System of Systems Concept for Effective Oceans to Near Space Observation

Prof. Daniel Hastings, MIT AeroAstro

2020-2023 Seed Project Report

MIT Portugal Partnership 2030

1. PROJECT TEAM

- ***MIT:***
    - *Principal investigator: Prof. Daniel Hastings (Aero Astro)*
    - *Graduate Students: Alice Cooper (Aero Astro)*
    - *Undergraduate Students: Joseph Merkel (Aero Astro), Parker Mayhew (Aero Astro), Angel Gomez (Aero Astro)*
- ***Collaborators in Portugal:***
    - *Industry Partners: N/A*
    - *Research Partners: Joao Tasso de Figueiredo Borges de Sousa, Professor at Faculdade de Engenharia - Porto University; Director of LSTS*

- ***Cornell:***
    - *Principal Investigator: Prof. Maha N. Haji (Sibley School of Mechanical and Aerospace Engineering)*
    - *Graduate Students:* Jiarui (Jordan) Yang (Systems Engineering, MEng), Jinzhou (Kane) Li (Systems Engineering, MEng), Ray Weng (Computer Science, MEng)

2. SUMMARY- 250+ WORDS; BRIEF PROJECT DESCRIPTION

Effective and efficient collection of synoptic data across a range of spatial and temporal scales from oceans to near space will drive the design of a complex system of systems (SoS). The SoS that has to be created will brings together sensing from four levels of platforms from (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles). In addition to long term observations, the SoS must be able to respond to short time scale phenomena with initially unknown spatial distribution. The SoS may involve many assets which will need to be dynamically allocated.

A SoS discrete event simulation was developed that models the physics of the different sensing platforms as well as the sensing capabilities on each platform. A planning and execution control framework will be used to explore a set of different objective functions. The most effective use of the SoS is expected to vary considerably with the choice of objective function.

3. OUTCOMES & ACHIEVEMENTS

MOTIVATION AND OVERVIEW

The vision for this project is to lay the foundations for an Atlantic ocean scientific observation platform, encompassing the three segments: space, air, and ocean (both surface, water column, and sea-bottom). The combination of all these segments allows the synoptic observation of scientific phenomena over a broad range of spatial and temporal resolutions: from large areas from space, to in-situ sensing at the ocean, from large timescales from archival data processing, to the time-efficient deployment of scientific assets on the air and ocean, triggered by real-time remote sensing from space, as well as by in-situ observations. For example, the problem of measuring air-sea fluxes is now receiving a lot of attention.

To accomplish this vision, a system of systems has to be created which sensing from four levels of platforms from (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles). This SoS needs to collect coordinated data over a wide range of spatial and temporal scales. The system concept, as well as a planning and execution control framework or concepts of operation, for ocean to near space observation will be based on networked underwater, surface, air, and space vehicles. The number of assets will be heterogeneous and may be present in large numbers (10s, 100s) endowed with self-organization capabilities and capable of long endurance missions for sustained observation.

The behavior of the SoS is event-driven in nature: once a scientific event is detected an agent needs to be mobilized to collect data at a specific location; once that agent has run of our resources (such as battery or memory) it needs to return to a command ship in order to replenish and a new agent must be sent to collect data; if the scientific event disappears, all agents must return to the command ship. Because of this, the SoS lends itself to being modeled using a discrete event simulation (DES). A DES models the operation of a system as a discrete sequence of events in time. It can also model the interactions between objects, and system operations within the system where these interactions are time-dependent.

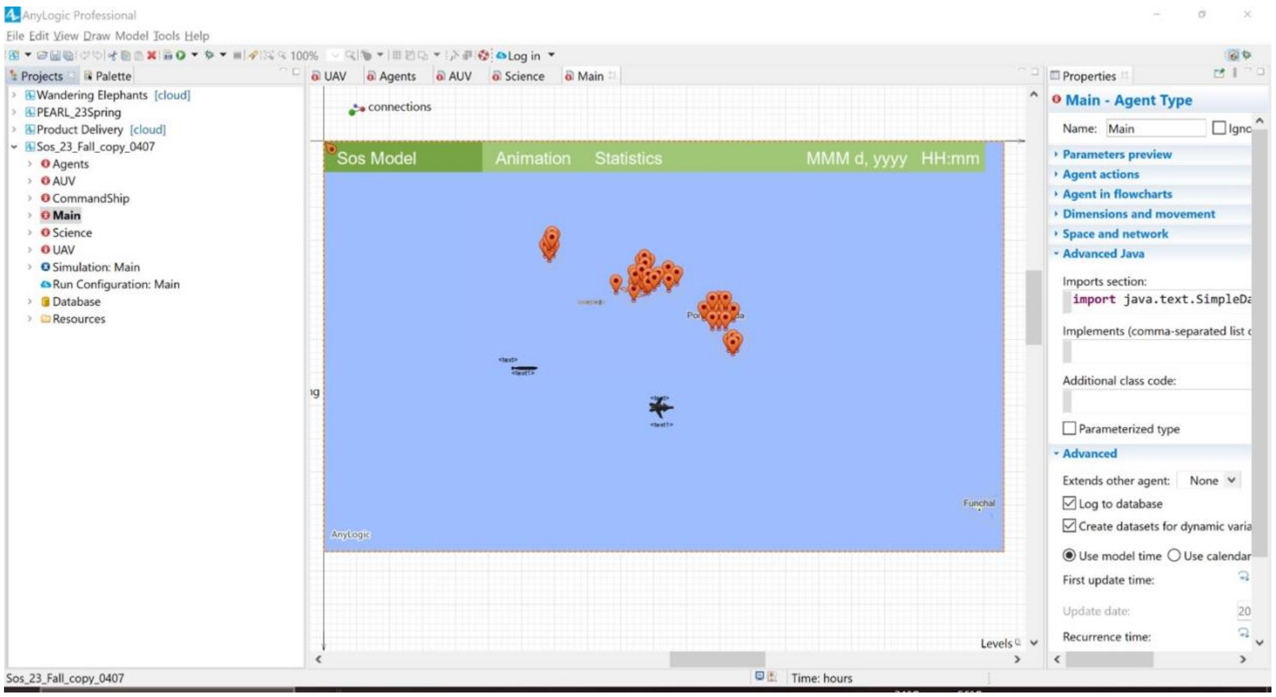
This report details the DES of the proposed SoS for Oceans to Space (satellites, UAVs or gliders, surface ships, ASVs and AUVs) developed so far that can be used to plan the architecture of the system of systems. The planning and execution framework developed using this model will be used to explore and outline the different concepts of operation.

SYSTEM OF SYSTEMS MODEL OVERVIEW

PROJECT BACKGROUND

For the project SoS, the Anylogic Professional Version is the chosen software for the simulation. AnyLogic is a multifaceted simulation modelling tool, characterized by its ability to support system dynamics, agent-based, and discrete event methodologies. It's well-recognized for its flexibility in modelling complex systems and allows users to construct models that more accurately represent real-world systems. In the context of our System of Systems (SoS) project, AnyLogic becomes an invaluable asset.

Employing AnyLogic in our SoS project enables us to create dynamic models of the project's various components, simulating their interactions and optimizing their collective performance. Given the multiple mobile assets involved, from satellites and UAVs to surface and underwater vehicles, it's essential to comprehend how changes in one system influence the others. AnyLogic facilitates this understanding by providing a platform where we can simulate and analyze different parameter configurations, thereby assisting us in identifying the optimal combination that will yield the best performance metrics. In essence, AnyLogic will help us reduce costs, improve operational efficiency, and meet the diverse needs of our stakeholders in the most effective way possible.



Wandering Elephants [cloud]
PEARL_23Spring
Product Delivery [cloud]
Sos_23_Fall_copy_0407
Agents
AUV
CommandShip
Main
Science
UAV
Simulation: Main
Run Configuration: Main
Database
Resources

Main - Agent Type
Name: Main
Parameters preview
Agent actions
Agent in flowcharts
Dimensions and movement
Space and network
Advanced Java
Imports section:
`import java.text.SimpleDa`
Implements (comma-separated list c
Additional class code:
Parameterized type
Advanced
Extends other agent: None
[✓] Log to database
[✓] Create datasets for dynamic varia
(●) Use model time ( ) Use calendar
First update time:
Update date: 20
Recurrence time:

Sos_23_Fall_copy_0407

Figure 1: Sample AnyLogic Screenshot

Project Implementation

In the forthcoming section dedicated to project execution, we delve into the operational aspects of each agent involved - namely, the UAV, AUV, command ship, and science event.

These intricate details are integral to gaining a comprehensive understanding of Project SoS's overall architecture, and how the agents interact with each other in the simulation run.

To begin with, the following flowchart explains the general process of the project SoS.

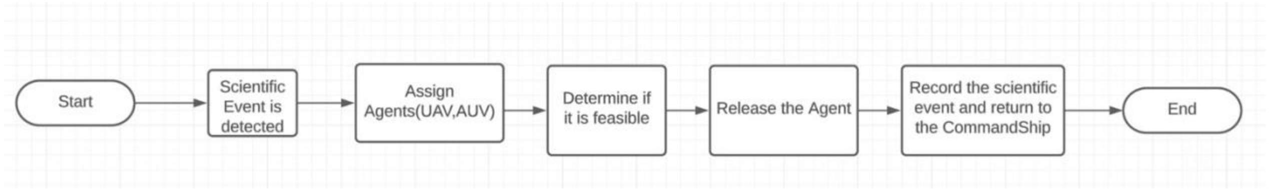

Start
Scientific Event is detected
Assign Agents(UAV,AUV)
Determine if it is feasible
Release the Agent
Record the scientific event and return to the CommandShip
End

Figure 2: General Process of SoS.

The initiation of all activities is predicated on the detection of a scientific event. Upon such detection, a specific agent—either an AUV or UAV—is selected based on the event's depth.

Subsequently, the SoS model verifies whether the chosen agent's current battery capacity can sustain both the outbound and return journeys and if its data storage has ample space to document the event. Once the event recording at the specified location is finalized, the agent embarks on its return journey. The destination is either the command ship or a proximate PEARL station, where it will undergo recharging and offload the collected data.

SCIENCE

To make the scientific measurements requires an effective combination of (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles). Since the assets are mobile, the combination will be inherently dynamic. The model under development detailed in this report will ultimately be used to find the best dynamic combinations depending on the choice of objective function.

Modeling the progression of science throughout the SoS with a process flowchart stood out as the most effective model approach. Representing science as a resource that passes through various processing states is much more representative of science management in the real world than modeling science as an independent agent or dynamic variable. Additionally, AnyLogic has a very broad and capable library for process modeling which allows us to simulate the science lifecycle more accurately than with other methods.

THE SCIENCE AGENT

In the section of the science agent, five variables are created to define the science event’s property. These are as follows:

- Duration: Refers to the temporal length during which the event persists on the map.
- Data Volume: Represents the quantity of data that the scientific event encompasses.
- Type: Classifies the scientific event into one of three categories: 'seamount,' 'front,' or 'fault,' each signifying a distinct type of event within the SoS.
- Location: Specifies the geographic position of the scientific event, as plotted on the GIS Map.
- Name: Assigns a unique string identifier to the scientific event, facilitating tracking and identification.

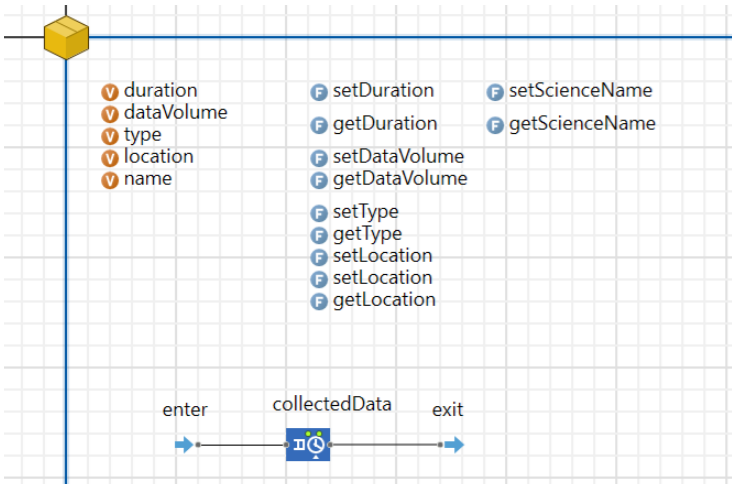


Figure 3: Science Agent in AnyLogic

These variables, encapsulated within the private scope, each possess dedicated setting and receiving methods for assigning or retrieving their values. These variables and functions serve as essential components of the source process library, a pivotal part of the main section within AnyLogic.

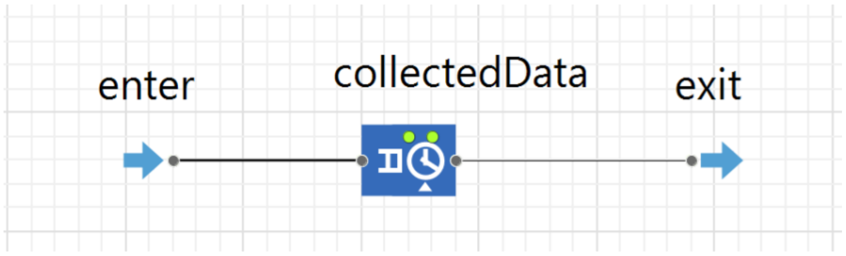
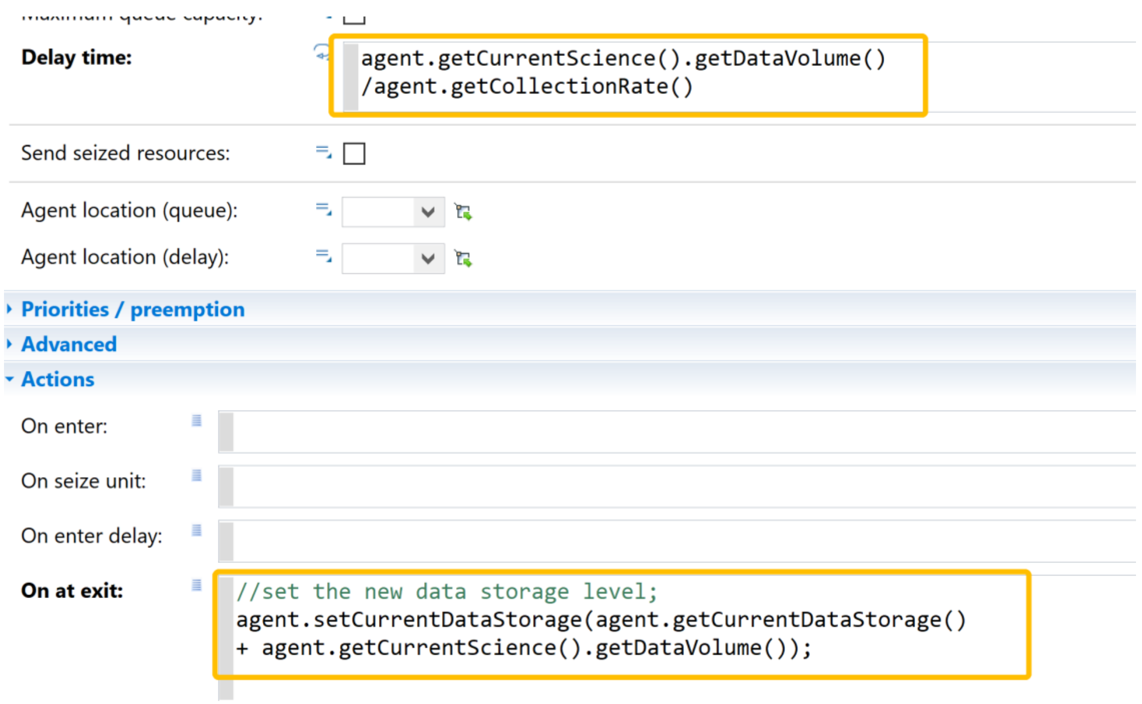


Figure 4:Science Agent General Process in AnyLogic

The aforementioned diagram delineates the typical interaction process between a scientific event and an agent, which can either be an AUV or a UAV. Upon arrival at the current scientific event's location, the agent's progression to the next step — returning to the command ship or a proximate pearl station — is temporarily paused by the scientific event. This hold persists until the data collection procedure associated with the scientific event is successfully completed.



Delay time:

```
agent.getCurrentScience().getDataVolume()
/agent.getCollectionRate()
```

On at exit:

```
//set the new data storage level;
agent.setCurrentDataStorage(agent.getCurrentDataStorage()
+ agent.getCurrentScience().getDataVolume());
```

Figure 5: The Collected Data Section

As depicted in Figure 5, the delay time signifies the duration for which an agent is required to remain stationed at the scientific event. This duration equates to the data volume divided by the data collection rate of the assigned agent, expressed in hours. Upon completion of the stipulated delay time, the agent, with its updated data storage level (in the second section indicated), is expected to depart and return to the command ship for recharging and data offloading.

Science agents are subdivided into three categories based on location of origin, which can be utilized in the system to model different movement patterns of the sciences, which can affect the cost of the system to collect data from it, as well as potential data storage capacity, or the "size" of the science in terms of data storage. These are shown in Fig. 6.

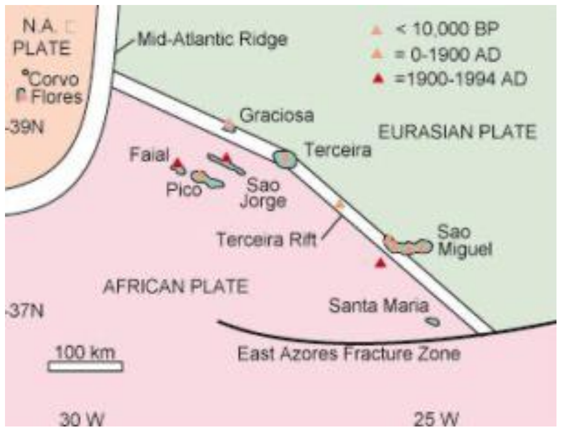


Figure 6: Rough visual showing key geological features near the Azores for approximation in model. (https://www.azoreschoice.com/blog/a-brief-history-of-the-azores-part-two/)

Seamount science: Between the islands Terceira and Sao Miguel and the Eurasian and African tectonic plates is the Joao de Castro Seamount, an underwater volcano that makes up a significant portion of underwater geological activity[1], although specific rates in proportion to the rest of the region's tectonic activity has yet to be concluded and implemented. The seamount is currently where most of the science spawns in the model. These underwater science events are collected by AUVs

Fault line science: The Islands Azores operate close to and functionally on top of a triple junction between the North American, African, and Eurasian tectonic plates. Similar to the seafront, this is an area of significant underwater tectonic activity.[2] These sciences are collected by AUVs

Front science: Oceanic research between agents with differing detection capabilities allows for the collection of data between all depths. The surface of the ocean also has valuable data potential, which in reality can potentially be tied to tectonic activity, but is mainly dependent on climate patterns.[2] These sciences operate just below the surface of the ocean and above, and have the potential to move around the map. These sciences are collected by UAVs

When science enters the model, a step referred to as "spawning", it is assigned a specific location within the GIS space. In order to place science randomly or according to a given distribution, it is helpful to create AnyLogic GIS regions. These regions are used to focus the possible spawning locations for sciences within a desired area. Additionally, regions play a major role in path planning (discussed below). Regions are currently implemented using the AnyLogic space markup palette, and are drawn by hand on the GIS map to meet our needs.

