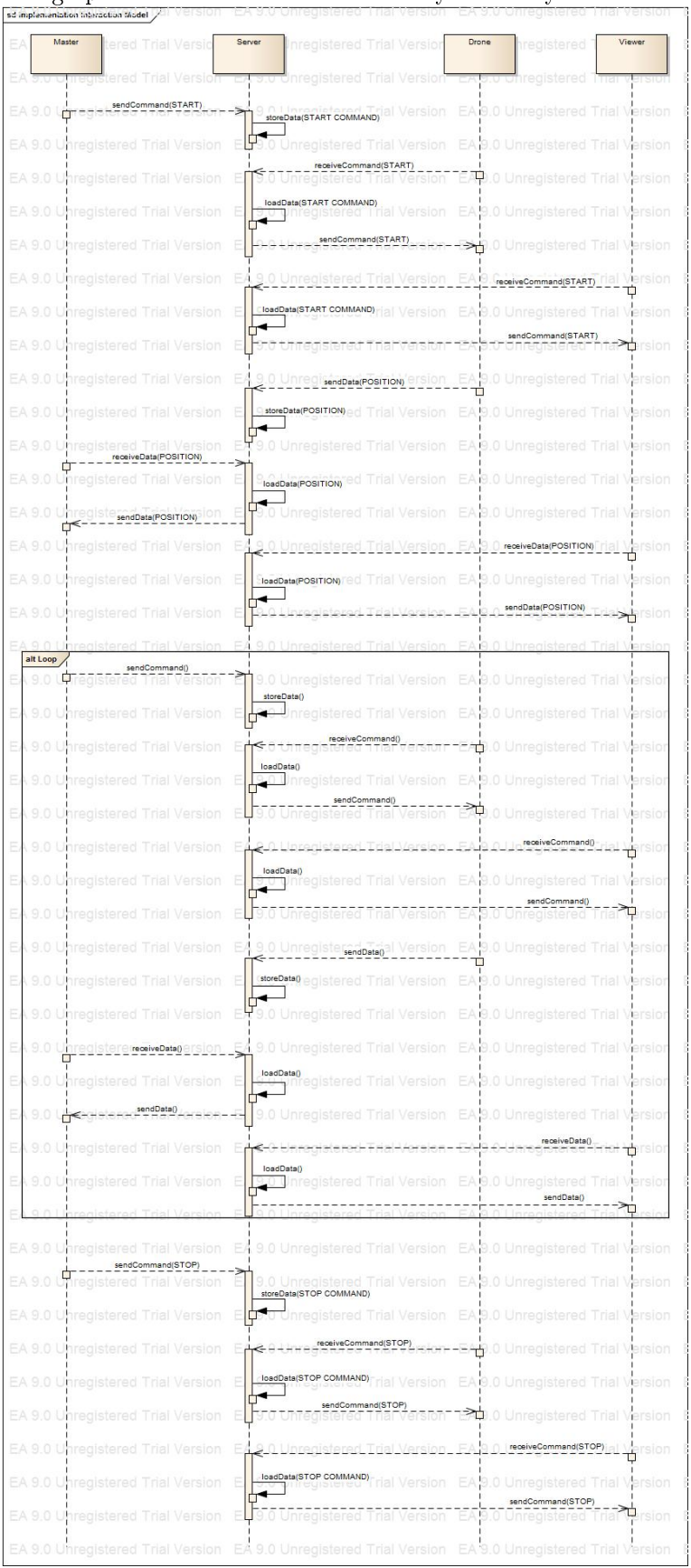
As follows is reported the structure of the Database.



The components in the various diagrams of the structure of the system and subsystems, where there no features are related to libraries and constructs already developed by third parties (eg developed by other software companies like Oracle - Java Standard Library).
The components present in the various diagrams relating to the structure of the system and subsystems, where it appears the syntax "<component Name> [<sequential number>]" relate to the same type of component, the syntax variation in the definition of the name of the component has been done to get a better view of the structure of the system, thus having a higher order in the graphics of the diagram.

