Welcome to the WSC 2026 simulation challenge. This year, we are focusing on shipping line network and port operations. This set of materials aims to help participants gain a better understanding of the system and what is required of them.

Contents

First, we will cover the preparation material offered to participants and how they can make full use of the materials. Second, we will provide a detailed illustration of the problem and what is expected from participants. Finally, we will conclude a summary for the simulation system.

01 Preparation

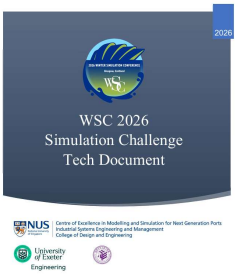
- Download Tech Document
- Download Source Code Package

Let’s start with the first part: preparation.

Preparation

Download Tech Document

- PDF Reader
- English Version



Download Source Code Package

- Decompress Package
- # Python

SimulationChallenge2026_Py.zip

Upon successful registration, participants will be provided with a technical document and a zip package containing the source code. Please refer to the document for further details regarding the descriptions on the case description, model structure, and instructions on file downloading and submission procedures. Once the source code is downloaded and unzipped, it can be viewed and executed via Visual Studio.

02 Problem Description

- Background
- Overview
- Diagrams
- Simulation Model
- Decision Scope
- Performance Evaluation
- Participant Task

Next, we have a detailed description of the problem and how the system operates.

Background

- The WSC Simulation Challenge promotes innovation in simulation-based decision-making.
- The 2026 edition aligns with the conference theme:
  Simulation for Climate Resilience
- Focus:
  climate-induced disruptions in maritime logistics networks

The WSC 2026 Simulation Challenge focuses on developing simulation-based solutions to improve the resilience of maritime logistics networks against climate-induced disruptions.

Overview

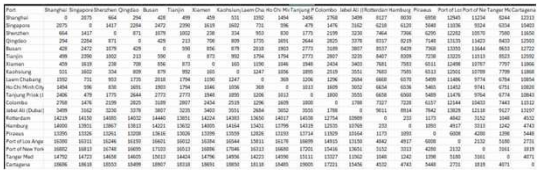
We simulate a liner shipping network with:

- Circular routes serving multiple ports
- Finite vessel capacity
- Stochastic container demand
- Climate-induced sailing disruptions

Containers have origin-destination pairs.

Mitigation decisions are determined by multiple stakeholders / agents.

20 Port Destinations



7 Service Routes

Service	NumStops	NumLegs	TotalSailingDistance_nm
S1_Asia-Europe-NorthEurope	16	15	34048
S2_Intra-EastAsia	8	7	3007
S3_SEA_Feeder	6	5	2654
S4_Transpacific_WestCoast	7	6	30563
S5_Asia-USEast	8	7	32957
S6_TransatlantIc_SouthConnector	9	8	12522
S7_IndianOcean-MiddleEast	5	4	6528

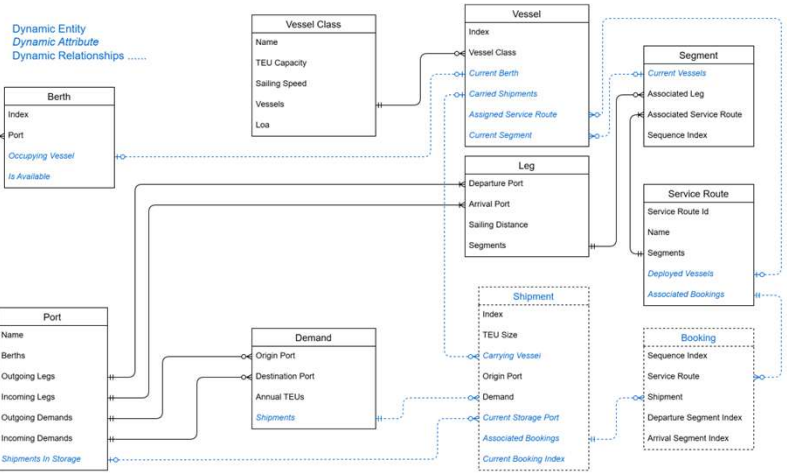
6 Vessel Types

VesselClass	Capacity_TEU
Feeder	1800
Feedermax	2800
Panamax	5000
Post-Panamax	8000
Neo-Panamax	13000
ULCV	18000

7

This simulation models a global liner shipping network in which vessels transport containers between ports under realistic operational constraints.
Key characteristics of the simulation:
20 destination ports connected by a shipping network.
7 circular service routes that visit multiple ports.
6 vessel types with different carrying capacities.
Finite vessel capacity, limiting the number of containers that can be transported.
Each container has a predefined origin and destination, while disruption mitigation strategies are determined by multiple stakeholders or agents within the shipping network.