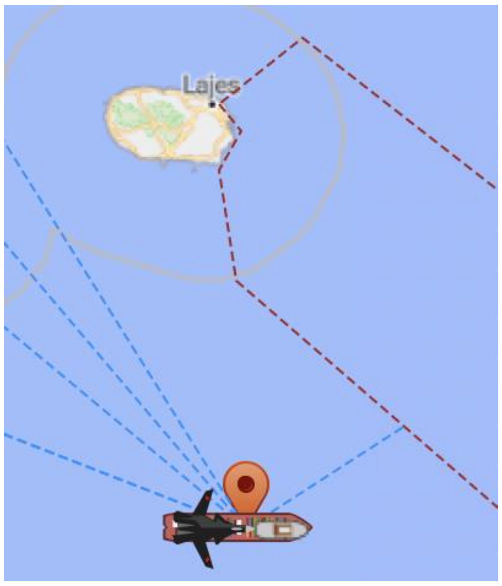


Figure 7: Image showing map used in AnyLogic with GIS regions outlined in red dashed lines. The dashed blue lines indicate paths that AUVs must follow to and from the GIS regions.

[1] Storch, B., Haase, K.M., Romer, R.H.W. *et al.* Rifting of the oceanic Azores Plateau with episodic volcanic activity. *Sci Rep* 10, 19718 (2020). https://doi.org/10.1038/s41598-020-76691-1

Seamount spawns in a two-dimensional gaussian distribution around the geographical center of the seamount with a rate of 1 instance per hour, which is subject and readily available to change given any change in information regarding activity rate. Our team chose 1 instance per hour because it gives a good representation of the shape of the seamount in terms of the underwater scientific activity it creates. Similar to this, the fault science spawns in a gaussian distribution around the geographical fault lines modeled by linear equations, with a spawn rate of 0.2 per hour. Currently, front sciences are stationary and spawn randomly around regions that are not on land. These sciences also have a spawn rate of 0.2 per hour. Currently, 0.2 spawns per hour is a placeholder number until more accurate data regarding oceanic activity can be incorporated into the model.

THE UAV AND AUV AGENTS

Given the substantial overlap in the functionalities of the UAV and AUV agents, we have designed a parent 'Agents' class as depicted above. This class will serve as a foundation for the individual UAV and AUV subclasses, which will be elaborated on subsequently. Within the scope of the SoS model, 'Agents' embody all entities tasked with data collection from scientific events, such as UAVs and AUVs.

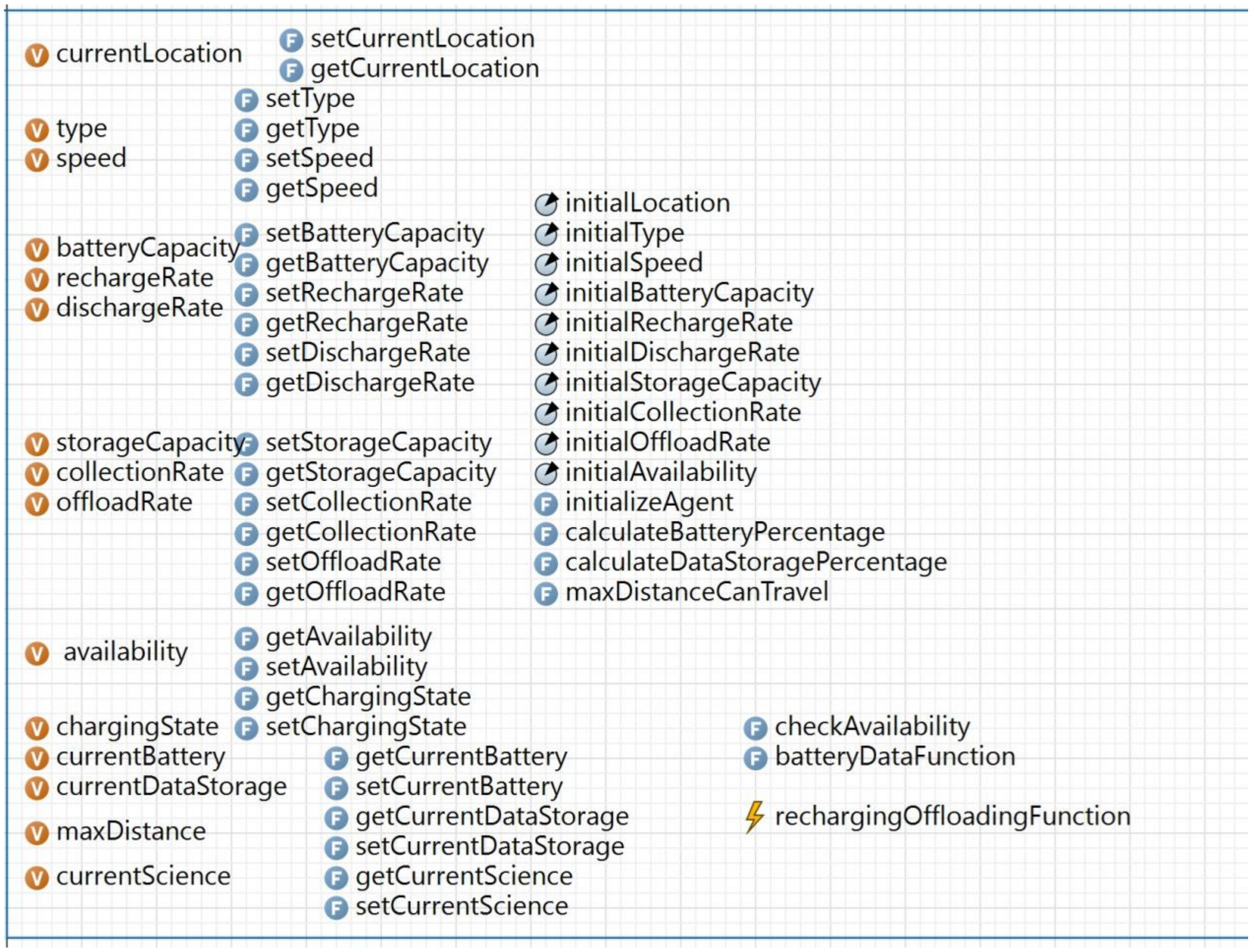


Figure 8: Agents Section in AnyLogic

The agent has several variables defined as follows:

- currentLocation: represents the current location of the agent(UAV, AUV)
- type: a string format to indicate the type of agent, either "AUV" or "UAV"
- batteryCapacity: the total battery capacity that the agent can have.
- rechargeRate: the amount of power that the agent can restore per hour if it is at the docking station.
- dischargeRate: the amount of power that the agent loses per hour if it is not at the docking station.
- storageCapacity: the amount of space that the agent can hold for the data.
- collectionRate: the amount of data that the agent can collect per hour.
- offloadRate: the amount of data that the agent can offload per hour.
- availability: the availability of the agent in boolean format.
- chargingState: a boolean value indicates if the agent is in the charging state.
- currentBattery: the current Battery level that the agent currently holds.
- currentDataStorage: the amount of data that the agent currently holds.
- maxDistance: the maximum distance in km that agent can travel based on the agnet's current battery level.
- currentScience: the scientific event that the current agent is pursuing.

Varibles are also put under the private scope to ensure better control and safety. Every variable has its unique getter and setter functions for retrieving and changing values. Conversely, the class incorporates several functions, including calculateBatteryPercentage and calculateDataStoragePercentage, among others. These functions are responsible for assessing the current state of the agent's data and battery reserves, primarily for visualization purposes. This functional representation is further elucidated in the ensuing figure:

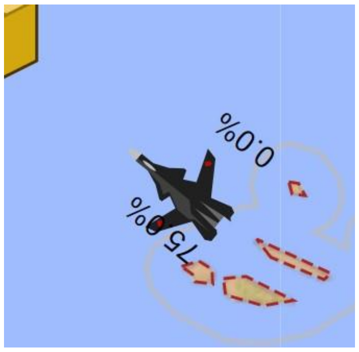
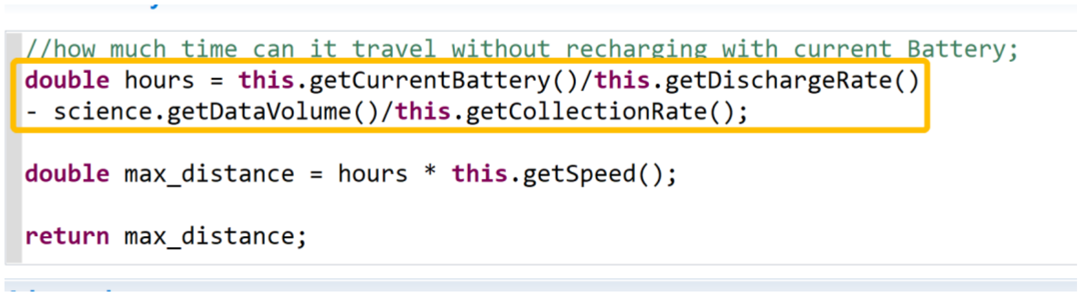


Figure 9: Agent Visualization.

As shown above, the bottom left number represents the available battery of the agent, and the number at the top right represents the amount of data the agent is holding.

A crucial consideration is guaranteeing the agent's ability to complete a trip safely with its available battery resources. This entails ensuring the power level never dips below zero.

Under such consideration, the distance to the current Science from the location of the agent and the maximum distance that the agent needs to be compared, thus the function "maxDistanceCanTravel" is created as follows:

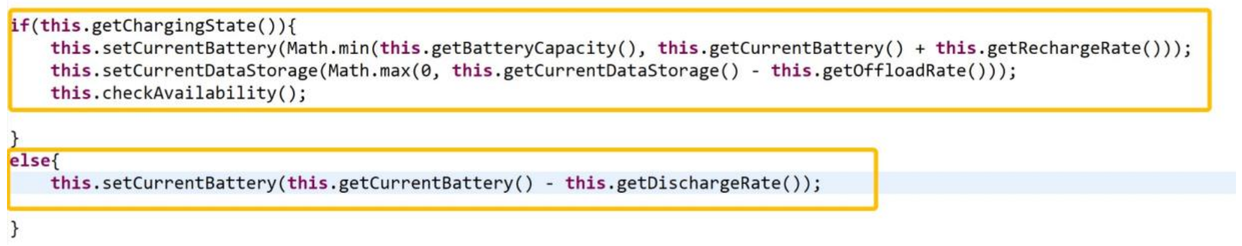


```
//how much time can it travel without recharging with current Battery;
double hours = this.getCurrentBattery()/this.getDischargeRate()
- science.getDataVolume()/this.getCollectionRate();

double max_distance = hours * this.getSpeed();

return max_distance;
```

Figure 10: maxDistanceCanTravel Function

In the highlighted 'maxDistanceCanTravel' section, time available for travel is represented in float format as 'hours'. This is calculated by subtracting data collection hours from the total available hours before recharging is required. The maximum distance that can be covered is then computed by multiplying the travel hours with the agent's speed in km per hour.

Moreover, agents invariably expend power once they depart from charging stations such as the command ship or PEARL station (see PEARL section). Concurrently, while recharging at these stations, the agents also offload the accumulated data. To continually monitor the battery level and data storage of these agents, a dynamic event named "rechargingOffloadingFunction" has been established. This event recurrently initializes another function once per hour, "batteryDataFunction", which systematically discharges the agent's battery during transit away from the charging station and initiates recharging and data offloading when the agent is stationed at the charging point.



```
if(this.getChargingState()){
    this.setCurrentBattery(Math.min(this.getBatteryCapacity(), this.getCurrentBattery() + this.getRechargeRate()));
    this.setCurrentDataStorage(Math.max(0, this.getCurrentDataStorage() - this.getOffloadRate()));
    this.checkAvailability();

}
else{
    this.setCurrentBattery(this.getCurrentBattery() - this.getDischargeRate());

}
```

Figure 11: batteryDataFunction Function

Figure 11 is divided into two distinct sections: the first illustrates the agent's status during charging, while the second depicts the agent engaged in a mission for a scientific event.

Importantly, when the agent is at the charging station and its battery and data storage levels exceed 80%, its status transitions to "available". This signifies that the agent is fully prepared and ready to be assigned to upcoming scientific events.

THE UAV AGENT

An UAV agent is assigned to any scientific event with a depth of lesser than 10 meters.

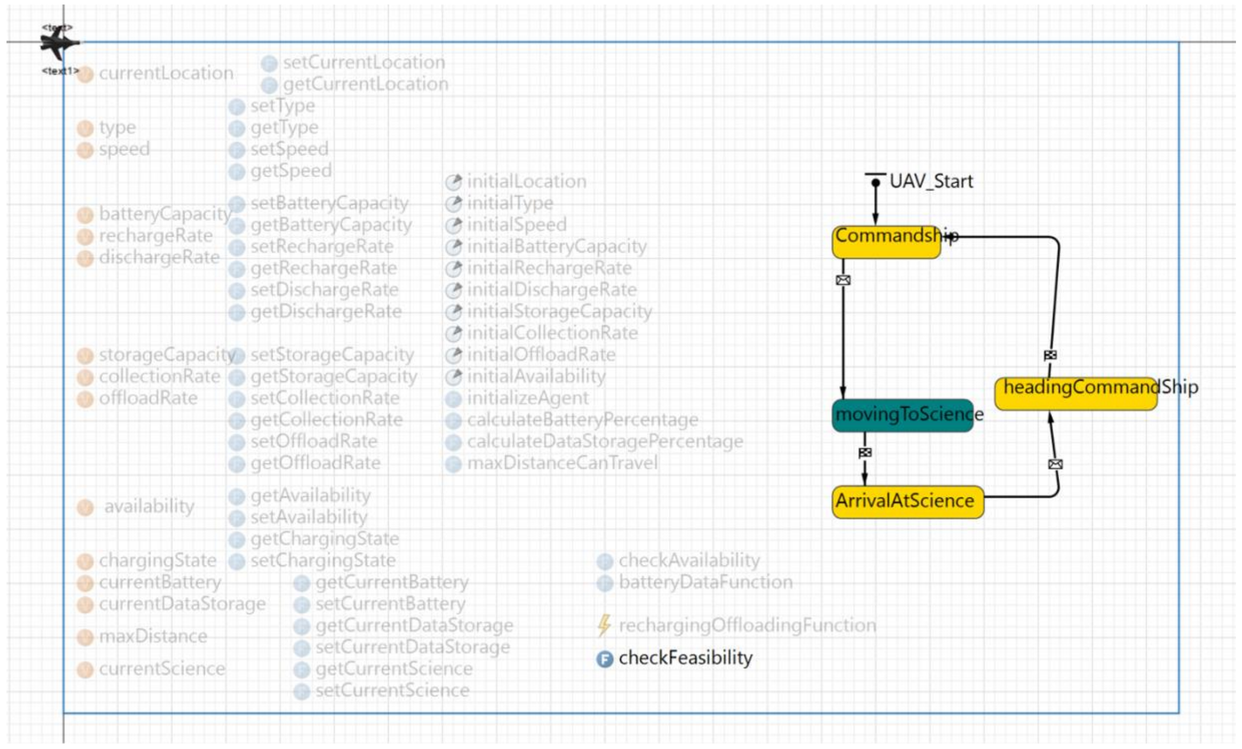
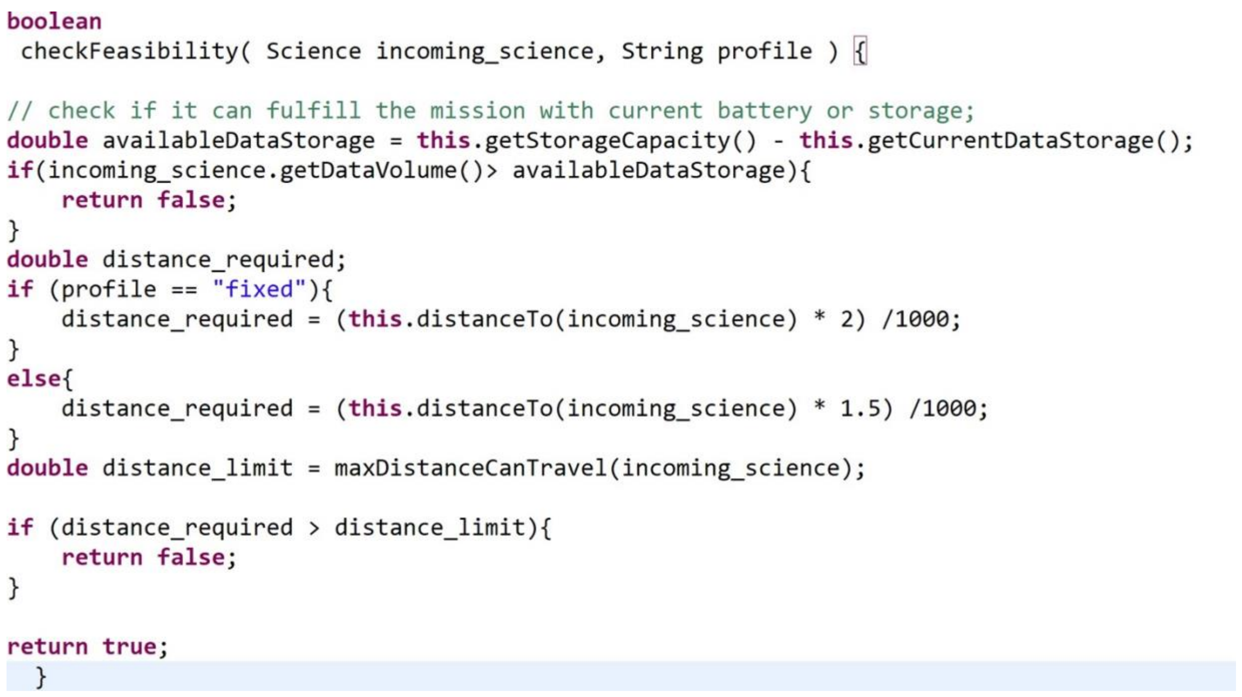


Figure 12: UAV Agent Class

As shown in Figure 12, the UAV agent class has only one extra function named as "checkFeasibility" and inherited every other function from Agents, the parent class.

As depicted in the accompanying figure, the 'checkFeasibility' function utilizes two parameters: the incoming science event assigned to the agent and the profile of the command ship, which can either be "fixed" or "center of mass". When the profile is "fixed", it implies that the command ship's location remains constant. Consequently, the function requires the maximum distance (in km) an agent can travel to exceed twice the distance to the science event's location. Contrastingly, if the profile is the "center of mass", the command ship progressively moves towards the scientific events. In this scenario, the function must validate that the agent possesses adequate battery power to cover 1.5 times the distance to the scientific event's location.