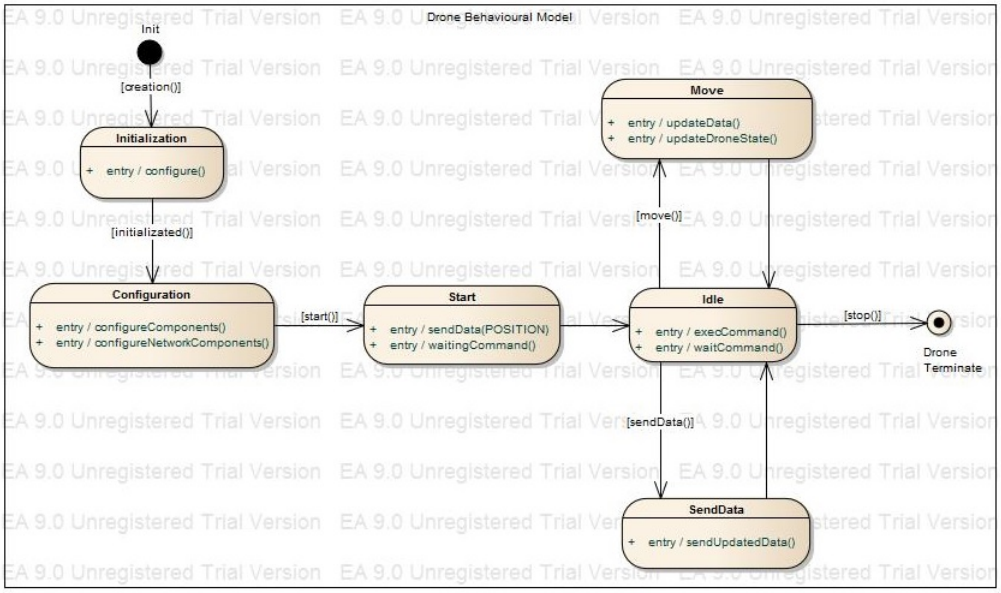
8.2 Interaction

The graphic below show us how the entitys of our system interact each other.