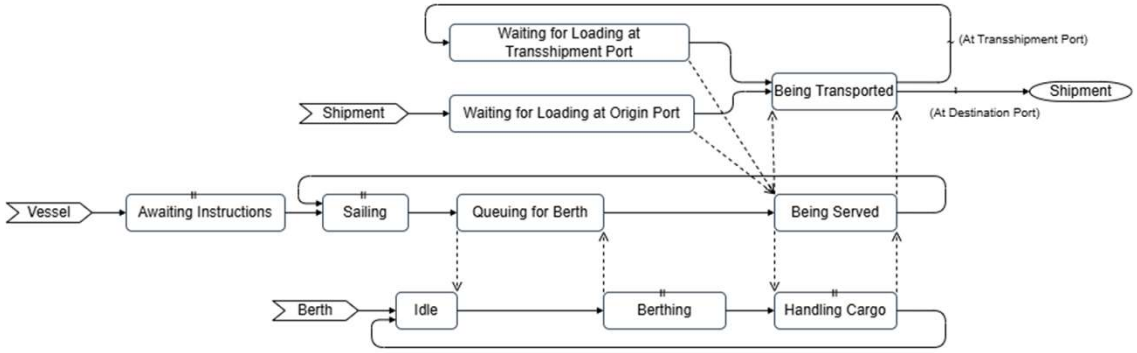
Diagrams: Entity Relationship Diagram (ERD)



8

Here is the Entity Relationship Diagram. There are ten entities and each entity has some attributes. Attributes in black represent static attributes, which are constant. Other attributes in blue represent dynamic attributes, which change over time. The details about the entity attributes and their definitions please refer our Tech Document.

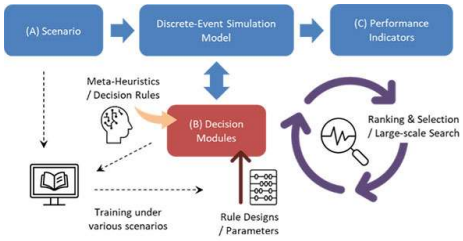
Diagrams: Event Flow Diagram (EFD)



For the event flow diagram, each entity goes through a series of activities. In the Tech Document, we describe the process of entities passing through activities and their interactions with each other.

Simulation Model: Baseline Scenario

- We will provide a baseline discrete-event simulation model (implemented in Python) with clear documentation describing the model structure under a rigorously defined modelling formalism.
- The provided package will specify three elements:
  - (A) Variable scenario parameters (e.g., real-time demand, resource availability, disruptions)
  - (B) Key operational decisions & default rules (e.g., routing/path planning, task assignment, sequencing)
  - (C) Performance indicators (KPIs) used for evaluation



Our simulation model includes baseline scenario and disruption scenario. In the baseline scenario, the simulation runs under normal operating conditions. It includes three main parts: scenario parameters, operational decision rules, and performance indicators used to evaluate your strategy. In the code, the baseline scenario is built from CSV input files in Input folder. The simulation then creates a Model, runs a warm-up period, measures daily operations, and writes KPI outputs to CSV files in Output folder.

Simulation Model: Climate Disruption Scenario

- Disruptions occur between port pairs
- Sailing time increases significantly
- Duration is stochastic
- Congestion propagates across the network

This models extreme climate volatility and systemic vulnerability.

A climate disruption scenario where bad weather delays vessel movements and causes network-wide congestion. In the model, this scenario is implemented in disruption_scenario.py.

Decision Scope

Participants can design dynamic policies at:

- Shipping company level (vessel allocation, rerouting)

- Port level (berth sequencing, priority rules)

- Cargo-routing level (adaptive path selection)

When disruptions occur, participants can design dynamic decision policies at three levels. At the shipping company level, decisions include vessel allocation and ship rerouting. At the port level, participants can optimize berth sequencing and priority rules for waiting vessels. At the cargo-routing level, participants can adapt cargo routes to ensure shipments continue moving efficiently despite changing conditions.