The UAV Agent operates in four unique states: "atCommandShip", "movingToScience", "ArrivalAtScience", and "headingCommandShip". Initially, every UAV agent is stationed at the command ship's location, existing in the "atCommandShip" state. Upon the detection of a scientific event with a depth of lesser than 10 meters, the UAV agent transitions into the "movingToScience" state and is directed toward the event. Upon arrival, signified by the "ArrivalAtScience" state, the scientific event retains the agent and imposes a delay until data collection is complete. Subsequently, having concluded data collection, the agent reverts to the "headingCommandShip" state, returning to the command ship for battery recharging and data offloading.



```
boolean
 checkFeasibility( Science incoming_science, String profile ) {

// check if it can fulfill the mission with current battery or storage;
double availableDataStorage = this.getStorageCapacity() - this.getCurrentDataStorage();
if(incoming_science.getDataVolume()> availableDataStorage){
    return false;
}
double distance_required;
if (profile == "fixed"){
    distance_required = (this.distanceTo(incoming_science) * 2) /1000;
}
else{
    distance_required = (this.distanceTo(incoming_science) * 1.5) /1000;
}
double distance_limit = maxDistanceCanTravel(incoming_science);

if (distance_required > distance_limit){
    return false;
}

return true;
  }
```

Figure 13: checkFeasibility Functionality

The AUV Agent

An AUV agent is assigned to any scientific event with a depth of greater than 10 meters.

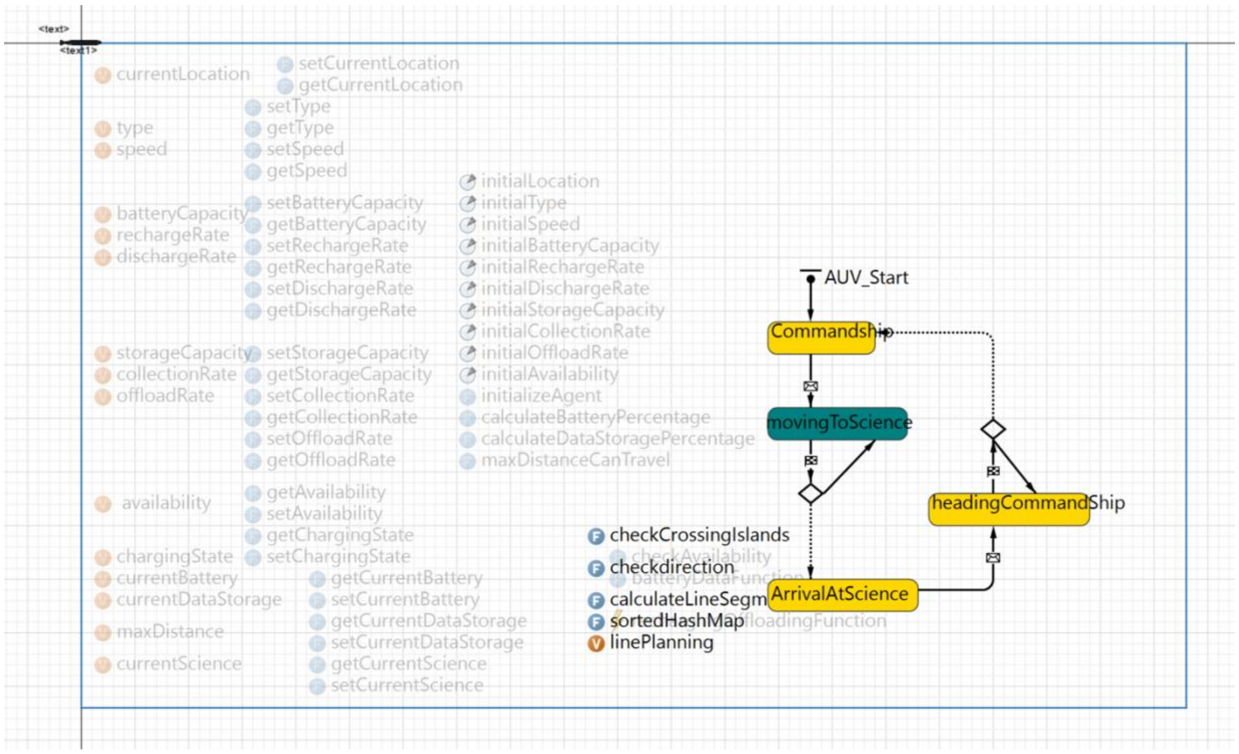


Figure 14: AUV Agent Class

As illustrated in Figure 14, the navigation process for AUV agents is more intricate compared to UAV agents. This complexity arises from the fact that AUV agents cannot directly traverse islands, necessitating the implementation of a rerouting algorithm to discover the optimal alternate route that circumvents these obstacles. Interestingly, this situation is not unique to AUV agents; the command ship with the "center of mass" profile also encounters similar circumstances. Therefore, we have devised an algorithm for generating such routes, which is expounded upon in the following section.

For each island displayed on the map, we have established cardinal points: North, South, East, and West. These points are considered alternative detour markers in the event that either an AUV agent or the command ship needs to bypass the island.

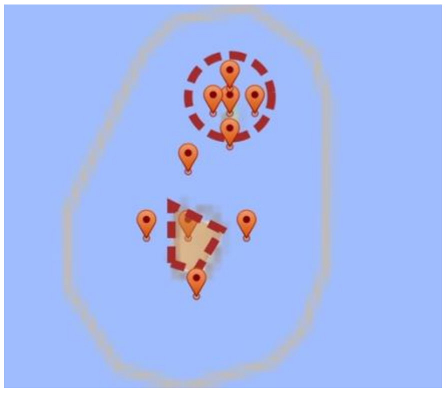


Figure 15: Two islands on the map

Each island possesses a unique collision radius. Upon detection of a potential collision, a specific set of points is taken into account based on the AUV agent's direction of travel. West and East points are collectively evaluated as one group, while North and South points constitute another group. When an AUV travels from East to West or vice versa, North and South points are brought into consideration for potential bypass routes. Conversely, when an AUV moves from South to North, the East and West points are evaluated. The decision for the optimal point to navigate towards is made by calculating the sum of each point's distance to the starting location and the science location, with preference given to the point yielding the shortest overall distance. This ensures an effective bypass strategy is generated to avoid the island, and it also works on multiple islands as long as 4 detour points are defined.

Figure 14 illustrates the function "checkCrossingIslands", devised to pinpoint potential island obstructions an agent might encounter if it adopts a direct, straight-line route to the science location. Additionally, the "checkDirection" function is employed to ascertain the current heading direction of the agent. The function "calculateLineSegments", on the other hand, aids in identifying suitable detour points for consideration, thus facilitating strategic route planning.

Upon establishing feasible detour points and verifying the agent's capacity for outbound and return trips, the AUV agent transitions from the "CommandShip" state and advances towards each detour point in the "movingToScience" phase. Upon reaching the scientific event, signified by the "arrivingAtScience" state, the agent is detained by the event until the data collection procedure concludes. Subsequently, once data collection is finished, the agent employs the same route optimization mechanism to identify the optimal detour paths back to the command ship for recharging and data offloading.

The Command Ship

The command ship serves as a crucial nexus, linking all operational components together.

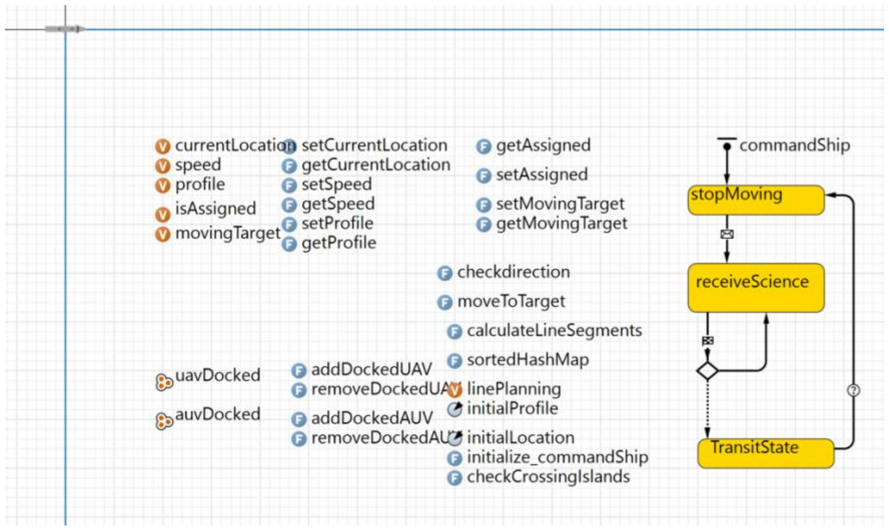


Figure 16: the CommandShip Class

The command ship operates under two distinct profiles that can significantly influence other agents like UAVs and AUVs. The first, termed as "fixed", signifies a stationary command ship that does not relocate. The second, labelled as "center of mass", denotes a dynamic command ship that continually maneuvers towards the epicentre of active scientific events, thereby optimizing the energy usage of the agents.

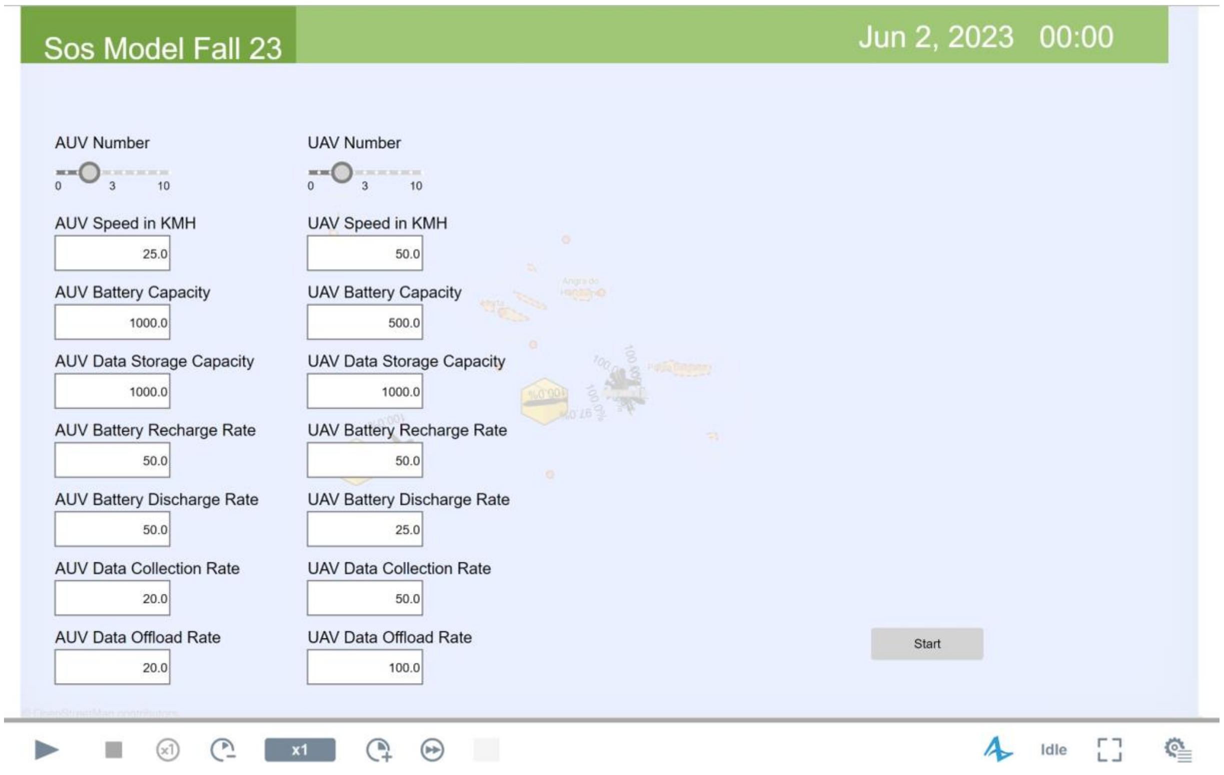
The command ship has the following variables defined in the private scope but each of them has a unique getter and setter functions for retrieving and getting its value.

- currentLocation: the current location of the command ship in lat and lon format.
- speed : the speed of the command ship
- profile: the profile of the command ship
- uavDocked: a list of uav that is currently docking at the command ship
- auvDocked: a list of auv that is currently docking at the command ship

As displayed in Figure 16, the command ship has four distinct states when operating under the "center of mass" profile. Upon confirmation and determination of the reachability of a scientific event by either AUV or UAV agents, the command ship employs the same algorithm used by the AUV to calculate its path to the event's central location. Upon reaching this point, it remains stationary until the roster of active scientific events is refreshed or updated.

Simulation and Visualization

As detailed in the "Project Implementation" section, numerous variables that can be optimized based on cost have been defined, such as the quantity of AUVs and UAVs, as well as their storage and battery capacities, among others. Consequently, I have developed a dedicated simulation page, as outlined below:



AUV Number: 3 (0–10)
UAV Number: 3 (0–10)

AUV Speed in KMH 25.0	UAV Speed in KMH 50.0
AUV Battery Capacity 1000.0	UAV Battery Capacity 500.0
AUV Data Storage Capacity 1000.0	UAV Data Storage Capacity 1000.0
AUV Battery Recharge Rate 50.0	UAV Battery Recharge Rate 50.0
AUV Battery Discharge Rate 50.0	UAV Battery Discharge Rate 25.0
AUV Data Collection Rate 20.0	UAV Data Collection Rate 50.0
AUV Data Offload Rate 20.0	UAV Data Offload Rate 100.0

Figure 17: the Simulation Page

On the Simulation page, users have the flexibility to customize these variables according to their specific preferences or requirements.

Regarding data visualization, we have designed a representative time plot as shown below for user reference:

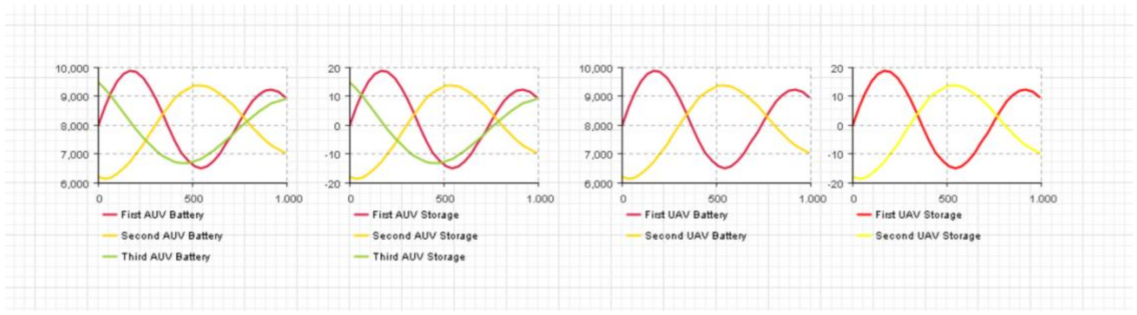


Figure 18: the visualization plots

These four distinct plots respectively represent the battery levels and data storage capacities for each AUV and UAV, providing clear and concise visualization of these crucial parameters.

Optimization

The model can be run to determine the optimal number of UAVs and AUVs. The objective function was to minimize the loss of the (random) science while making maximal use of the AUVs and UAVs. Each of the cases run represents a different starting point for the command ship, UAVs, and AUVs. Figs. 19-22 show the cases and the table below details the results of the optimal number of UAVs and AUVs in each case. Case 3 requires more assets since the AUVS need to get around the islands.

CASE	**LOCATION**	**OPTIMAL NUMBER OF UAVS, AUVS**
Case 1	38.36N, 30.17W	UAVs = 5 AUV s = 9
Case 2	38.31 N 26.1W	UAVs = 5 AUV s = 9
Case 3	40.7N, 27.59W	UAVs = 10 AUV s = 9
Case 4	37.33N, 27.59W	UAVs = 5 AUV s = 9