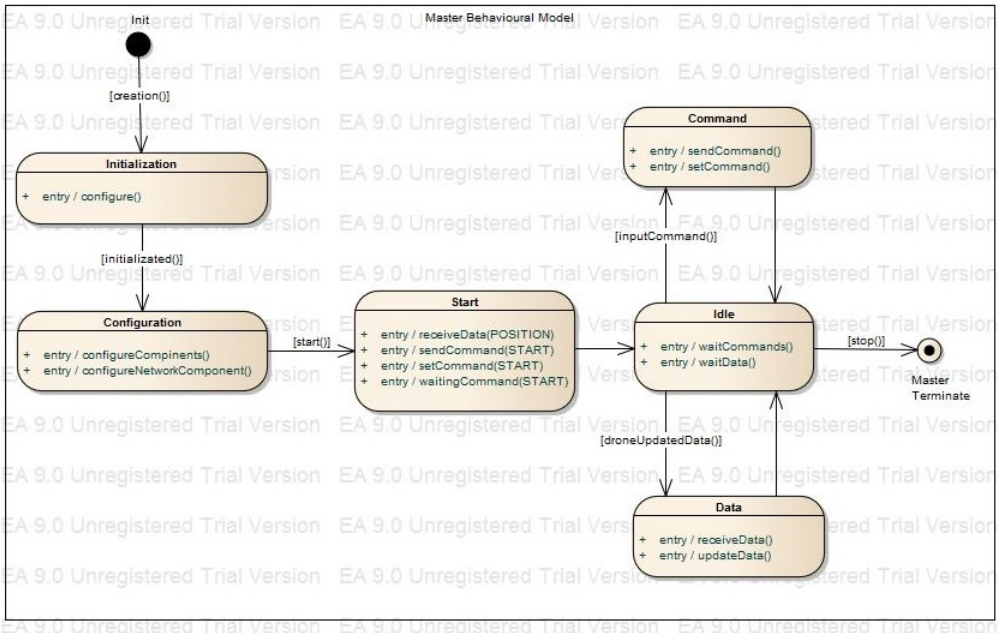


8.3 Behavior

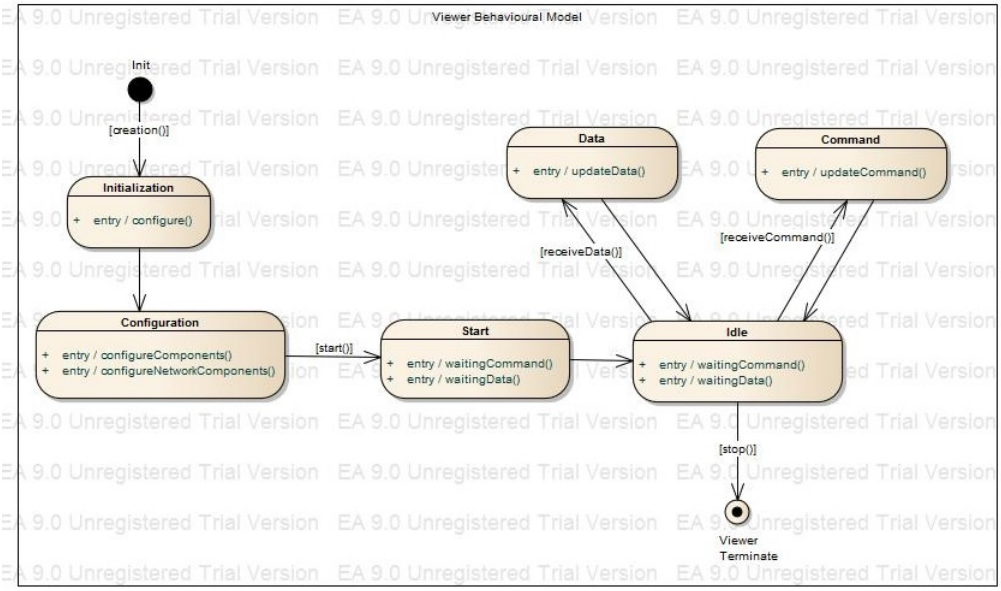
The following illustration shows the behavior of the Drone:



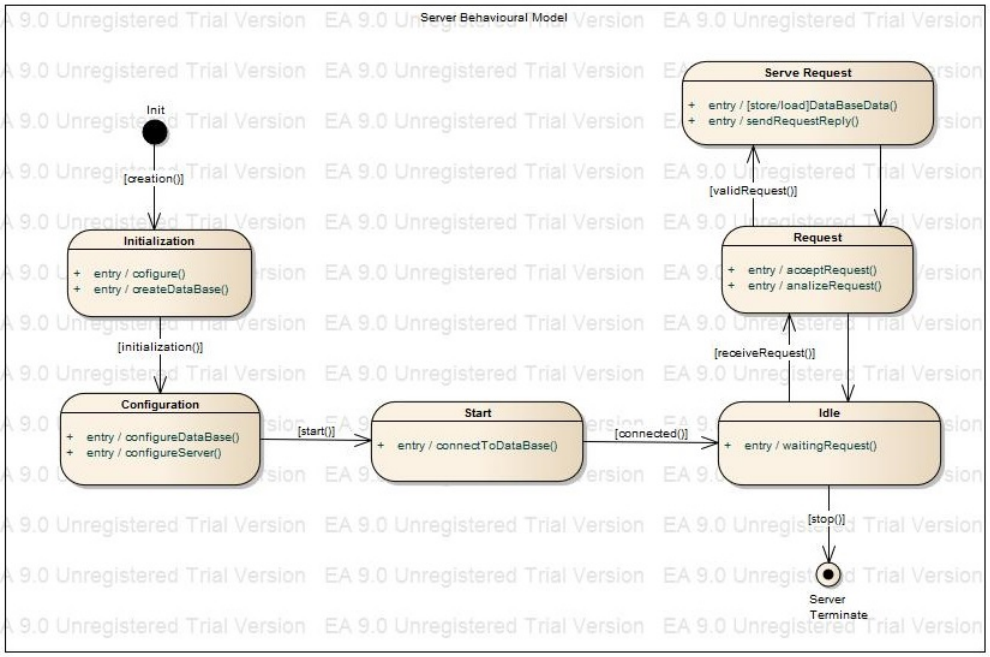
The following illustration shows the behavior of the Master:



The following illustration shows the behavior of the Viewer:



The following illustration shows the behavior of the Server:



9 Implementation

For the implementation of the first version system "Drone Explorer" it was decided to consider the communication between the entities in the following way:

- The Master entity is responsible for sending commands to the drone to its direction of movement. It also deals with the navigation data display (current position) of the Drone, and free blocks and occupied found.
- The Drone entity will take care of the movement based on the command given by the Master entity and, subsequently, to send data on the free blocks or occupied in its path and its location to other entities of the system (Master and Viewer).
- The Viewer entity shall display data of the cruise Drone (current location, and messages related to the free blocks and occupied blocks found).

All communication is through the Server entity that is responsible for receiving data and Drone Master and redirect the data to the entities concerned when needed. The Server entity is also responsible for contain:

- All data that related to the free blocks and occupied blocks found.
- The current position of the Drone.
- The current command sent from the master to Drone.