Performance Evaluation

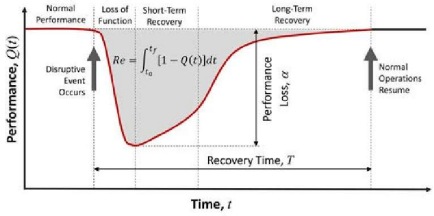
System resilience will be evaluated using a resilience curve.

Define:

$$Q(t) = \frac{\text{Average Transport Time under Normal Conditions}}{\text{Actual Average Transport Time at Time } t}$$

Resilience loss over time is measured as:

$$\int (1 - Q(t))dt$$



Participants aim to:

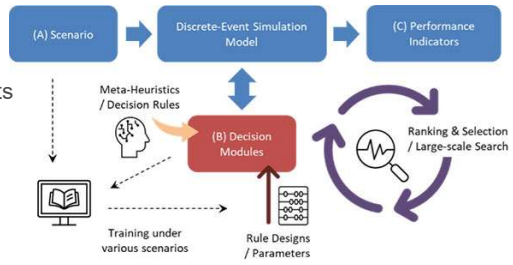
- Minimize performance degradation
- Accelerate recovery
- Reduce total resilience loss

This captures not only efficiency, but also **dynamic recovery capability under climate disruption**.

The performance is evaluated using a resilience curve that compares normal transport time with actual transport time during disruptions. Your goal is to reduce performance loss, recover faster, and lower the total resilience loss over time.

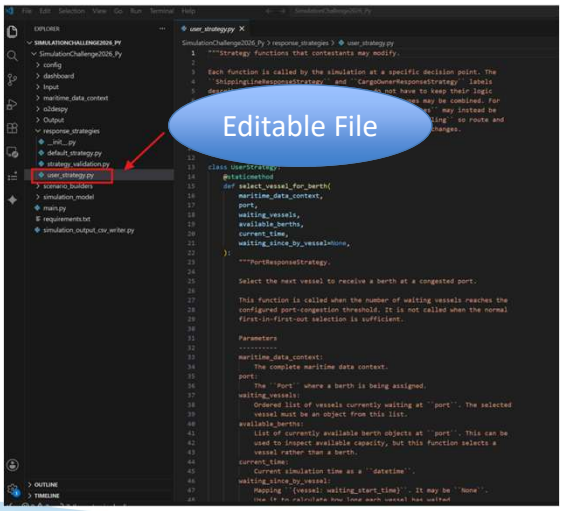
Participant Task

- Submit only the Decision Modules (B): rewrite/replace the designated decision-event algorithms to maximize performance (C) across scenarios (A).
- Approaches may include (non-exhaustive):
  - Rule-based scripts / heuristics embedded in decision events
  - Simulation embedded in external optimization / search
  - Learning-based methods to tune or infer decision-rule parameters, then embed into decision events



Participants need to improve the simulation by changing only the decision modules. You can use rules, optimization, or learning methods to make better decisions

Participant Task



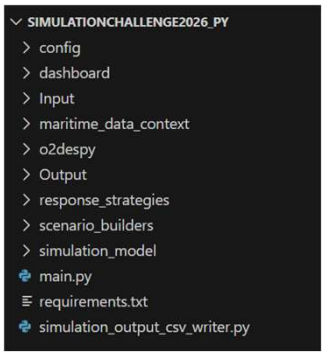
User_strategy.py is the decision module edited by participants. Here, you are invited to implement your own strategy for port, cargo owner and shipping line.

03 Summary

- Structure of Simulation System

Finally, we will show a simple summary for the simulation system.

Structure of Simulation System



Folder Name	Summary
config	Simulation configuration settings
dashboard	Browser-based dashboard files for viewing simulation outputs
Input	Input datasets for the baseline scenario
maritime_data_context	Data model classes for maritime entities
o2despy	Local bundled simulation framework package used by the project.
Output	Generated simulation output CSV files
response_strategies	Strategy files for handling disruptions or operational decisions
scenario_builders	Code that builds simulation scenarios
simulation_model	Main maritime simulation logic
main.py	Runs the simulation
requirements.txt	Lists dependencies
simulation_output_csv_writer.py	Writes simulation results to CSV files

Here is a simple summary of what each main folder contains in the simulation system.