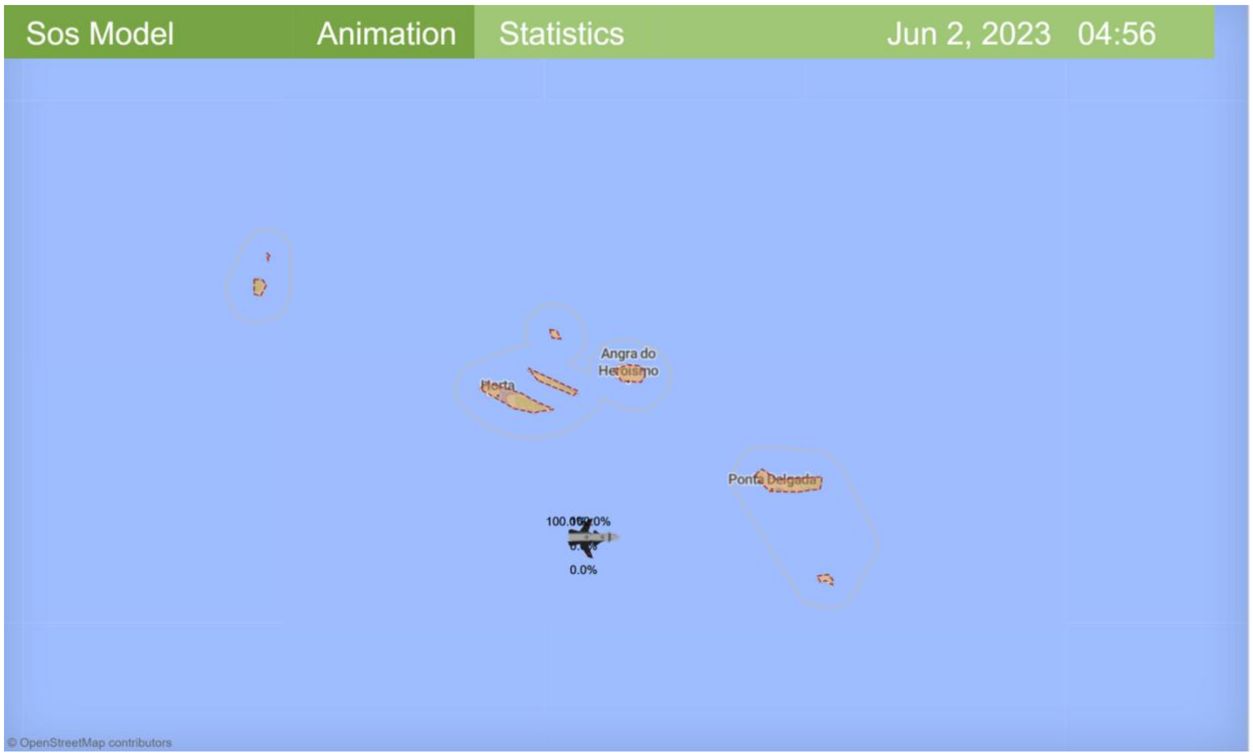


Figure 19: Optimization case 1

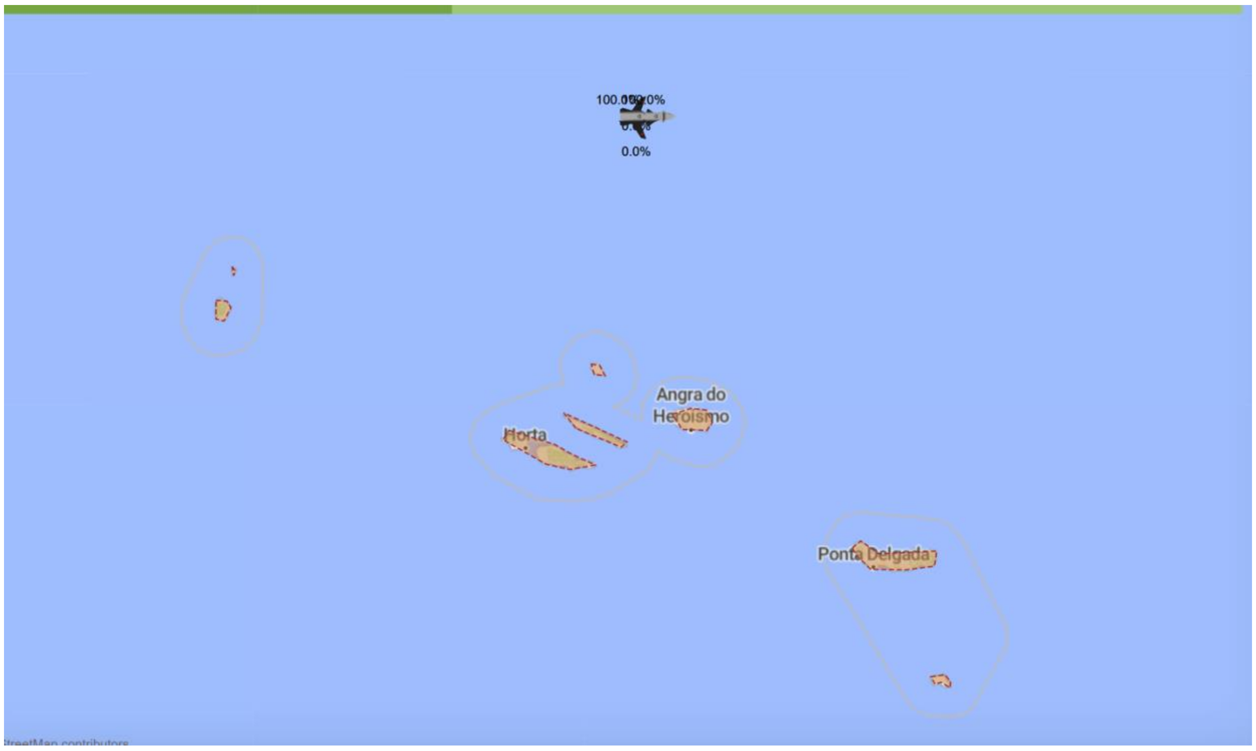


Figure 20: Optimization case 2

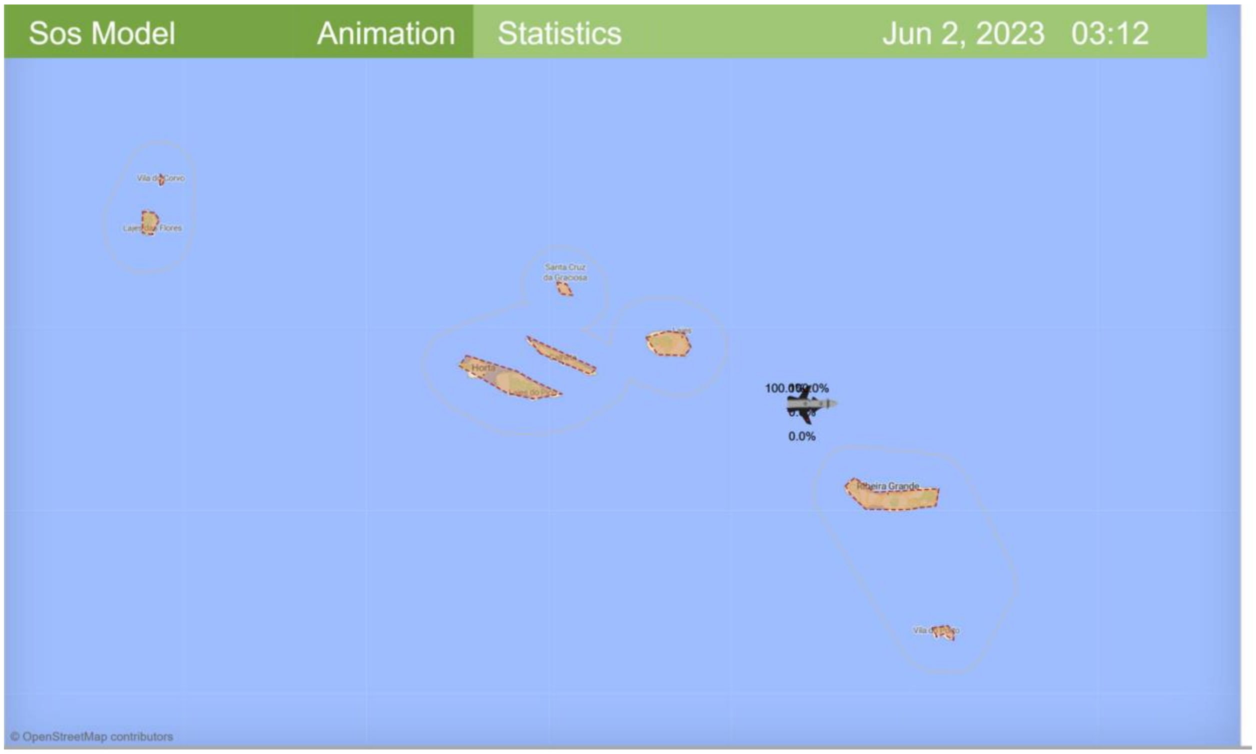


Figure 21: Optimization case 3

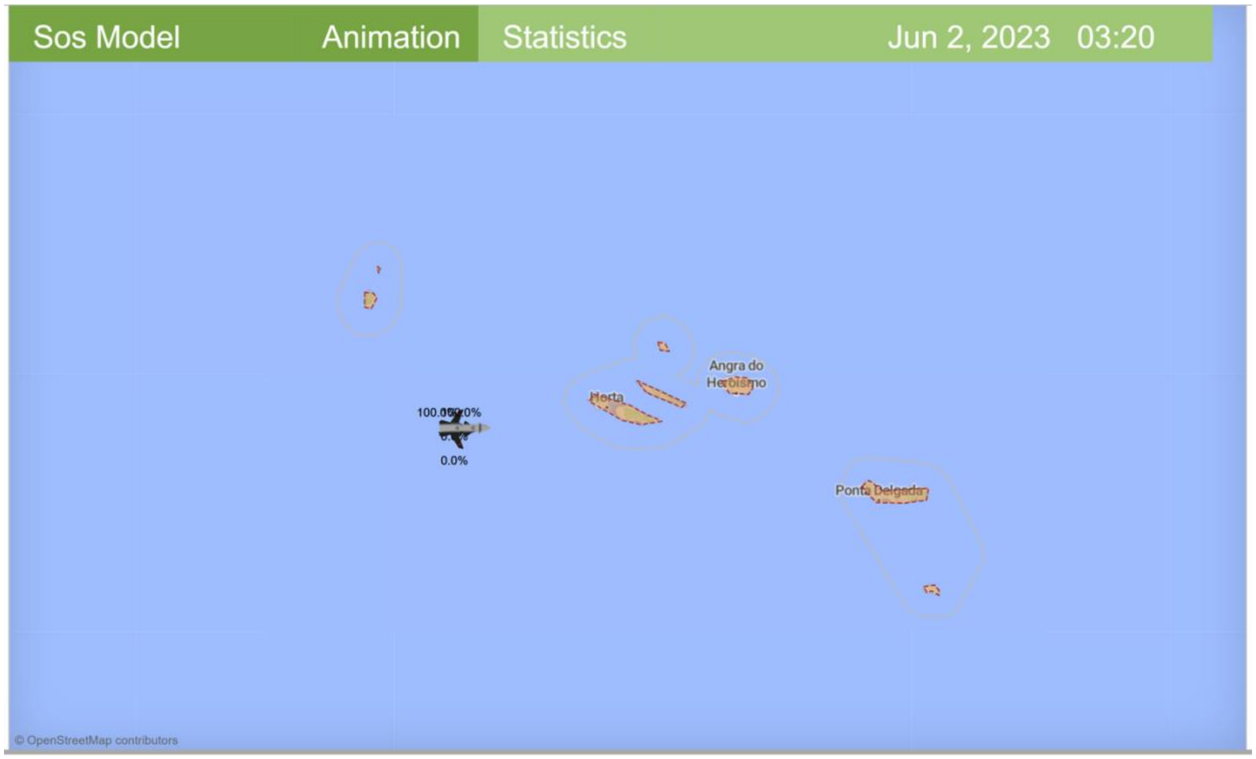


Figure 22: Optimization case 4

Platform for Expanding AUV exploRation to Longer ranges (PEARL)

Model Overview

In this section we detail the development of another DES using AnyLogic for PEARL. Autonomous Underwater Vehicles (AUVs) offer the ability to provide persistent and expanded ocean observations and measurements. The battery capacity of AUVs, however, currently limits the range and duration of missions. At the same time, sensor payload and ocean measurement resolution are limited by data storage space onboard AUVs. These limitations require that AUVs be frequently recovered to recharge and offload data, a process that often requires the assistance of a support vessel and crew and can cost in excess of $30,000 per day. The Platform for Expanding AUV exploRation to Longer ranges (PEARL) developed in the Engineering Systems Lab, can extend the range and endurance of AUVs (e.g. from 8 hours to 240 hours), while reducing data latency and operating costs. PEARL is an integrated autonomous floating docking station which simultaneously provides AUV battery recharging and data uplink via the new generation of high-bandwidth low- Earth orbit (LEO) constellations (OneWeb). PEARL can ensure worldwide connectivity and control of AUVs, allowing for near-real-time underwater data from across the globe. By utilizing solar power paired with integrated battery modules, PEARL can harvest power during daytime hours, allowing reliable, on-demand recharging of vehicles and data transfer, effectively extending AUV endurance and return. Fig. 23 shows an example use-case for PEARL in the open ocean.

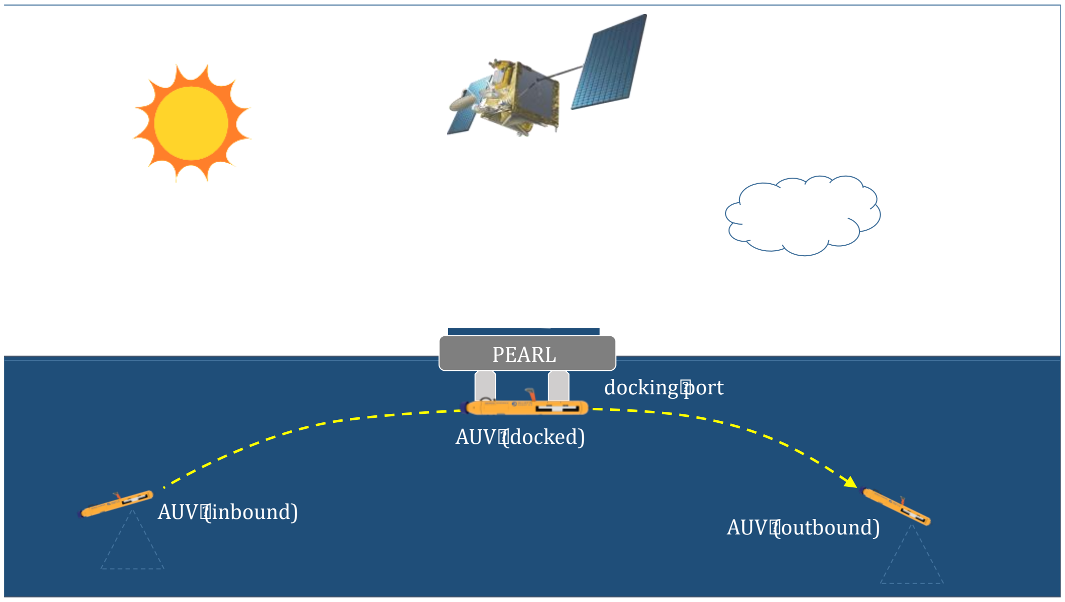


Figure 23: An inbound AUV can dock with PEARL, which simultaneously recharges the vehicle using renewable solar power and offloads data via a high-bandwidth LEO satellite link, enabling the AUV to conduct longer-range missions and collect higher resolution ocean measurements.

As with the SoS, an agent-based model was developed to understand how PEARL could enable AUVs to achieve longer-range missions Each of the agents in the PEARL model are detailed below.

AUV Agents

The AUV agents, shown in Fig. 24, have an "idle" state for when it is just sitting in the ocean or sitting at PEARL without charging. Each AUV also has a "MovingToPEARL" state when it is moving to PEARL, and a "MovingToScience" state when it is moving to a scientific event. If the AUVs are in the "AtPEARL" state, their batteries will recharge at their recharge rate until their batteries are fully charged. In the same way, data will be offloaded from AUVs when they are in the "AtPEARL" state at their offload rate until their available data storage is restored to their full data storage capacity. "AtScience" means the AUV has arrived at the location of the scientific event and is performing data collection.

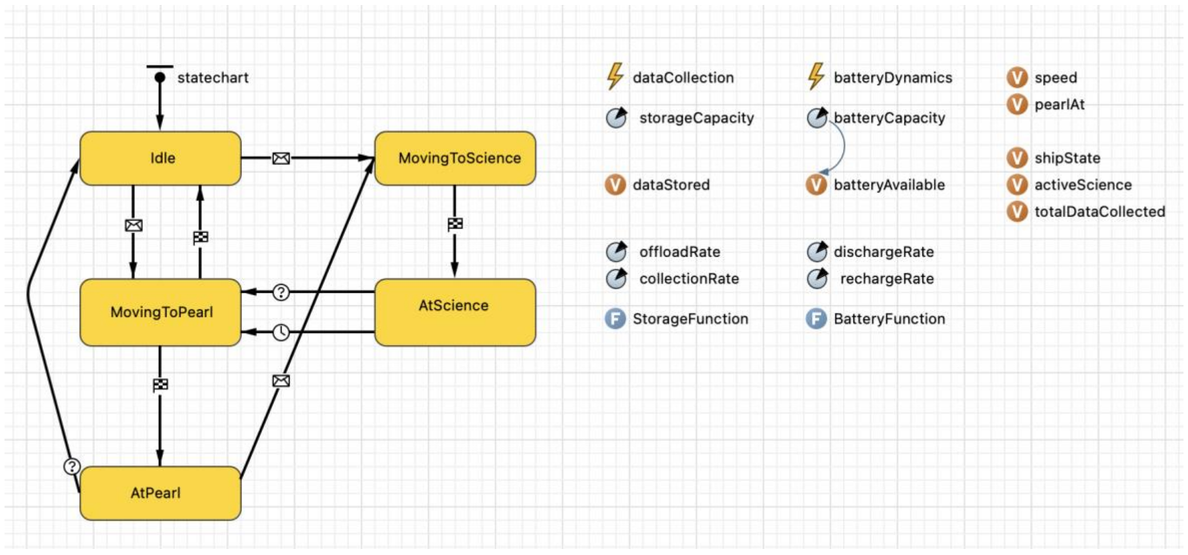
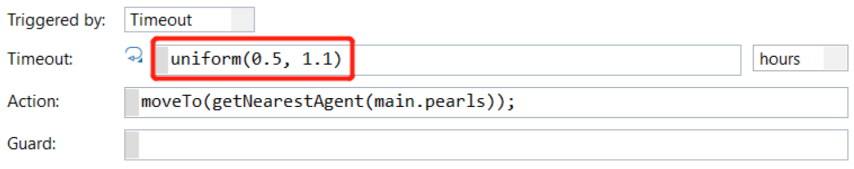


Figure 24: State chart for AUV agents

In the "AtScience" state of the AUV model, there are two transitions that can prompt the AUV to return to the PEARL ("MovingToPEARL"):

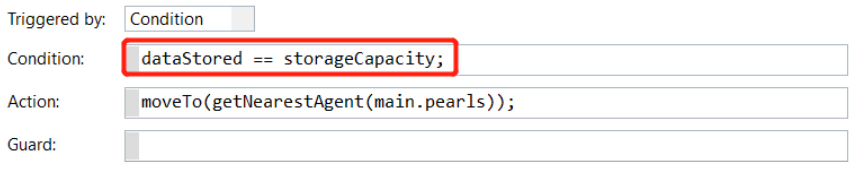
- **Event "end"**: This event gets triggered when the AUV finishes its data collection at the science station. The time spent at the science station is modeled as a **uniform distribution**, representing variability in the data collection process. (This part can be buggy because the time to collect data is always short and the amount collected is about 10% of the total capacity. Most of the power is consumed on the way to and from.)



Triggered by:	Timeout
Timeout:	`uniform(0.5, 1.1)` hours
Action:	`moveTo(getNearestAgent(main.pearls));`
Guard:	

Figure 25: Event end condition or AUV agents in PEARL model.

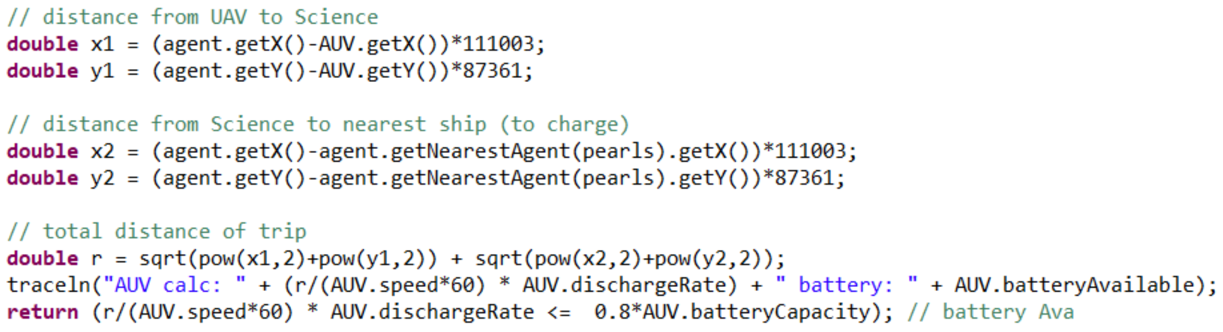
- **Full storage**: The second transition occurs when the AUV's storage is completely utilized. Once the AUV's storage is full, it will execute a command to return to the PEARL.



Triggered by:	Condition
Condition:	`dataStored == storageCapacity;`
Action:	`moveTo(getNearestAgent(main.pearls));`
Guard:	

Figure 25: Storage full condition for AUV agents in PEARL model.

In addition to that, the system should calculate whether there is enough power to perform the task before sending the UAV. This code calculates the total distance AUVs need to travel for a data collection mission and return to the pearl. The distance is determined using coordinates and is converted into meters. The code then calculates if the AUV has enough battery power for the trip, taking into account an 80% safety margin. If the AUV's energy requirement for the trip is within this margin, it is deemed capable of undertaking the mission; otherwise, it is not.

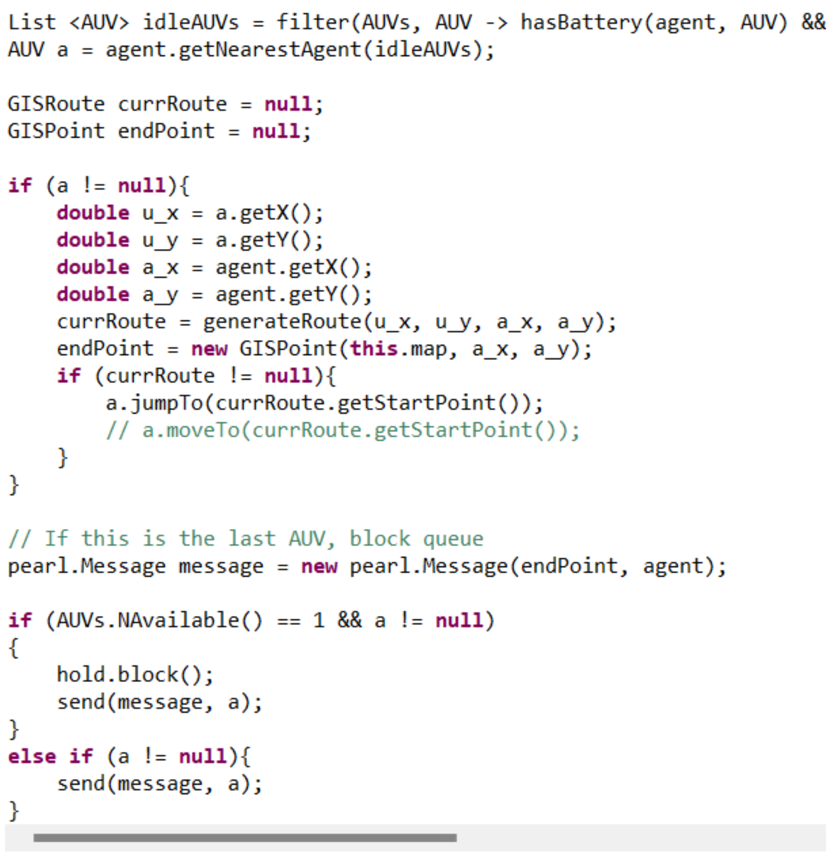


```
// distance from UAV to Science
double x1 = (agent.getX()-AUV.getX())*111003;
double y1 = (agent.getY()-AUV.getY())*87361;

// distance from Science to nearest ship (to charge)
double x2 = (agent.getX()-agent.getNearestAgent(pearls).getX())*111003;
double y2 = (agent.getY()-agent.getNearestAgent(pearls).getY())*87361;

// total distance of trip
double r = sqrt(pow(x1,2)+pow(y1,2)) + sqrt(pow(x2,2)+pow(y2,2));
traceln("AUV calc: " + (r/(AUV.speed*60) * AUV.dischargeRate) + " battery: " + AUV.batteryAvailable);
return (r/(AUV.speed*60) * AUV.dischargeRate <=  0.8*AUV.batteryCapacity); // battery Ava
```

Figure 26: Determination of energy storage available to reach and return from science event.

Note: This part of the code may be buggy, and the calculation should take into account the amount of power consumed during data collection, as mentioned above. If we just consider the round trip and ignore the power consumed during data collection, it will result in a very short collection time.

The code detailed below demonstrates how routes are generated for AUVs to reach science events. The code generates a list of idle AUVs that are either at "idle" or "AtPEARL". It then finds the closest "idle" AUV and generates a route for this AUV to the event's location. The agent then creates a message containing the agent's information and the final destination of the AUV. The message is held in a queue until the last available AUV has been located, at which point it's sent to that AUV to initiate its route. If other AUVs are available, the message is sent immediately without waiting in the queue.



```java
List <AUV> idleAUVs = filter(AUVs, AUV -> hasBattery(agent, AUV) &&
AUV a = agent.getNearestAgent(idleAUVs);

GISRoute currRoute = null;
GISPoint endPoint = null;

if (a != null){
    double u_x = a.getX();
    double u_y = a.getY();
    double a_x = agent.getX();
    double a_y = agent.getY();
    currRoute = generateRoute(u_x, u_y, a_x, a_y);
    endPoint = new GISPoint(this.map, a_x, a_y);
    if (currRoute != null){
        a.jumpTo(currRoute.getStartPoint());
        // a.moveTo(currRoute.getStartPoint());
    }
}

// If this is the last AUV, block queue
pearl.Message message = new pearl.Message(endPoint, agent);

if (AUVs.NAvailable() == 1 && a != null)
{
    hold.block();
    send(message, a);
}
else if (a != null){
    send(message, a);
}
```

Figure 27: Generating routes and sending AUV agents to science events.

Further details on the battery and storage function of each AUV agent are detailed below.

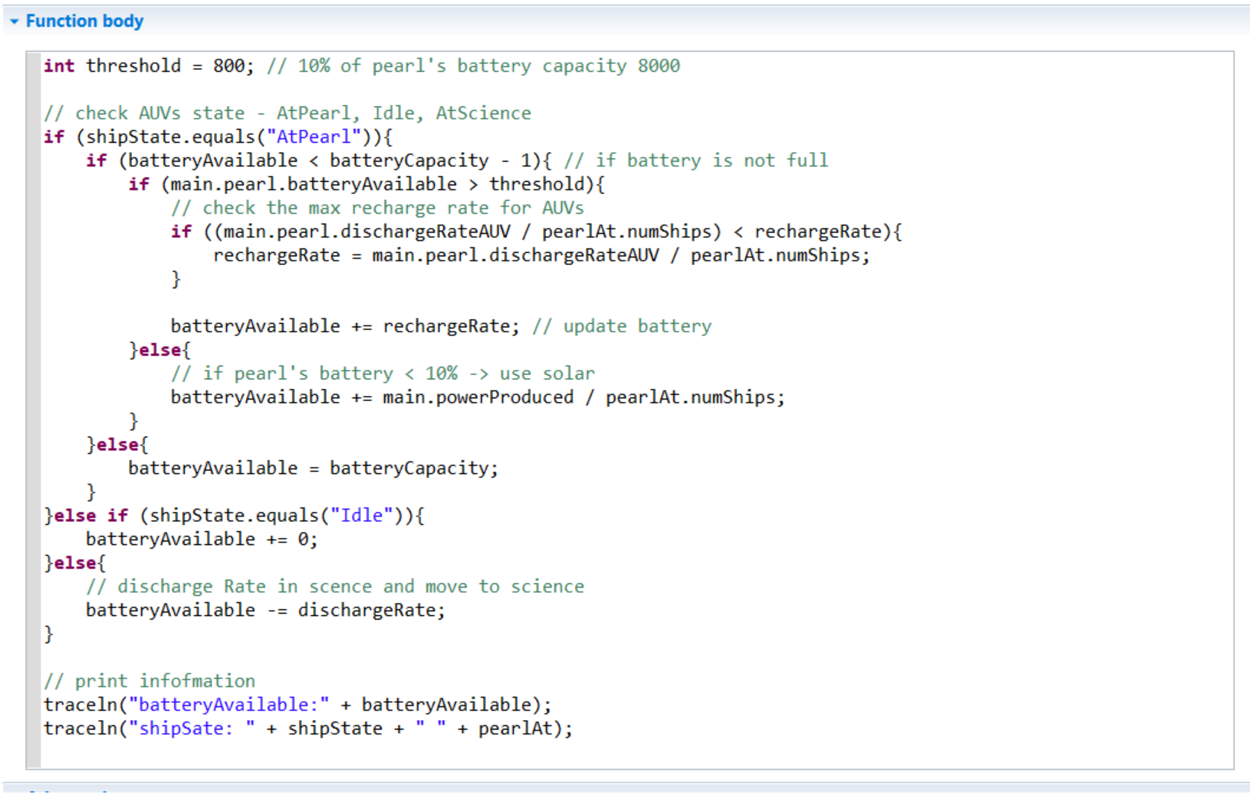
AUV Agent Battery Function

The code below in Fig. 28 represents a system in which Autonomous Underwater Vehicles (AUVs), referred to as 'ships', operate in different states: "**AtPEARL**", "**Idle**", or "**AtScience**".

AtPEARL: When a ship is docked at the "PEARL" station, the code manages the recharging of the ship's battery. If the battery isn't already full, the available battery power at PEARL is checked. If it is above a certain threshold (10% = 800 units), the ship calculates the maximum possible recharge rate considering that the discharge rate from PEARL is shared among all ships present.

If this maximum rate is lower than the ship's typical recharge rate, the recharge rate is adjusted downwards. The ship's battery is then recharged at this rate. If PEARL's available battery power is below the threshold, the ship uses solar power for recharging, with the power produced being shared among all ships present. If the ship's battery is already full, it is capped at its maximum capacity to prevent overcharging.

Idle: When a ship is in the "Idle" state, its battery power remains unchanged. It neither consumes power nor recharges.



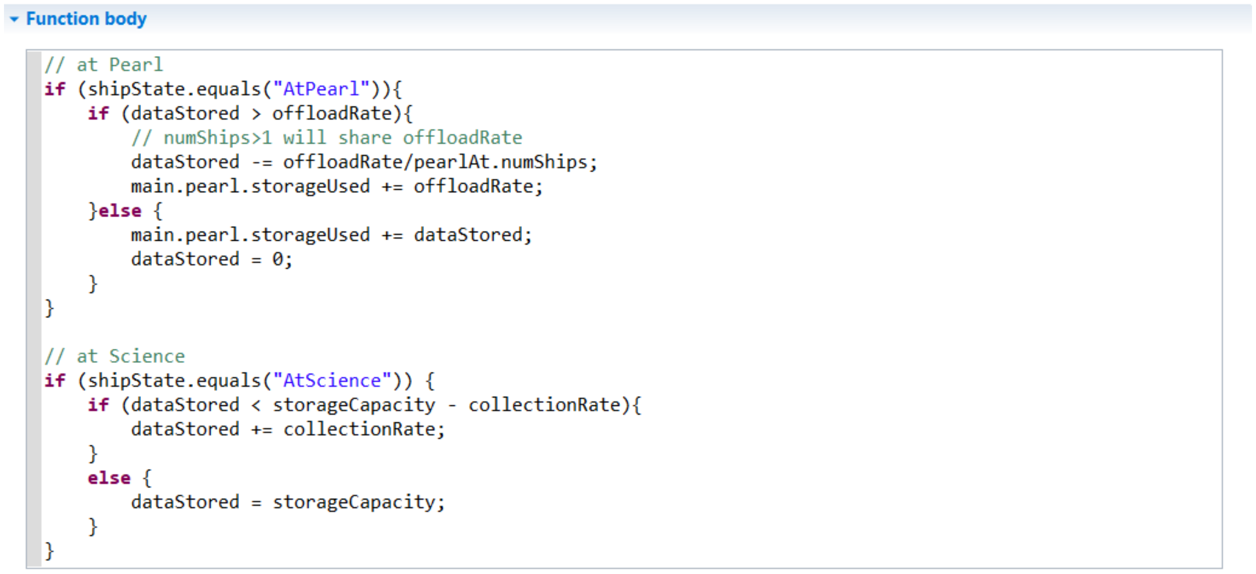
```
int threshold = 800; // 10% of pearl's battery capacity 8000

// check AUVs state - AtPearl, Idle, AtScience
if (shipState.equals("AtPearl")){
    if (batteryAvailable < batteryCapacity - 1){ // if battery is not full
        if (main.pearl.batteryAvailable > threshold){
            // check the max recharge rate for AUVs
            if ((main.pearl.dischargeRateAUV / pearlAt.numShips) < rechargeRate){
                rechargeRate = main.pearl.dischargeRateAUV / pearlAt.numShips;
            }

            batteryAvailable += rechargeRate; // update battery
        }else{
            // if pearl's battery < 10% -> use solar
            batteryAvailable += main.powerProduced / pearlAt.numShips;
        }
    }else{
        batteryAvailable = batteryCapacity;
    }
}else if (shipState.equals("Idle")){
    batteryAvailable += 0;
}else{
    // discharge Rate in scence and move to science
    batteryAvailable -= dischargeRate;
}

// print infofmation
traceln("batteryAvailable:" + batteryAvailable);
traceln("shipSate: " + shipState + " " + pearlAt);
```

Figure 28: Code calculating battery charge and resulting states for AUV agents.

AtScience: In this state or any other state not specified before, the ship discharges power at a set rate, reducing the available battery power. This represents the ship conducting scientific tasks or moving around, activities that consume battery power.

AUV Agent Storage Function

The storage function has two states, a "PEARL" state and a "Science" state. The PEARL state means that the AUVs will offload their data. The Science state means that the AUVs will perform data collection. The PEARL state considers two cases, i.e., stored data is greater than the offload rate and less than the offload rate. "dataStored" indicates the data that the AUVs have collected. This means that they will perform data offloading with different logic. The "Science'' state considers the relationship between collection rate of AUVs and storage capacity of the AUVs. The AUVs will collect data based on the available storage capacity. It's very simple logic.



Function body

```
// at Pearl
if (shipState.equals("AtPearl")){
    if (dataStored > offloadRate){
        // numShips>1 will share offloadRate
        dataStored -= offloadRate/pearlAt.numShips;
        main.pearl.storageUsed += offloadRate;
    }else {
        main.pearl.storageUsed += dataStored;
        dataStored = 0;
    }
}

// at Science
if (shipState.equals("AtScience")) {
    if (dataStored < storageCapacity - collectionRate){
        dataStored += collectionRate;
    }
    else {
        dataStored = storageCapacity;
    }
}
```

Figure 29: Code calculating data collected and stored by AUV agents when visiting science events.

“totalDataCollected” represents all the data that was offloaded to PEARL.

The PEARL Agent

The PEARL model is simpler, and has two states – idle, for when no ships are docking and charging, and ContainsShip, meaning a ship is charging on PEARL.

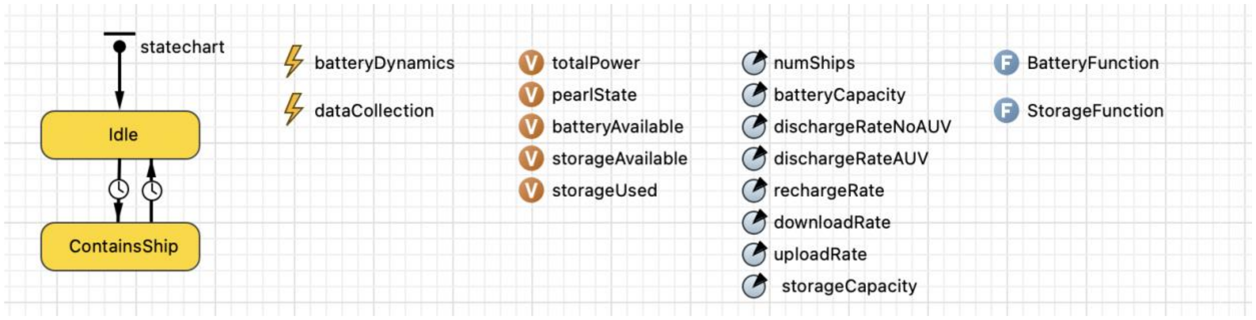


Figure 30: PEARL agent state chart.

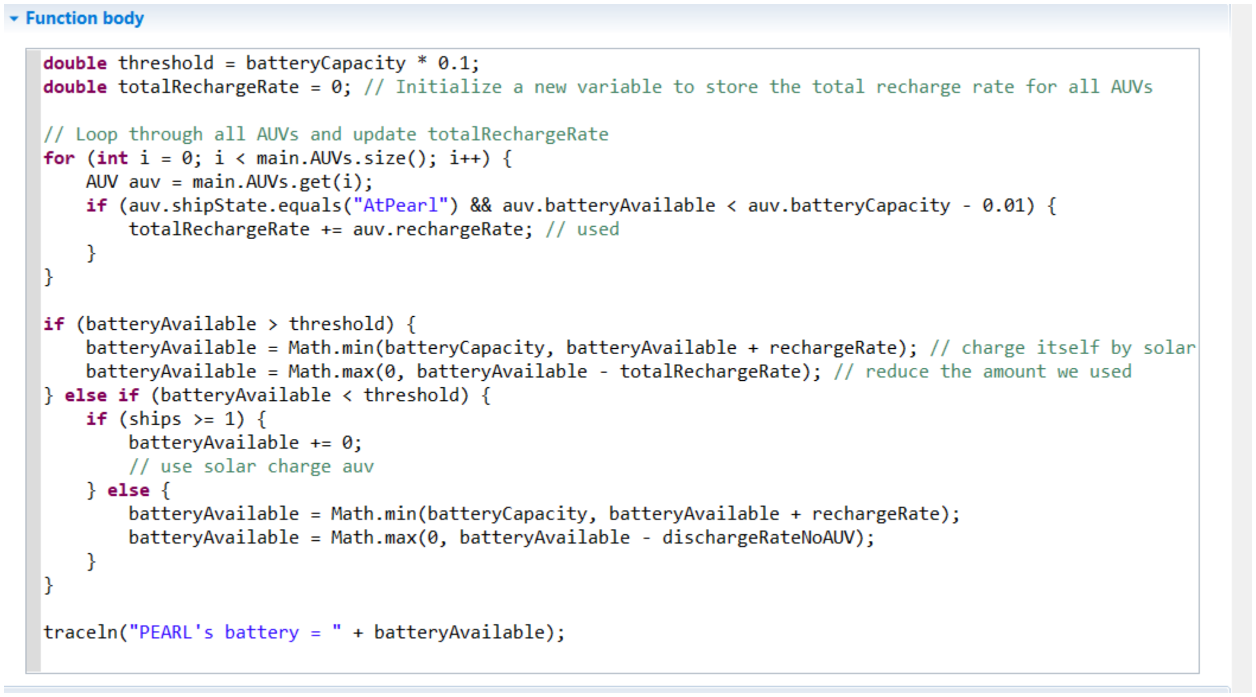
PEARL Agent Battery Function

To determine the amount of battery stored in PEARL, the system first counts the number of AUVs at PEARL whose batteries are not fully charged and calculates the total recharge demand. If the total recharge demand exceeds 80% of the station's maximum discharge rate for AUVs, it is capped at that maximum rate.

PEARL then manages its battery power depending on whether it has above or below the threshold charge:

- **Above Threshold**: If PEARL's available battery is above the threshold, it first recharges its battery using solar power, ensuring not to exceed its battery capacity. Then, if AUVs exist, it uses the calculated recharge rate to recharge the AUVs, reducing its available battery.

- **Below Threshold**: If PEARL's battery power is below the threshold, its behavior depends on the presence of AUVs. If there's at least one AUV, it doesn't use any of its battery power, leaving the solar power to charge the AUVs. If there are no AUVs, it recharges its battery using solar power and then discharges at a rate defined by discharge Rate without AUVs. This prevents draining PEARL's battery further while still providing power to the AUVs.



```
double threshold = batteryCapacity * 0.1;
double totalRechargeRate = 0; // Initialize a new variable to store the total recharge rate for all AUVs

// Loop through all AUVs and update totalRechargeRate
for (int i = 0; i < main.AUVs.size(); i++) {
    AUV auv = main.AUVs.get(i);
    if (auv.shipState.equals("AtPearl") && auv.batteryAvailable < auv.batteryCapacity - 0.01) {
        totalRechargeRate += auv.rechargeRate; // used
    }
}

if (batteryAvailable > threshold) {
    batteryAvailable = Math.min(batteryCapacity, batteryAvailable + rechargeRate); // charge itself by solar
    batteryAvailable = Math.max(0, batteryAvailable - totalRechargeRate); // reduce the amount we used
} else if (batteryAvailable < threshold) {
    if (ships >= 1) {
        batteryAvailable += 0;
        // use solar charge auv
    } else {
        batteryAvailable = Math.min(batteryCapacity, batteryAvailable + rechargeRate);
        batteryAvailable = Math.max(0, batteryAvailable - dischargeRateNoAUV);
    }
}

traceln("PEARL's battery = " + batteryAvailable);
```

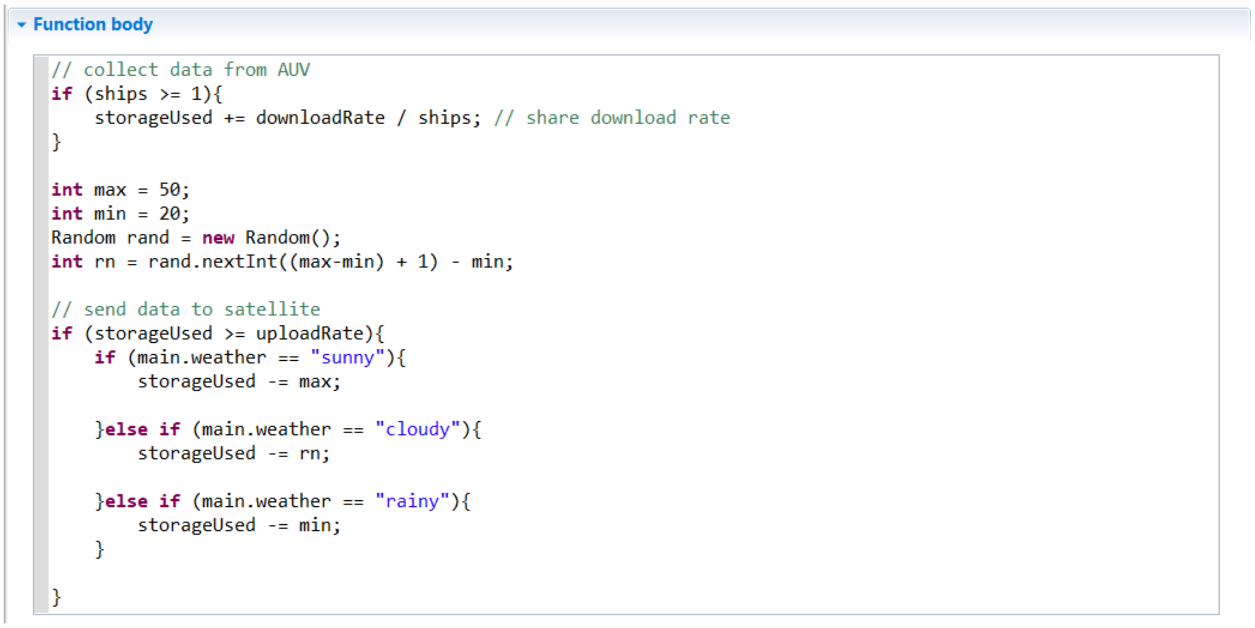
Figure 31: PEARL battery storage function

PEARL Agent Storage Function

First, it checks if there are AUVs present. If so, the AUVs collect data, and the storageUsed variable is increased accordingly, divided among the AUVs.

The system then checks if the accumulated data (storageUsed) has reached the upload capacity (uploadRate). If so, it uploads data to a satellite based on the current weather conditions.

If it's sunny, the system can transmit the maximum amount of data, which is subtracted from storageUsed. If it's cloudy, the system can only send a randomly determined amount of data within the previously generated range, also subtracted from storageUsed. If it's rainy, the system can only send the minimum amount of data, again subtracted from storageUsed.

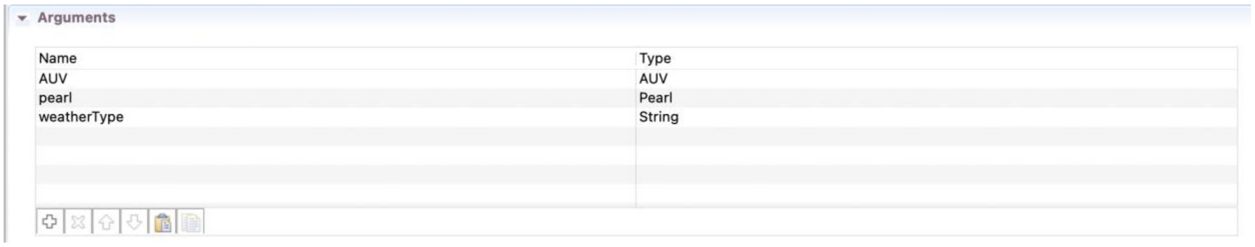


```
// collect data from AUV
if (ships >= 1){
    storageUsed += downloadRate / ships; // share download rate
}

int max = 50;
int min = 20;
Random rand = new Random();
int rn = rand.nextInt((max-min) + 1) - min;

// send data to satellite
if (storageUsed >= uploadRate){
    if (main.weather == "sunny"){
        storageUsed -= max;

    }else if (main.weather == "cloudy"){
        storageUsed -= rn;

    }else if (main.weather == "rainy"){
        storageUsed -= min;
    }

}
```

Figure 32: PEARL data storage function

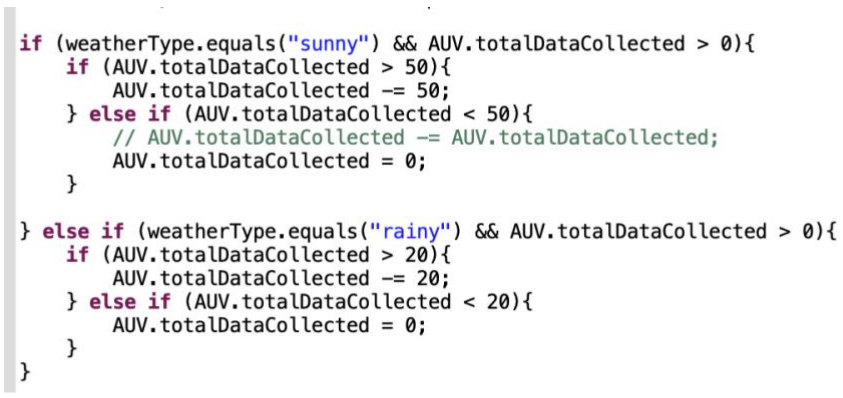
PEARL Agent Data Transmission

Data transmission is the transfer of data from PEARL to the satellite. The rate of satellite transmission is completely determined by the weather. On a perfectly clear day, the data transfer rate is considered to be a maximum of 50 MBps, while on a rainy day it is a minimum of 20 MBps and is linearly related to the cloudiness.



Name	Type
AUV	AUV
pearl	Pearl
weatherType	String

Figure 33: PEARL model variables



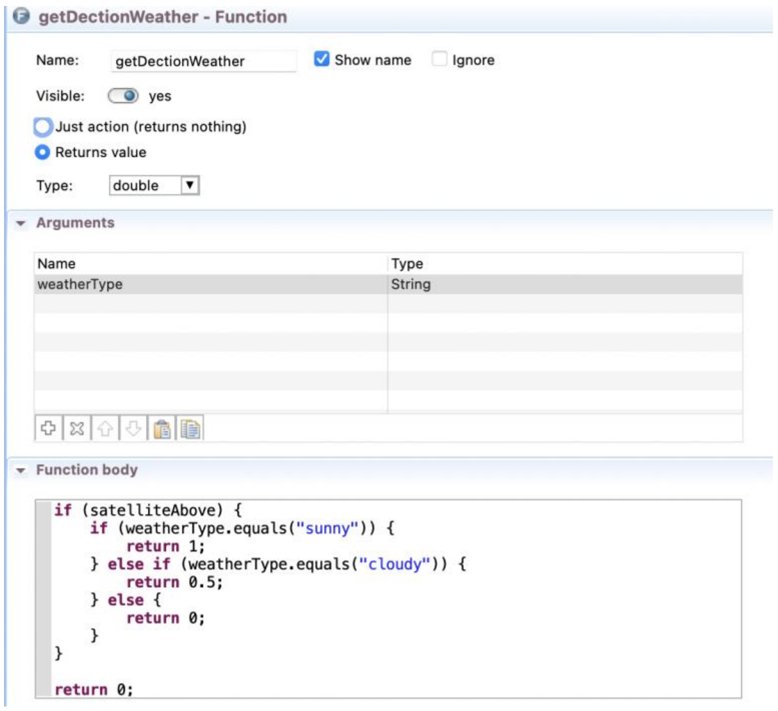
```
if (weatherType.equals("sunny") && AUV.totalDataCollected > 0){
    if (AUV.totalDataCollected > 50){
        AUV.totalDataCollected -= 50;
    } else if (AUV.totalDataCollected < 50){
        // AUV.totalDataCollected -= AUV.totalDataCollected;
        AUV.totalDataCollected = 0;
    }

} else if (weatherType.equals("rainy") && AUV.totalDataCollected > 0){
    if (AUV.totalDataCollected > 20){
        AUV.totalDataCollected -= 20;
    } else if (AUV.totalDataCollected < 20){
        AUV.totalDataCollected = 0;
    }
}
```

Figure 34: PEARL model weather function

There is a variable in the AUV that records all the data collected. The speed of upload depends on the weather. We have two conditions here: rain and sunny days. All the data stored in the pearl will be transferred to the satellite. In other words, the storage capacity of PEARL will increase again. This part of the code is still buggy and needs further verification if the logic of the code is correct.

PEARL Model Detection of Science Event

Detection of science is presumed to be done by a satellite passing overhead. "satelliteAbove" is used as a variable of type boolean as a condition. Since the science detection is based on the cloud cover in the area, different weather will affect the detection results.



getDectionWeather - Function

Name: getDectionWeather [✓ Show name] [ ] Ignore

Visible: yes

Just action (returns nothing)
[✓ Returns value]

Type: double

Arguments

Name	Type
weatherType	String

Function body

```
if (satelliteAbove) {
    if (weatherType.equals("sunny")) {
        return 1;
    } else if (weatherType.equals("cloudy")) {
        return 0.5;
    } else {
        return 0;
    }
}

return 0;
```

Figure 35: PEARL data transmission function.

PEARL Agent Path Planning

The routes for agents in the PEARL model are generated by AnyLogic's GIS Map functionality, which is a different implementation method compared with the larger SOS model described earlier. In particular, the path planning is comprised of the following steps:

- **Initialization of GIS Points**: The GISPoint instances startPoint and endPoint are initialized using the given fromX, fromY, toX, toY coordinates.
- **Checking the Destination**: The inIsland(toX, toY) function is presumably checking whether the destination lies inside an island. If it does, the function returns null, as a route can't be created.
- **Calculating Direction**: v_x and v_y represent the direction vector from the current position to the destination. h is the magnitude (length) of this vector.
- **Considering PEARL Position**: The direction vector is normalized and scaled by a factor of 0.1, resulting in a step vector (v_x, v_y) of length 0.1 in the direction of the destination.
- **Checking for Route Intersection**: The routeIntersect(origFromX, origFromY, targetX, targetY) function checks if the straight path from the current position to the new position intersects with any obstacles. If it does, or if the straight path from the new position to the final destination intersects with any obstacles, the path needs to be corrected.
- **Correcting Path**: This step involves adjusting the current position (fromX, fromY) in either the x or y direction depending on which absolute direction value (v_x or v_y) is greater. This seems to be a strategy to avoid the obstacle by taking a step to the side with the smallest absolute direction component, thus trying to get around the obstacle.
- **Creating Segment**: Once a valid step is found that doesn't lead to an intersection with an obstacle, a GISMarkupSegmentLine object is created, which represents a straight line segment from the old position to the new position. This segment is added to segmentsToKeep.

Once the destination is reached, all the kept segments are used to create a GISRoute object, routeUpToIsland. This is a path that represents a valid route from the starting point to the destination. A GISNetwork object, n, is then created. This represents a network of routes on the GIS map. The network is created from the startPoint, through routeUpToIsland, to the endPoint. The function returns routeUpToIsland which is a valid route from the starting point to the destination.

PEARL Model Future Work

The current PEARL model has all the basic functions detailed previously. Future work will test and check for logical errors and run different cases to optimize an overall system to maximize science data collected. Some initial bugs from testing are detailed below:

a) The data collection duration appears to be quite brief according to the current implementation, as mentioned in the previous discussion about the AUVs section. To thoroughly test and verify the round-trip logic and data collection functionality, I try to assign tasks at varying distances around the PEARL station (close to the pearl). This method is used to test whether the round-trip logic and collection is working.

b) The rate at which AUVs offload data doesn't follow a linear pattern. We need to test the storage logic of pearls and AUVs.

Some initial optimization was conducted with the PEARL system in the larger SoS model and is detailed below.

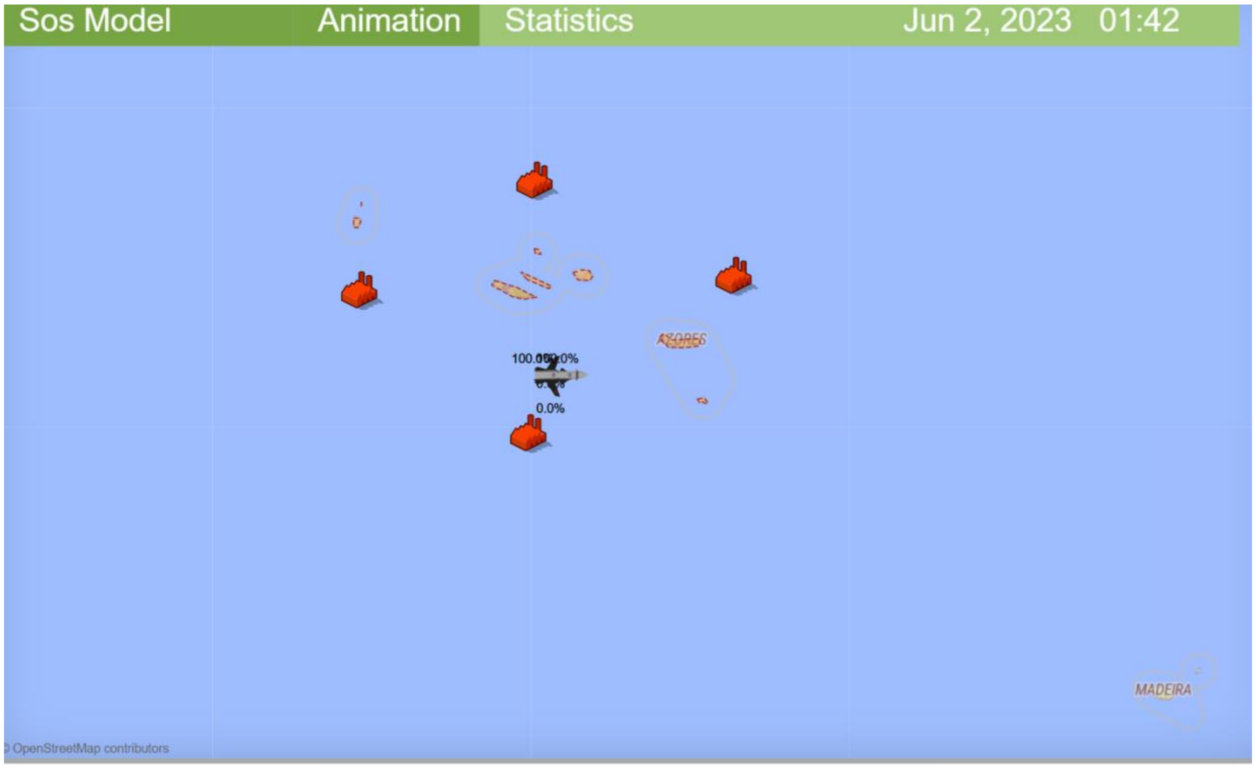
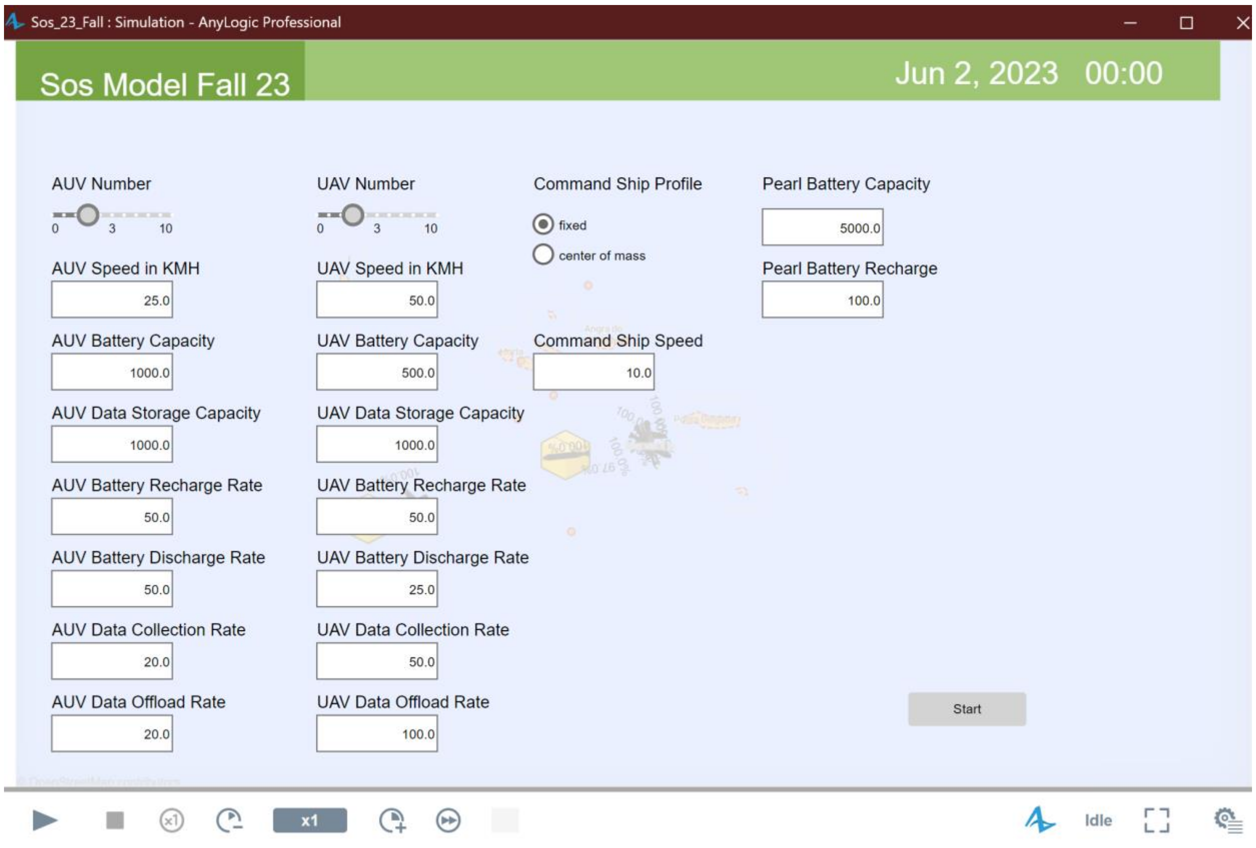


Figure 36: Initialization of SoS model with PEARL implemented showing initial location of all agents.



Sos_23_Fall : Simulation - AnyLogic Professional

Sos Model Fall 23 — Jun 2, 2023 00:00

AUV Number (slider 0 – 3 – 10)	UAV Number (slider 0 – 3 – 10)	Command Ship Profile: [✓ fixed] center of mass	Pearl Battery Capacity: 5000.0
AUV Speed in KMH: 25.0	UAV Speed in KMH: 50.0		Pearl Battery Recharge: 100.0
AUV Battery Capacity: 1000.0	UAV Battery Capacity: 500.0	Command Ship Speed: 10.0	
AUV Data Storage Capacity: 1000.0	UAV Data Storage Capacity: 1000.0		
AUV Battery Recharge Rate: 50.0	UAV Battery Recharge Rate: 50.0		
AUV Battery Discharge Rate: 50.0	UAV Battery Discharge Rate: 25.0		
AUV Data Collection Rate: 20.0	UAV Data Collection Rate: 50.0		
AUV Data Offload Rate: 20.0	UAV Data Offload Rate: 100.0		

Figure 37: Results of one optimization run.

With PEARL, the optimal number of UAVs is 3 and AUVs is 1. Without PEARL, the optimal number of UAVs is 2 and AUVs is 2. Hence, the inclusion of PEARL allows the system to utilize AUVs better but still requires the use of an additional UAV.

SUMMARY

A working discrete event simulation now exists for modeling the science capture around the islands of the Azores. The simulation is a system of systems simulation which models a satellite overpass (through when science is observed given weather), a command ship with a variable number of UAVs and AUVs and a variable number of Pearl (remote charging and docking stations for the AUVs).

The simulation can return the optimal number of AUVs and UAVs depending on the location of the command ship and how many Pearls are deployed.

LOCATION OF THE CODE

The code can be downloaded at

https://github.com/SoS-Ocean-Observation/SoS-Models-FA22/releases/tag/SOS_release

APPENDIX

System of Systems Concept for Effective Oceans to Near Space Observation

Prof. Daniel Hastings, MIT AeroAstro

2020-2021 Seed Project Report

MIT Portugal Partnership 2030

MIT Portugal

1. PROJECT TEAM

- ***MIT:***

o *Principal investigator: Prof. Daniel Hastings (Aero Astro)*

o *Postdocs: Dr. Maha N. Haji (Aero Astro)*

o *Graduate Students: Alice Cooper (Aero Astro)*

o *Undergraduate Students: Joseph Merkel (Aero Astro), Parker Mayhew (Aero Astro)*

- ***Collaborators in Portugal:***

o Industry Partners: N/A

o Research Partners: Joao Tasso de Figueiredo Borges de Sousa, Professor at Faculdade de Engenharia - Porto University; *Director of LSTS*

- ***Other collaborators:***

o N/A

2. SUMMARY- *250+ words; brief project description*

Effective and efficient collection of synoptic data across a range of spatial and temporal scales from oceans to near space will drive the design of a complex system of systems (SoS). The SoS that has to be created will brings together sensing from four levels of platforms from (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles). In addition to long term observations, the SoS must be able to respond to short time scale phenomena with initially unknown spatial distribution. The SoS may involve a large number of assets which will need to be dynamically allocated.

A SoS discrete event simulation was developed that models the physics of the different sensing platforms as well as the sensing capabilities on each platform. A planning and execution control framework will be used to explore a set of different objective functions. The most effective use of the SoS is expected to vary considerably with the choice of objective function.

3. OUTCOMES & ACHIEVEMENTS

Motivation and Overview

The vision for this project is to lay the foundations for an Atlantic ocean scientific observation platform, encompassing the three segments: space, air, and ocean (both surface, water column, and sea-bottom). The combination of all these segments allows the synoptic observation of scientific phenomena over a broad range of spatial and temporal resolutions: from large areas from space, to in-situ sensing at the ocean, from large timescales from archival data processing, to the time-efficient deployment of scientific assets on the air and ocean, triggered by real-time remote sensing from space, as well as by in-situ observations. For example, the problem of measuring air-sea fluxes is now receiving a lot of attention

In order to accomplish this vision, a system of systems has to be created which sensing from four levels of platforms from (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles). This SoS needs to collect coordinated data over a wide range of spatial and temporal scales. The system concept, as well as a planning and execution control framework or concepts of operation, for ocean to near space observation will be based on networked underwater, surface, air and space vehicles.

The number of assets will be heterogeneous and may be present in large numbers (10s, 100s) endowed with self-organization capabilities and capable of long endurance missions for sustained observation.

The behavior of the SoS is event-driven in nature: once a scientific event is detected an agent needs to be mobilized to collect data at a specific location; once that agent has run of our resources (such as battery or memory) it needs to return to a command ship in order to replenish and a new agent must be sent to collect data; if the scientific event disappears, all agents must return to the command ship. Because of this, the SoS lends itself to being modeled using a discrete event simulation (DES). A DES models the operation of a system as a discrete sequence of events in time. It can also model the interactions between objects, and system operations within the system where these interactions are time-dependent.

This report details the DES of the proposed SoS for Oceans to Space (satellites, UAVs or gliders, surface ships, ASVs and AUVs) developed so far that can be used to plan the architecture of the system of systems. The planning and execution framework developed using this model will be used to explore and outline the different concepts of operation.

Model Overview

The discrete event simulation model is developed using a commercially available software called AnyLogic.[2] AnyLogic is a multimethod simulation modeling tool that supports agent-based, discrete event, and system dynamics simulation methodologies. The logic of the model follows flowchart-like behavior, which makes the use of AnyLogic particularly helpful given the proclivity the software has for flowchart-like behavior, including process flow modeling and statecharts for agents. For the System of Systems model, the decision-making behavior that agents use and the probability-based spawning and detection logic can be generalized with the following flowchart shown in Figure 1.

The AnyLogic model is centered around a moveable map (shown in Figure 2) with sliders to adjust the number of AUV and UAV agents. The map is centered around the Azores in the Atlantic ocean, the main area of interest for our Portugese collaborators. Currently there are no other adjustable parameters at run time, but this will likely change as the model improves. Agents are shown in model time as they move to and from sciences, while plots and statistics are collected and displayed to the right of the model.

Science

In order to take the scientific measurements, an effective combination of (1) space systems (satellites), (2) airborne systems (uncrewed aerial vehicles), (3) surface systems (crewed ships and autonomous surface vehicles), and (4) underwater systems (autonomous underwater vehicles) must be built. Since the assets are mobile, the combination will be inherently dynamic. The model under development detailed in this report will ultimately be used to find the best dynamic combinations depending on the choice of objective function.

Modeling the progression of science throughout the SoS with a process flowchart stood out as the most effective model approach. Representing science as a resource that passes through various processing states is much more representative of science management in the real world than modeling science as an independent agent

[2] https://www.anylogic.com

or dynamic variable. Additionally, AnyLogic has a very broad and capable library for process modeling which allows us to simulate the science lifecycle more accurately than with other methods.

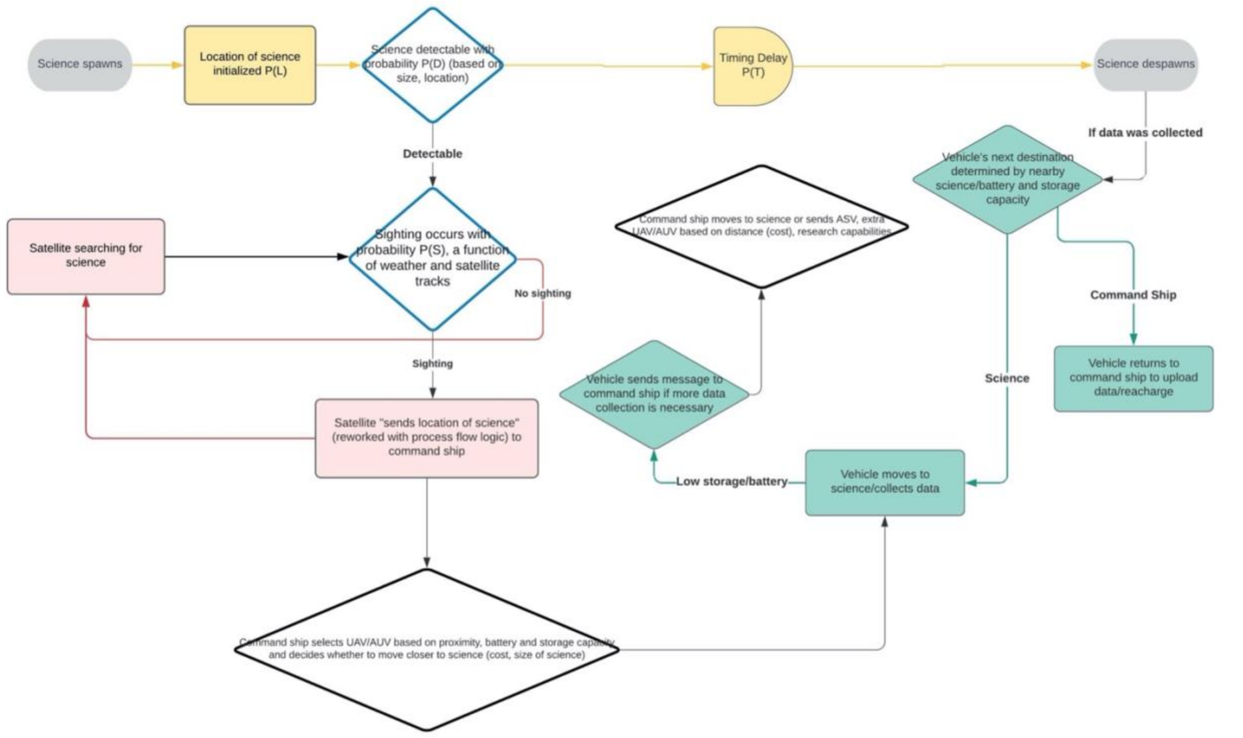

Science spawns
Location of science initialized P(L)
Science detectable with probability P(D) (based on size, location)
Timing Delay P(T)
Science despawns
Detectable
If data was collected
Vehicle's next destination determined by nearby science/battery and storage capacity
Command ship moves to science or sends ASV, extra UAV/AUV based on distance (cost), research capabilities
Satellite searching for science
Sighting occurs with probability P(S), a function of weather and satellite tracks
No sighting
Command Ship
Sighting
Vehicle sends message to command ship if more data collection is necessary
Science
Vehicle returns to command ship to upload data/reacharge
Satellite "sends location of science" (reworked with process flow logic) to command ship
Low storage/battery
Vehicle moves to science/collects data
Command ship selects UAV/AUV based on proximity, battery and storage capacity and decides whether to move closer to science (cost, size of science)

Figure 1: Flow chart detailing the logic of the model.

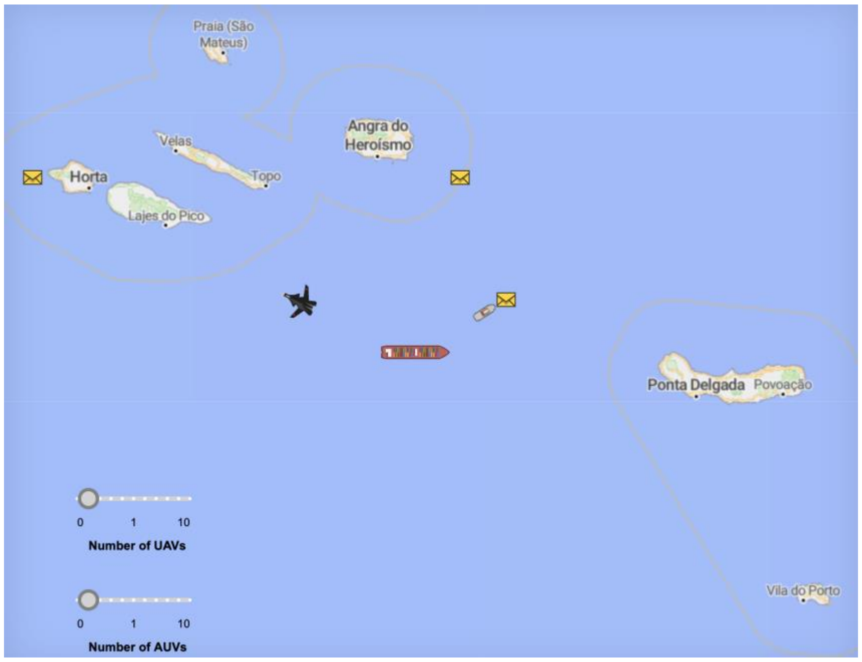


Figure 2: Image of map used in model.

The logic of science is modeled through several flowchart states. Specific types of science such as fault line science will enter the model in stage one, after which they will be passed to command ship operations for stage two, until finally they are passed onto stage three for agent handling and model exit. Each of these stages represents a phase of science processing in the real world.

Within each flowchart stage, the science will undergo various steps modeled by AnyLogic processing blocks. These blocks represent how science is interacted with, such as queueing science on the command ship for agent pickup. Science is displayed on the GIS map as a yellow envelope.

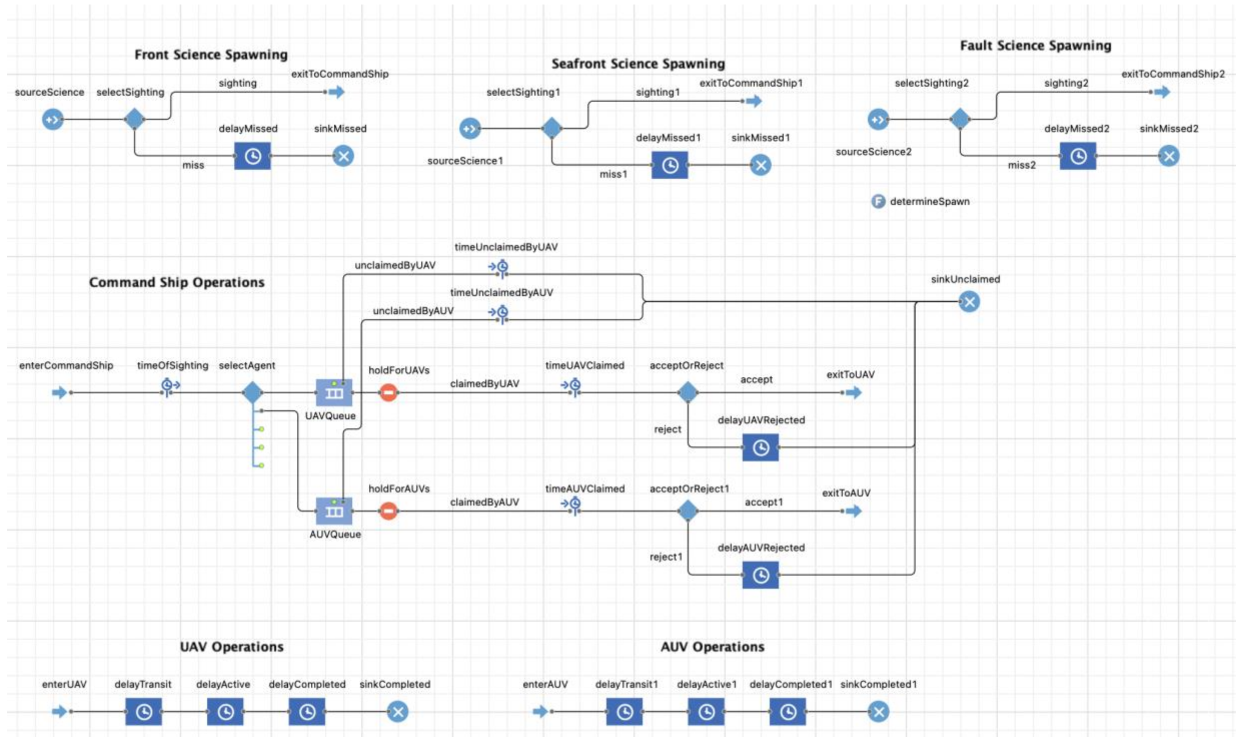


Figure 3: Image of flowchart showing how science is spawned and moves through the various agents in the model.

Science agents are subdivided into three categories based on location of origin, which can be utilized in the system to model different movement patterns of the sciences, which can affect the cost of the system to collect data from it, as well as potential data storage capacity, or the "size" of the science in terms of data storage.

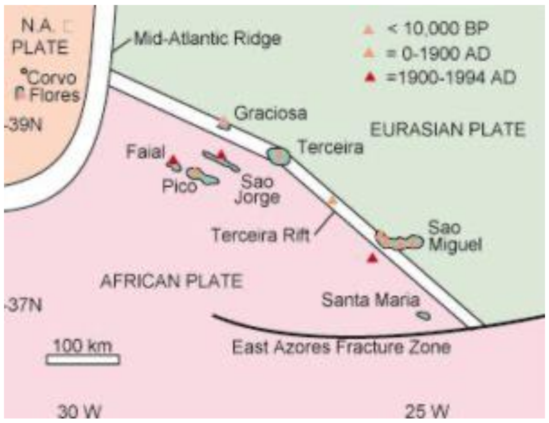


Figure 4: Rough visual showing key geological features near the Azores for approximation in model. (https://www.azoreschoice.com/blog/a-brief-history-of-the-azores-part-two/)

Seamount science: Between the islands Terceira and Sao Miguel and the Eurasian and African tectonic plates is the Joao de Castro Seamount, an underwater volcano that makes up a significant portion of underwater geological activity[3], although specific rates in proportion to the rest of the region's tectonic activity has yet to be concluded and implemented. The seamount is currently where most of the science spawns in the model. These underwater science events are collected by AUVs

Fault line science: The Islands Azores operate close to and functionally on top of a triple junction between the North American, African, and Eurasian tectonic plates. Similar to the seafront, this is an area of significant underwater tectonic activity.[2] These sciences are collected by AUVs

Front science: Oceanic research between agents with differing detection capabilities allows for the collection of data between all depths. The surface of the ocean also has valuable data potential, which in reality can potentially be tied to tectonic activity, but is mainly dependent on climate patterns.[2] These sciences operate just below the surface of the ocean and above, and have the potential to move around the map. These sciences are collected by UAVs

When science enters the model, a step referred to as "spawning", it is assigned a specific location within the GIS space. In order to place science randomly or according to a given distribution, it is helpful to create AnyLogic GIS regions. These regions are used to focus the possible spawning locations for sciences within a desired area. Additionally, regions play a major role in path planning (discussed below). Regions are currently implemented using the AnyLogic space markup palette, and are drawn by hand on the GIS map to meet our needs.

[3] Storch, B., Haase, K.M., Romer, R.H.W. *et al.* Rifting of the oceanic Azores Plateau with episodic volcanic activity. *Sci Rep* 10, 19718 (2020). https://doi.org/10.1038/s41598-020-76691-1

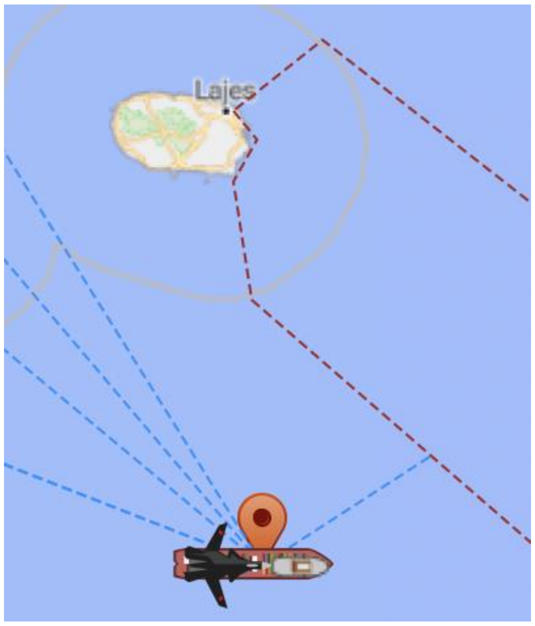


Figure 5: Image showing map used in AnyLogic with GIS regions outlined in red dashed lines. The dashed blue lines indicate paths that AUVs must follow to and from the GIS regions.

Seamount spawns in a two-dimensional gaussian distribution around the geographical center of the seamount with a rate of 1 instance per hour, which is subject and readily available to change given any change in information regarding activity rate. Our team chose 1 instance per hour because it gives a good representation of the shape of the seamount in terms of the underwater scientific activity it creates. Similar to this, the fault science spawns in a gaussian distribution around the geographical fault lines modeled by linear equations, with a spawn rate of 0.2 per hour. Currently, front sciences are stationary and spawn randomly around regions that are not on land. These sciences also have a spawn rate of 0.2 per hour. Currently, 0.2 spawns per hour is a placeholder number until more accurate data regarding oceanic activity can be incorporated into the model.

Model Agents

The key agents in the SoS model are the ship, the AUVs, and the UAVs. Data was provided by our partners in Portugal of nine different AUVs and three different UAVs they often use in the field for scientific research. The specifications of each vehicle include attributes such as battery capacity, maximum ascent and descent rates, and data storage capacity. Based on these various vehicle types, an "average" AUV and UAV was used for the initial prototype model developed in this effort.

Ship: The central piece in the model is a command ship that houses on board the AUVs and UAVs. When an agent (AUV or UAV) is selected to visit a specific science in the queue, the agent is released from the ship for the mission. When the agent is low on battery or storage space for the data collected, it will return to the ship to recharge its batteries and offload its data. In this model, the ship is a stationary feature, however in future prototypes it could move, enabling the entire SoS to move closer to a specific scientific event.

UAVs and AUVs: Both UAVs and AUVs are modeled as AnyLogic agents, which are defined by various states contained within a statechart. For our purposes, the UAVs and AUVs vary between the states "At Command Ship", "Moving To Science", "At Science", and "Moving to Command Ship" (as shown in Figure 6). However, additional states may be added in the future such as "Charging". The UAV and AUV agents transition between these states by various cues. For example, an AUV will transition from “At Command Ship” to “Moving to Science” upon receiving a message that a new science has been sighted and assigned to it. Transitions can also trigger based on other criteria such as arrival at a point, timeout, or when a condition is met.

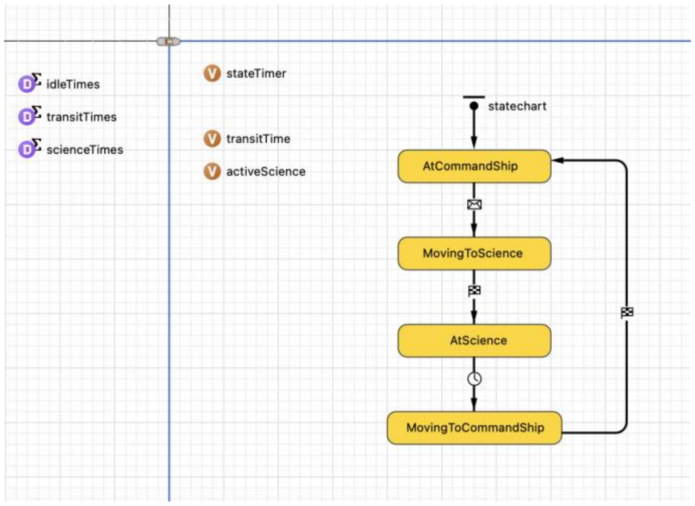


Figure 6: State chart for the AUV (which is identical to the UAV with the exception of the icon in the upper right) showing the various states the agent can move through.

As shown in Figure 7, UAV agents are indicated on the GIS map by a black plane, whereas AUV agents are indicated on the GIS map by a gray tugboat icon.

Figure 7: UAV agent (left) and AUV agent (right)

Currently, the AnyLogic model only has one type of UAV and AUV agent with constant performance parameters. These agents are assigned to any science that spawns below a certain depth threshold. In the future, however, additional UAV and AUV types may be introduced with different performance characteristics, capabilities, and responsibilities.

Each UAV and AUV agent has several variables used to track its assigned science, the time it has spent in its current state, and a dataset of times spent in each state throughout the model run. This information is used to gather data on UAV and AUV performance, which is then displayed through the data visualization tools to inform future decisions. (Datasets are shown by purple icons, variables are gold icons, and statechart is yellow chart with transitions as arrows)

Note that UAV and AUV have a lot of overlap. They act differently (speed, which sciences they claim, etc.) but the logic for states and datasets is very similar. Unlike UAVs, which are able to travel in straight lines, AUVs must avoid obstacles such as land and follow certain routes. This adds the requirement of path planning for AUV agents within the model, as shown previously in Figure 5. Obstacle avoidance and custom routing using

AnyLogic is difficult, so the model currently relies on simple static routes that the AUV can follow between any two neighboring regions. Together, these routes form a GIS network that maps a path from any one region to another region of interest. When traveling to a specific point, the agents will follow the path until at the desired region, and then follow a straight line from the region entry point to the specific point of interest.

Note that in the future, more realistic UAV flight modeling may require similar path planning. Future work may change this approach as it is possible to add a shapefile with waterways that provides a routing option in the properties of the population with agents.

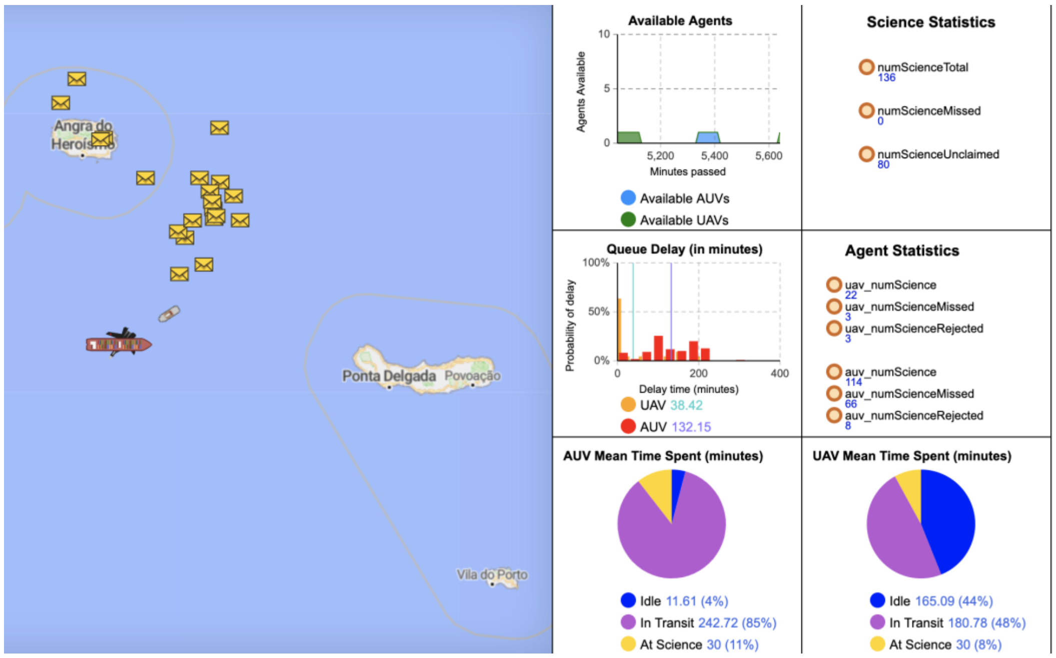
Resource use

Battery capacity and battery discharge has been incorporated into the UAV/AUV agents' statechart as a parameter that the model makes before choosing the appropriate agent to send to spawning sciences. Similar to the distance-duration calculation that the model makes in order to choose whether to send an agent to any given science, the battery capacity parameter allows the model to choose which specific agent collects data from science. The model does this by tracking the battery percentage it would lose making the trip to and from the science and sending an agent if the battery drainage from the total route leaves the agent's battery capacity above 20% full. The model sends the first agent that meets this criteria, assuming it will not "despawn" by the time the agent reaches the science. Using state chart logic, the "battery life" parameter drains whenever the agent is not in the "At Command Ship" state, and recharges when it is in this state.

Future work with resource use can easily be implemented with the use of the logic implemented with battery capacity. The team plans on incorporating a data storage parameter that adds another constraint to optimize for in the model. Similar to battery life, the data storage parameter will increase in the state "At Science" and decrease in the state "At Command Ship", modeling the collection and offloading of scientific data.

Data visualization

To evaluate the result of the model, a number of data visualizations have been implemented in this prototype model. They are described in detail below and show in Figure 8.



numScienceTotal 136
numScienceMissed 0
numScienceUnclaimed 80

uav_numScience 22
uav_numScienceMissed 3
uav_numScienceRejected 3

auv_numScience 114
auv_numScienceMissed 66
auv_numScienceRejected 8

Figure 8: Screenshot of model showing data visualization

Time series of available agents: This plot currently shows a stack chart representing the number of agents available at any given time. While the default plot displays a maximum of ten agents, this parameter can be changed prior to running according to the expected maximum available agents.

Histogram of time to science: This histogram displays the time spent by sciences in command ship queues, waiting to be assigned to an agent. Specifically, it groups queue times into bins and shows the percentage of sciences that stayed in the queue for each time range. In addition to the histogram, the mean queue time for each agent is shown by a vertical line. This plot provides insights on the delay time from science identification to agent assignment, which is correlated with the availability of agents.

Pie chart of time agents spend in various states: Each agent can be categorized into three general states - idle, in transit, or at science. This pie chart shows the percentage of time agents spend in each of the three states. This is particularly useful as a reference plot in combination with other plots to evaluate whether the number, location, or type of agents should be varied to maximize time at science. Note that in the future, we may add "charging", "memory upload", etc. as other agent states.

Key statistics: In addition to plots, some key variables are displayed for reference. For example, the number of total sciences compared with the sciences that go unclaimed by a certain agent type may inform decisions about where we have an understaffing or surplus of agents.

The data visualization plots in the AnyLogic model can be used to gain insights and drive key decisions in the scientific mission. For instance, if the pie chart of time agents spend in various states shows that the most time is spent in transit to science, this might suggest that a large amount of transit time could be reduced by moving the command ship (from where the agents are deployed) closer to the location of most science spawning.

How to run the model

To run the model, open the file titled "SoS Model.alp" using AnyLogic. Note that you must be running AnyLogic 8.7.2 or newer. This will bring you to the AnyLogic Workspace, a graphical editor that will allow you to edit the model.

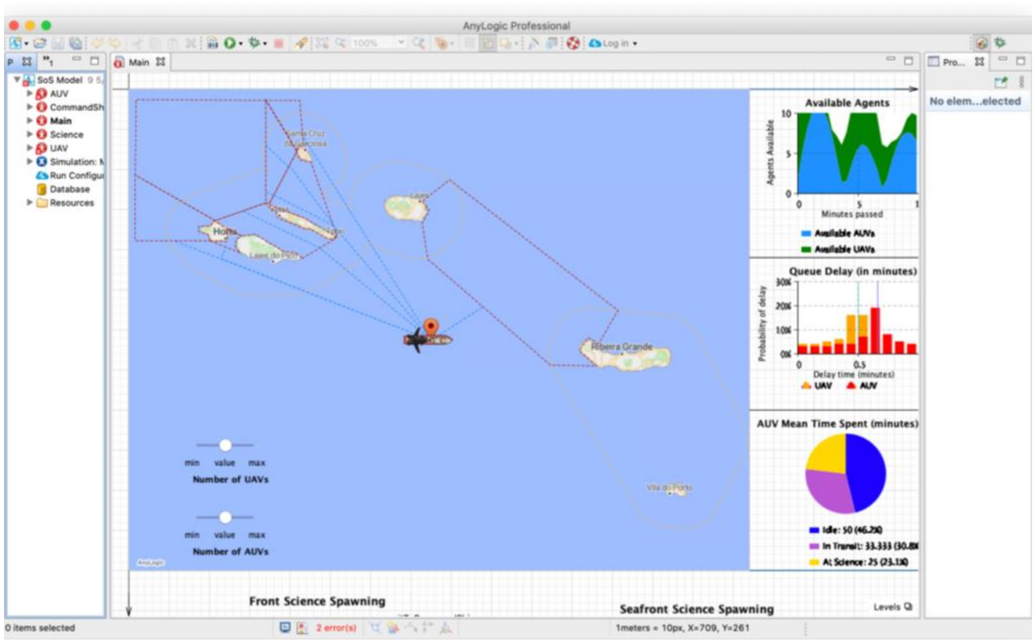


Figure 9: SoS Model AnyLogic Workspace.

To run the model and view the simulation, select "Model" and then "Run," and select "SoS Model/Simulation." A new window will pop up that shows you the GIS map for the model, a command ship, an AUV, and a UAV at the center, two slides on the bottom left to indicate the number of AUVs and UAVs to be used in the system, and the six data visualization plots on the right (you will need to pan right in order to see the last three of these).

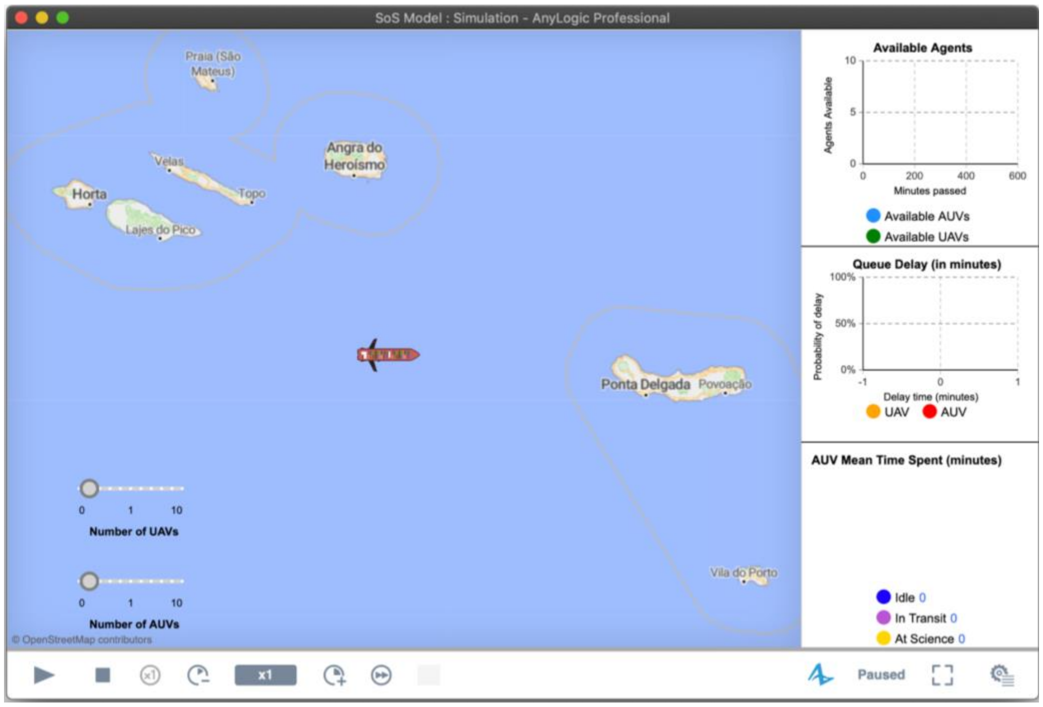


Figure 10: SoS Model Simulation Window.

Select the number of AUVs and UAVs you would like the simulation to use and press the play button. The model runs in real time (minutes) and therefore you will need to speed it up by pressing the button. Usually a speed of at least 25x should suffice to allow you to see the appearance of science and the movement of the agents.

As the model plays out, the plots and statistics on the right will update. You may need to pan right in order to view all six of them.

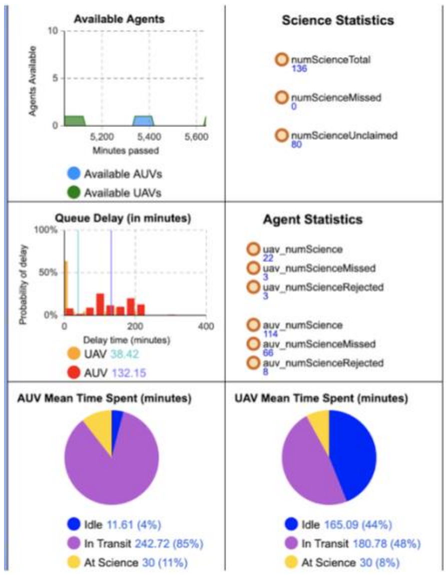


Figure 11: Example plots output during a model run.

Continued work

The development of the SoS model is ongoing and continued work will focus on improving the model in a number of ways. One key element is the addition of dynamic science. Dynamic science is essential to model front science spawns, which move at a wide range of depths depending on underwater currents, or wind and pressure patterns above water. Incorporating these, however, will require planning with how exactly front science events move across the map, whether it be randomly or some implemented pathing system for events above and below sea level.

Secondly, the current model only has one type of AUV and UAV to choose from. Future iterations of the model will include multiple types of these vehicles, based on the data from our Portuguese collaborators. It is hypothesized that the choice of agent deployed to a specific scientific event will depend not only on the battery and data storage capacity of that agent, but also the payload of scientific instruments onboard, which vary greatly from agent to agent.

Finally, objective functions will be incorporated into the model so that the SoS can be optimized with a particular mission in mind. A critical question for the architecture of the systems of systems composed at any given point in time of the four constituent systems is to define the objective function that must be optimized. Different objective functions lead to changes in the architecture. For example, time efficient deployment of the SoS would imply that the slowest components (UAVs, AUVs) must be kept near the expected presence of the scientific phenomena. However, if the objective function is maximum resiliency (that is always able to track a phenomenon) then there are implications for the location(s) of the coordinating center as well as the locations of places from which the UAVs and AUVs are launched. Moreover, of course, an objective function which is to minimize cost would likely have a very different architecture driven by which system is the most costly to operate.

4. Attachments

- Presentations
    - Alice Cooper and Daniel Hastings, "System of Systems Concept for Effective Oceans to Near Space Observation," 2020 MIT-Portugal Annual Conference, Virtual. Ponta Delgada, Azores, October 15, 2019 (Poster Presentation).

- Publications
    - N/A

- Media (photos and video links)
    - N/A